SEVENTH FRAMEWORK PROGRAMME

Information & Communication Technologies
Trustworthy ICT

NETWORK OF EXCELLENCE

A European Network of Excellence in Managing Threats and Vulnerabilities in the Future Internet: *Europe for the World* [†]

Deliverable D7.1: Review of the State-of-the-Art in Cyberattacks

Abstract: This deliverable presents a review of the State-of-the-Art in cyberattacks. Cyberattacks are categorized in a number of main classes, and within these classes we present the most important types of attacks.

Contractual Date of Delivery	May 2011
Actual Date of Delivery	June 2011
Deliverable Dissemination Level	Public
Editor	Sotiris Ioannidis
Contributors	All SysSec Partners

The *SysSec* consortium consists of:

FORTH-ICS	Coordinator	Greece
Politecnico Di Milano	Principal Contractor	Italy
Vrije Universiteit Amsterdam	Principal Contractor	The Netherlands
Institut Eurécom	Principal Contractor	France
IPP-BAS	Principal Contractor	Bulgaria
Technical University of Vienna	Principal Contractor	Austria
Chalmers University	Principal Contractor	Sweden
TUBITAK-UEKAE	Principal Contractor	Turkey

[†] The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257007.

Contents

List of Figures

1 Introduction

This report provides an overview on some of the main categories of cyber-attacks threatening modern computer systems and networks. The topics covered in this document were selected to be complementary to the ones presented in Deliverables 5.1 and 6.1 of the *SysSec* Project, as to avoid overlaps and possible confusion.

Since the area of cyberattacks is very broad, we followed a systematic approach. Along with consultation from other experts, we divided up the space in eight broad categories:

- Memory Attacks and Exploitation Techniques
- Attacks on Devices
- Denial of Service
- Critical Infrastructure Attacks
- Social Network and Privacy Attacks
- Web Attacks
- Network-level Attacks
- Virtualization and Cloud Attacks

Topics not covered include trust, cryptography, measurement, etc. as they do not fall under the area of systems security. Also, topics of malware, the underground economy, sensor networks, etc., are covered in reports from other *SysSec* working groups, as we mentioned before.

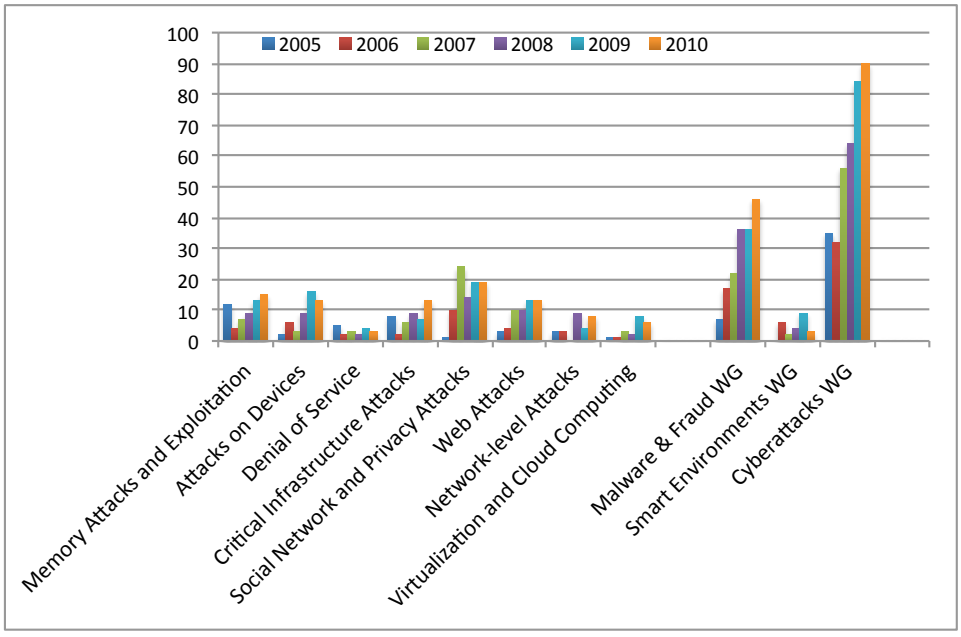
Additionally, we also set out to investigate the research trends in the area of Systems Security. This was accomplished by closely exploring the proceedings of the major and most highly rated conferences in the area of

security. Specifically, we thoroughly went over their proceedings of the last six years (2005-2010) and we mapped each paper to a category.

The publications we looked into appeared in the following conferences:

- Usenix Security Symposium
- ACM Conference on Computer and Communications Security (CCS)
- IEEE Symposium on Security and Privacy (S&P)
- Network & Distributed System Security Conference (NDSS)
- Annual Computer Security Applications Conference (ACSAC)
- European Symposium on Research in Computer Security (ESORICS)
- Usenix Workshop on Large-scale Exploits and Emergent Threats (LEET)
- Recent Advances in Intrusion Detection Symposium (RAID)

In the graph below we present the cumulative number of papers for each of the last six years, for each cyberattack category. We also present the cumulative numbers of papers per year in the specific focus topic of each of the three working groups of *SysSec*.



The reader will notice that in almost all cases there is growing interest by the research community in the topics of cyberattacks, and the topics covered by the working groups in general. One main exception is the case of the Smart Environments working group topics. The reason for this is twofold.

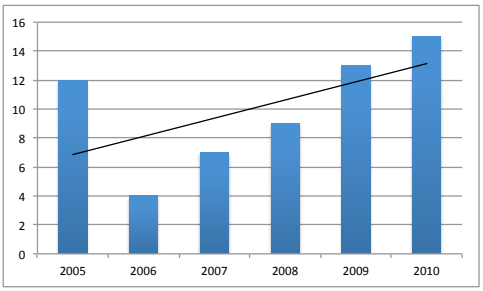
Firstly, this is a relative new field, and secondly, the publications we analyzed in our trends analysis do not typically cover these topics.

In the remainder of the document we go over each class and we present the most important attacks, and in some cases defenses, that belong in that class. It is important to note that our study is not solely focused on recent years like our research trends analysis. Instead we look at all major attacks that have appeared in the literature.

2 Memory Attacks and Exploitation Techniques

2.1 Introduction

In this section, we consider memory corruption attacks. Memory corruption attacks have been around for four decades and have been the attack vector for some of the most famous cyberattacks. In the last fifteen years especially, a plethora of defense mechanisms has been deployed to try and stop such attacks, begging the question where we stand today. We will look at the attacks from an historical perspective, and look at measures, counter-measures, counter-counter-measures, etc.



2.2 Memory corruption from a historical perspective

We provide an overview of the most important memory-corruption related events that occurred in computer security history. We distinguish separate classes of attack types and organize them in a branched timeline. Each class will be discussed in more details in the following sections.

Consider Figure 2.1 for our general timeline. Although not every event is listed in full detail, the timeline shows us that a lot happened over the years.

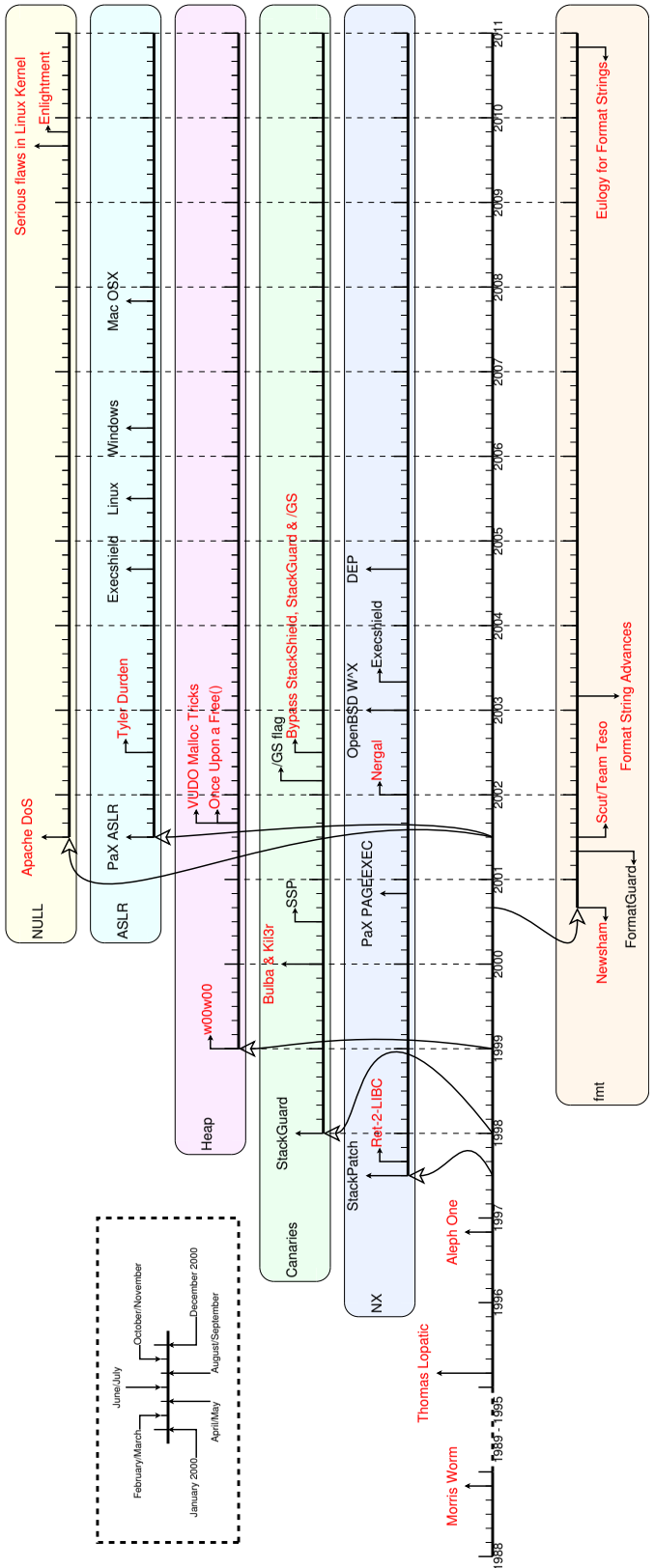


Figure 2.1: Timeline of memory corruption attacks

One of the first documents to discuss memory corruptions attacks was published in 1972 by the Computer Security Technology Planning Study Panel. It mentions the possibility of obtaining unauthorized data for a user by supplying addresses outside the space allocated to that user's programs. The first large-scale memory corruption attack that started the whole memory corruption security business was the Morris Worm, released on November 2nd, 1988[301]. To propagate his worm to other hosts on the net, Robert Morris Jr. used a buffer overflow vulnerability (the C `gets()` function) in the fingerd daemon. The Morris Worm turned out to be a wakeup call for many system administrators and programmers, and led to establishment of the first CERT. Many more were to follow.

Specifically, CERT was formed late November 1988, which provided a patch for a hole in `passwd.c` late January, 1989 . Barton Miller et al. published an empirical study . of the reliability of UNIX utilities in late 1990, in which they could crash almost 25% of these utilities by just providing random input strings.

Meanwhile, BUGTRAQ was formed in May 1993. Shortly after, Thomas Lopatic made an interesting contribution to the security world by demonstrating an exploit in the NCSA http daemon on BUGTRAQ. He posted the steps he took to exploit the daemon, which illustrated that the lack of bounds checking could be disastrous [208]. Shortly after Lopatic's post, a private note showed up in which the author (Peiter Zatko a.k.a. Mudge) describes how to write a buffer overflow exploit. More than Lopatic's post, this document describes a step by step approach about how to exploit a buffer overflow.

Without a doubt, however, the most referenced paper on memory corruption attacks is Aleph1's paper, released in August 1996: Smashing the Stack for fun and profit[23]. Together with Mudge's personal notes, this paper pioneered in the world of memory corruptions and triggered a Bugtraq discussion on stack smashing prevention in early 1997. The early years of memory corruption belonged to the stack, but in January 1999, Matt Conover and the w00w00 security team were the first to publish a paper on heap overflows [76].

The security issue discussed above led to our first branch in the timeline: the introduction of the NX bit which marks code as writeable or executable, but not both. The first implementation of an NX-like system was proposed by Solar Designer in 1997. The entire lifecycle of NX-like systems will be discussed in Section 2.3.

In January 1998, another defense mechanism, other than the non-executable stack, was introduced by Chris Cowan: StackGuard [82]. More details in Section 2.4.

In mid September 1999, a completely different type of memory corruption was demonstrated by Tymm Twillman [333], who posted an e-mail to

bugtraq concerning a format string bug in proftpd. Format strings are covered in Section 2.5 of this document.

A different defense mechanism in which code is randomized each time a program is started, was developed by the PaX Team and first released (according to WikiPedia) in July 2001. PaX ASLR and the various implementations of ASLR for Linux, Windows and Mac OS X are discussed in Section 2.6.

The first vulnerability that concerned a NULL pointer dereference was submitted in May 2001. It took attackers up to 2008 to exploit such dereference. The how and what behind NULL pointer dereferences are discussed in Section 2.7.

2.3 Non-Executable Data segments

Buffer overflow attacks are characterized by overrunning the buffer and overwriting the adjacent memory locations. They mostly take advantage of programs requiring user input. Stack based buffer overflow exploits typically involve writing beyond the buffer size, and overwriting the function's return address on the stack to point to some user-controlled code, which is also present on the stack.

Most of the original solutions to extenuate buffer overflow attacks by making stack non-executable used the segmentation feature of x86 to make the text segment executable, while setting the data area (up to the end of userland) as not executable.

A first attempt to mitigate stack-based buffer overflows came out in June, 1997, when Solar Designer released the Non-Executable Stack patch for Linux [98, 96]. The idea was simple and intuitive: make the stack area non-executable. In theory, doing so renders most of the buffer overflow vulnerabilities harder to exploit. To execute code the attacker should either find a way to turn off the execution protection or place his shellcode in an unprotected memory region. Unfortunately, it is not hard to find unprotected memory regions to place your shellcode (like the heap). In addition, we shall see presently that other approaches like return-to-libc are also able to evade the stack execution protection. Thus, this protective measure by no means is a complete solution and can only prevent canonical approaches of stack-based buffer overflow. However, most of the desktop processors today come with hardware support for the no-execute flag.

Later in August, 1997 Solar Designer himself posted a return-into-libc overflow exploit[97]. This post described the classic return-into-libc technique. Rather than returning to the code located on the stack, the vulnerable function should return to the memory location of a dynamically linked library. In essence, the return address on the stack is replaced with address of a instruction (which is generally a dynamic library) and adjacent mem-

ory regions are filled with arguments to be provided to that instruction. This attack is called return-to-libc because return address is mostly replaced with `libc` shared library. Typically, the instruction/function will be the `libc` `system()` function, and the first argument will be `"/bin/sh"`.

In November, 2000 The PaX Team (with its anonymous prinicipal coder) released the PaX project[323]. PaX prescribes that for all pages where data reads and writes are allowed, code execution should cause a page fault. In addition, to combat return-to-libc attacks, PaX randomized the mmap base. The randomization causes the first loaded library to be mmapped at a random location. The same is true for the program's stack. As a result, PaX prevents direct code execution and makes return-to-libc attacks harder to exploit. As most current processors have hardware support for the NX bit, PaX can safely use it in a secure and efficient way. We will discuss randomization in more detail later (Section 2.6).

The NX bit and PaX have been important milestones in protecting systems from memory corruption attacks. As we shall see shortly, however, they do not prevent them completely. Nevertheless, these features were quickly adopted by most major operating systems.

OpenBSD version 3.3 release in May, 2003 featured various buffer overflow solutions [94]. One of these is what OpenBSD terms W$\oplus$X, which means a process cannot have memory which is both writeable and executable. A related change was that OpenBSD made its read-only data a separate `.rodata` segement with `PROT_READ` permissions (unlike historical implementation with `PROT_READ|PROT_EXEC` permissions).

By this time all major operating systems were picking up on measures against buffer overflows. Red Hat introduced new security enahancements to combat buffer overflow attacks in its Enterprise Linux Version 3 [334]. It featured a new kernel based security solution "Exec Shield" [234]. Unlike other solutions, like the non-executable patch by Solar Designer, the new solution tried to make non-executable the largest possible part of the virtual memory, not just the stack. In essence, they were also using the x86 segment limit feature to enforce, mostly, data and stack non-executability. However, the segment limit puts the burden on the operating system to ensure that all program code must be below this limit, while the data and stack areas should be at higher virtual addresses. This is not convenient. So, for the Intel and AMD NX-enabled processors, Exec Shield would use the NX bit instead of virtual address segmentation. In addition, Exec Shield also randomizes some critical components of the program stack, the location of shared libraries, and the start of programs heap.

Microsoft Windows XP service pack2 (SP2) released in August, 2004 also introduced non-executability for data areas, under the name DEP (Data Execution Prevention) [227]. DEP was implemented on two layers: Hardware-enforced DEP and Software-enforced DEP. Hardware-enforced DEP takes advantage of no-execute-page-protection (NX) provided by AMD and Exe-

cute Disable bit (XD) provided by Intel in their processors to mark a program's memory locations as non-executable unless it explicity contains executable code.

2.4 Canaries

Canaries are used by vendors to secure their products against buffer overflow attacks. The solution involved modifying compilers to change the structure of data in memory in order to provide enhanced security.

StackGuard, one of the first solutions to use canaries was announced on December 18, 1997 [79] by Crispin Cowan and released on January 29, 1999 [82]. The idea presented is to place a canary value on the stack just below the function return address. When the function returns, this value is then compared to the saved copy of the canary—which is saved when stack is prepared for a function call. If the values do not match, the function return address is assumed to be overwritten and appropriate steps are taken to terminate the process.

This solution clearly changes the data organisation in a stack frame to include an extra "canary" value. It provides protection (only) against stack based buffer overflow attacks while leaving the system vulnerable to many other kinds of overflow attacks. Even so, similar concepts can be applied to Heap-based overflows also. StackGuard was made available as a standard part of the Immunix Linux distribution from 1998 to 2003, providing both Red Hat-compatible binary RPMs and patched GCC sources from GCC 2.7.2.3 through 2.96. Stackguard was suggested for implementation in GCC in 2003 [340].

StackGuard assumes that to over-write the function return address, one will change the canary value (placed before return address) also. As we shall see shortly, this is not always true. StackShield [313], released later in 1999, therefore tried to address this issue by concentrating on the return address itself. As stated on its webpage, the StackShield protection system copies the return address in an non-overflowable location (the begining of the DATA segment) on function prologues, and checks if the two values are different on function epilogues (before the function returns). If the two values are different the return address was modified and StackShield terminates the program.

Both of the above compiler-based solutions have weaknesses in protecting against overflow attacks. Phrack released a "BYPASSING STACKGUARD AND STACKSHIELD" article in January 2000 [56] which explained various shortcomings of StackGuard and StackShield, as well as ways to exploit them to evade these protections (even in hostile envrionments like when the stack is non-executable).

As mentioned above, it is not always necessary to overwrite canary values to change the function's return address. For instance, if we have a pointer `p`, physically located above buffer on the stack and later a copy function (`strcpy`, `memcpy`, etc) which copies from user specified data to `p`, we can overflow the buffer to only overwrite the pointer value to make it point to the return address on the stack. The next copy function will then overwrite the RET address, through this pointer, without touching the canary value. This kind of attack can be easily performed against StackGuard protected systems, but not against StackShield. StackShield protection can be cracked by overwriting specific things, using a pointer, like functions in *fnlist* structure which are called when a program calls *exit()* function (usually when an error occurs, which we can easily force). Other exploits involve overwriting a libc function's GOT entry by our shellcode address.

David Litchfield [206] introduced a novel approach to bypass StackGuard prevention mechanism on Windows systems. On Windows systems, when a canary value does not match the saved value, an exception handler is invoked to take the appropriate actions. The exception handling is mantained by a series of structures on the stack. This structure includes a function pointer to the current exception handler and this function pointer is called when an exception is raised. Interestingly, we can still gain root (administrator) access to the system by overflowing the buffer to such an extent that we can overwrite the exception handler function pointer, and have our own function called instead of actual exception handler.

ProPolice or Stack Smashing Protector (SSP) [118] is an enhancement of StackGuard concept, as it prevents the corruption of pointers, which can be further used to corrupt arbitrary memory locations, by:

- Reordering of local variables to place buffers after pointers to avoid the corruption of pointers;
- Copying of pointers in function arguments to an area preceding local variable buffers

SSP was implemented as patch to GCC 3.x and was included in GCC 4.1 release. FreeBSD, OpenBSD, DragonflyBSD and Ubuntu uses ProPolice as a standard for stack smashing protection.

Microsoft's /GC compiler flag[53], introduced in .NET compiler, also protects the *frame pointer*, by placing a random canary, called a *cookie* between it and local variables. The function prologue is modified to call instructions to fetch a cookie, and then placing the result of a XOR of the cookie and the return address value directly below the return address on the stack. The stack smashing check is done by the epilogue which issues instructions to retrieve the cookie from the stack, XOR it with the return address value and compare the result with the saved cookie value. If the values do not match, the appropriate exception handler function is called. Mi-

crosoft's initiative to protect the *frame Pointer* as well was helpful in combatting *Frame Pointer Overwrite Attacks* [**?**], where the frame pointer is changed and later, in a second return, the execution flow is hooked.

Finally, Gerardo Richarte [284] presented various advanced tricks to bypass StackGuard, StackShield and /GS by altering Local variables, functions arguments, saved frame pointers.

2.5 Format String Attacks

Format String attacks refer to the vulnerabilities introduced in the code written in programming languages like C which lacks type safety. The issue typically arises from a C function using unchecked user input as the format string parameter. The most commonly quoted example of such a mistake is when programmer writes `printf(buffer)` as short-hand for `printf("%s", buffer)`. The former is vulnerable to % directive since it interprets `buffer` as a format string. If the user passes any format parameter like %x, %d etc., these directives will be parsed by *printf* as formatting instructions, values will be popped from the stack and printed. The result of such vulnerability could end up having exploits generating similar results as of buffer overflow vulnerabilities.

Formar string issues were published in September 1999 while auditing the source code of ProFTPD [333]. By 2000 a lot of such vulnearabilities in other utilities started surfacing on security mailing lists like "Wu-Ftpd Remote Format String Stack Overwrite Vulnerability" posted on Bugtraq in June, 2000 [55]. Amidst the growing number of format string bugs, Tim Newsham [251] published a document which succinctly put down some fundamental concepts behind the attack and various implications of having such vulnerabilities in your code.

One of the little-known features of `printf`, which as Tim put it "isn't taught to us in school", plays a vital role in performing this attack. The `%n` format parameter allows one to get the number of bytes printed by `printf` at any point, and can be written to the corresponding argument, of type `int *`, in `printf`. Moreover, it returns the *number* of characters that should have been output, not the characters output so far. This means that even if the user string is truncated while formatting it for a fixed-size buffer, `%n` will give the count as actual string length.

The problem with this directive is that it assumes that there is a corresponding argument supplied in `printf` to store the count, even if none is supplied. As a result, the attacker can scroll up the stack (e.g., by giving a couple of `%d` directives), and after reaching a suitable address or location (e.g., the return address) on the stack he can write at that location using the `%n` directive. Thus, one implication of this "feature" is that an attacker can write arbitray values to an arbitrary addresses in the victim's program ad-

dress space (also known as a write-anything-anywhere primitive), imposing a big security threat. One of the hurdles in exploiting this vulnerability is to get the precise value of all the offsets.

WireX Communications, Purdue University and CERN collectively published a research paper in May 2007 introducing *FormatGuard*, which provides protection against format string bugs [81, 80]. As mentioned earlier, one of the key reasons for format string vulnerability is C's varargs mechanism's type unsafe behaviour. As a result, the attacker can provide a number of bogus % format directives in the user input that is subsequently used as format string by `printf` call, resulting in popping bytes off the stack without checking the type or even the existence of the arguments. The solution proposed by the team was to compare the number of arguments provided by format string (supplied by the user) with the number of arguments passed to `printf`–by means of static analysis. If the arguments presented to `printf` are less than the arguments in format string then it is considered as an attack, and the process is killed.

Unfortunately, like any other preventive measure, this approach also has a few shortcomings; it will not work if the attacker's format string undercounts or matches the `printf` argument count, or if some program dynamically constructs a variable list of arguments, which would result in failing to count the number of argument by static analysis.

One of the most extensive articles on format strings, discussing traditional format string attacks and introducing some variations, was published by Scut/ Team Teso in July 2002 [302]. Along with detailing conventional format string exploits like viewing process memory (stack or any other memory location) and over-writing arbitrary memory location in process address space, it also presented some novel tricks to exploit the vulnerability.

A response-based brute force attack, which takes advantage of printed format reply from the program, and blind brute force attack can be used to overcome the problem of finding exact offsets. Another trick presented was to overwrite alternative locations, instead of a program's memory, like GOT (Globa Offset Table), DTORS, etc. Overwriting a GOT entry, using a format string vulnerability, with the address of our own code will let the program run our code whenever it calls the function corresponding to the table entry. This hack will bypass any stack protections based on checking return addresses or canaries. Various other techniques like return-into-libc, and storing format strings on the heap can also be used to make these attacks resilient against protection facilities like StackGuard, StackShield, and non-exectuable stacks. Some advanced format string based attacks are discussed in [286, 265]

MITRE's CVE project lists roughly 500 format string vulnerable programs as of June 2007, and a trend analysis ranks it the 9th most-reported vulnerability type between 2001 and 2006.

2.6 Address Space Layout Randomization

To exploit a buffer overflow, an attacker must have knowledge about what is where in the program's address space. For example, to exploit a vulnerable program by returning into a specific buffer which contains shellcode, the attacker must know the approximate address of this buffer on the stack or heap. To exploit a vulnerable program by using a return-to-libc attack, an attacker must know the address of the libc function to return to.

Based on this observation, security experts came up with the idea of adding a random offset to the addresses of those (and other) components. As mentioned earlier, the PaX Team was one of the first to implement this technique in their kernel patch and called this Address Space Layout Randomization, or ASLR.

According to its documentation, the goal of PaX Address Space Layout Randomization (ASLR) is to "introduce randomness into addresses used by a given task. This will make a class of exploit techniques fail with a quantifiable probability and also allow their detection since failed attempts will most likely crash the attacked task". In this section, we will take a closer look at how ASLR works and which important ASLR related events occurred after its first introduction. For a detailed timeline of ASLR, consider Figure 2.2.

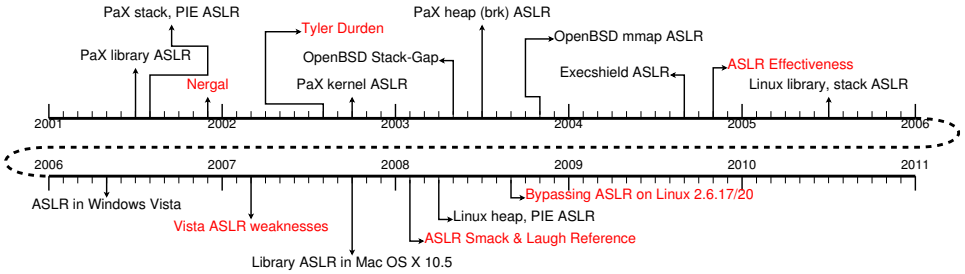


Figure 2.2: Detailed timeline for Address Space Layout Randomization

The PaX Team was one of the first who implemented ASLR. They added "mmap randomization" as a new feature to their Linux kernel patch in June, 2001[1]. After this first implementation, other regions of the address space got randomized by PaX as well:

- `mmap` randomization. Released by PaX in July, 2001. When randomized mmap() base is enabled, dynamically linked code like shared objects will be mapped at a different, randomly selected offset each time. This causes library functions to be located at a different address each time a program is executed and hence makes return-to-libc attacks difficult.

[1] http://en.wikipedia.org/wiki/PaX

- Stack randomization. Released by PaX in August 2001. Stack randomization makes shellcode attacks harder, since the attacker is no longer able to find his payload at a hard coded address each run. It also protects against return-to-libc attacks that use injected fixed width stack frames.
- PIE randomization. Released by PaX in August 2001. A Position-Independent Executable (PIE) is solely build of Position-Independent Code (PIC) so that it can be loaded in any location in each program address space. This reduces the chance of working return-to-libc attacks that rely on knowledge of the offset of the executable code in binary.
- Kernel stack randomization. Released by Pax in October, 2002. Kernel stack randomization protects the kernel stack in a similar way as the user stack when 'normal' stack randomization is applied.
- `brk` randomization. Released by PaX in July, 2003. This randomizes the location of the heap (brk).

Since the PaX ASLR implementation was released as part of their Linux Kernel patch, users had to patch their kernel first, before they could benefit of the advantages that ASLR comes with. It took the operating system community some time before ASLR was adopted in their main kernel.

2.6.1 Implementations

OpenBSD became the first mainstream operating system to support ASLR (and to activate it by default[2]. Theo De Raadt stated that stack randomization was to be released in OpenBSD 3.3 (May 1, 2003). This version also included ProPolice and WX̂[94] . OpenBSD 3.4 included library ASLR as well (released on November 1, 2003)[95]. OpenBSD does not support kernel stack randomization, since it would break POSIX standards. In fact, an intense discussion between De Raadt and the PaX Team about who was first with randomization and NX and whether or not standards were broken, started late April 2003[147, 324, 299, 310].

The Red Hat ExecShield project (also discussed in Section 2.3 added ASLR support in August, 2004[334]. Version 3, update 3 supports stack, library and heap randomization. It also support Position-Independent Executables to be randomized.

The Linux kernel enabled stack and library randomization by default ever since their 2.6.12-rc1 kernel, released on March 18, 2005 [335]. Linux has stack and library randomization enabled by default. The first patches for randomization were released on January 27, 2005 [336], by Arjan van der Ven, who, at that time, was also working on the Red Hat ExecShield

[2] http://en.wikipedia.org/wiki/Address_space_layout_randomization#OpenBSD

project. It took Linux until April, 2008, before heap and PIE randomization were available as well (Linux Kernel 2.6.25) [328]. Although it still lacks kernel stack randomization, it is said that Linux now supports "full" ASLR, by supporting stack, library, heap and PIE randomization[62].

Microsoft added ASLR support to their new Windows operating system as well. The first Windows version with stack, heap and library randomization was Windows Vista Beta 2, released on May 26, 2006 [157]. This version also provided in something that looks like PIE (/SafeSEH compiler flag). Shortly after this release, Ali Rahbar stated in his analysis of Microsoft Windows Vistas ASLR that there are some bugs in its implementation [279]. In a response, Howard refuted these accusations [158].

Later, when Windows Vista was officially released, another analysis of ASLR on Windows Vista was done by Whitehouse [344]. He concludes that "the protection offered by ASLR under Windows Vista may not be as robust as expected". It is uncertain what happened after this when Windows 7 was released. Although the deficiencies in the ASLR implementation were acknowledged by Microsoft, there is little to find about any follow up. In June, 2010, however, Alin Rad Pop published a paper which discusses the use of ASLR and DEP in Third-party Windows Applications. He concludes that those third-parties are slow in adding ASLR support to their applications. In June 2010, only Google Chrome and Adope Flash Player gained Full ASLR protection. Popular applications like Adobe Reader, Mozilla Firefox and Apple iTunes did not have full ASLR support[267].

Apple introduced partial ASLR support in Mac OS X 10.5, called library randomization[173]. Only library functions are randomized on Mac OS X; full ASLR is not supported. The lack of full ASLR support brings some security risks that were not fixed in later Mac OS X releases.

2.6.2 Attacks

One of the first attacks on ASLR and NX was mentioned by Nergal in his PaX Case Study [250]. It focuses mainly on bypassing NX, but the second part is devoted to "methods of bypassing PaX in case of stack buffer overflow (other types of vulnerabilities are discussed at the end). The recent PaX improvements, namely randomization of the addresses at which the stack and the libraries are mmapped, pose an untrivial challenge for an exploit coder. An original technique of calling directly the dynamic linker's symbol resolution procedure is presented. This method is very generic and the conditions required for successful exploitation are usually satisfied." This attack is also known as a return-to-plt attack. Later, when PaX released their PIE randomization feature, executables that were compiled as PIE, would be unaffected by Nergal's attack.

In 2002, Tyler Durden showed that certain buffer overflow vulnerabilities could be converted into a format string vulnerability, which could then

be used to leak information about the address space of the attacked executable [105]. This type of information-leakage would be the standard for upcoming attacks on ASLR.

In 2004, Shacham et al showed that ASLR implementations on 32-bit platforms were limited, due to the number of available bits for randomization [305]. They successfully exploit a buffer overflow vulnerability in Apache by brute forcing every possible memory address.

Tilo Müller later provides an in-depth discussion about attacks against ASLR in his paper called "ASLR Smack & Laugh Reference" [237]. He mentions return-to-text, return-to-bss, return-to-data, return-to-heap, return-to-return, return-to-pop, return-to-esp, return-to-eax, return-to-got and many other attack types. Some of these techniques like return-to-text or return-to-got are also useful to bypass the NX bit. He concludes that "ASLR and therefore e.g. a standard linux installation is still highly vulnerable against memory manipulation. But ASLR is complementary to other prophylactic security techniques and a combination of these technologies could provide a stronger defense." Note that Linux implemented heap and PIE ASLR only two months after this paper got published.

FHM Crew wrote a different attack that exploits the fact that the linux-gate shared library in Linux Kernel 2.6.17 was not randomized. They looked for a `jmp esp` instruction in this shared code, which would redirect EIP to the stack [128].

Mark Dowd and Alex Sotirov publish a paper at BlackHat USA titled "Bypassing Browser Memory Protections". Amongst many other attacks they make use of .NET controls embedded in a page to load shellcode into executable sections of memory. From the paper: Since the .NET binaries have the same basic format as PE files, the CLR maps them into memory as images. This means that the kernel parses the PE header and loads all PE sections in memory the same way it does for normal executables or DLLs. In doing this, it sets the page permissions for each section according to the flags in the PE header. If the binary contains an executable section, it will be loaded in memory and its pages will be marked executable. They also covered a number of different spraying techniques to bypass ASLR, including .NET spraying.

Finally, Dion Blazakis presented a talk on bypassing DEP and ASLR on Windows Vista using implementation details of the Flash Player virtual machine and Pointer Inference. Pointer inference uses an indirect method (ordering of a Dictionary iteration) to disclose heap addresses of runtime objects. JIT spraying uses predictable code generation patterns to construct shellcode in executable memory bypassing DEP.

2.7 NULL Pointer Dereferences

A NULL pointer dereference occurs when a pointer is dereferenced expecting it to be pointing to a valid memory area, but being a NULL pointer it causes the application to crash or exit.

A pointer, by definition, stores a reference to another value. In other words, it simply refers to a value stored somewhere in physical memory using its address—and in the case of a NULL pointer that value is 0. By convention, any attempt to read or write via this pointer leads to *segmentation faults*; thus rendering it to be an invalid pointer.

The vulnerability arises from the fact that the kernel's code and data segment both have a base zero (kernel 2.0s didn't follow the same convention; its data segment was above PAGE_OFFSET), and any existing kernel code which makes a null pointer dereference will ask the kernel to access page zero, a page which can be memory mapped to a region filled with bytes of our choice—e.g., a simple shellcode to gain root access).

A large number of such vulnerabilities have been reported recently. In June, 2009 Julien Tinnes and Tavis Ormandy [326] reported a workaround to evade the check performed by the LSM hook. The relevant code in *security/capability.c* prevents the pages below `mmap_min_addr` to be memory mapped. The observation they made was that the process with CAP_SYS_RAWIO could bypass this check. They managed to do so by running a setuid binary, and eventually succeeded in mapping a zeroeth page. Further, the mapped area can be grown using *mremap* and access rights can be gained over that memory region using *mprotect*. They also published a patch, to fix the issue, which adds the MMAP_PAGE_ZERO to the PER_CLEAR_ON_SETID mask, which was later included (in July) in Linux kernel 2.6.31-rc3 and was also included into -stable.

Later, Brad Spengler [311] also published an exploit which not only works on systems without SELINUX using *CR0* exploit, but also on Redhat systems with default SELINUX policy. A bug was present in the Fedora and RHEL5 SELINUX policy which makes `unconfined_t` ignore the boolean disallowing the access to mmap page zero [282].

Brad found that the `unconfined_t` domain was ignoring the boolean and is always allowed to `mmap_zero`. Consequently, a default logged in user on a targeted SELINUX system has the ability to `mmap_zero`. Later, this boolean bug was fixed in following SELINUX versions: selinux-policy-3.5.13-66.fc10, selinux-policy-3.6.12-66.fc11, selinux-policy-3.6.22-1.fc12.noarch, selinux-policy-2.4.6-253.el5.

The exploit written by Brad was interesting in many aspects and relies not only on existing kernel code but also on default compiler optimizations. Virtual network devices required by many applications such as virtualisation, virtual private networks etc. are provided by the TUN/TAP device driver. A patch introduced in TUN/TAP driver for the kernel's packet ac-

counting mechanism added a more severe problem. A good programming practice is to avoid dereferencing pointers which might be NULL, but in this patch a line was added which dereferences the pointer prior to the check. As the check is still there, normally this would not be of any help to the attacker. But this is where the GCC compiler comes into the picture. GCC, by default, will optimize the NULL test out considering the fact that since the pointer has already been dereferenced, it cannot be NULL. A more detailed explanation of the exploit can be found on this LWN article [78]. Brad also wrote a reliable shellcode which should work, essentially, on all x86 and x86_64 machines.

Dan Walsh, interestingly, reported that even correcting the SELINUX boolean was not enough to stop the attacker [341]. He noticed that a user logged in with the `unconfined_t` domain can compile the executable provided by Brad and run the script, which gives you the *mmap_zero* privilege, by using runcon to change its security context from `unconfined_t` to `initrc_t` and, then, to `vbetool_t`.

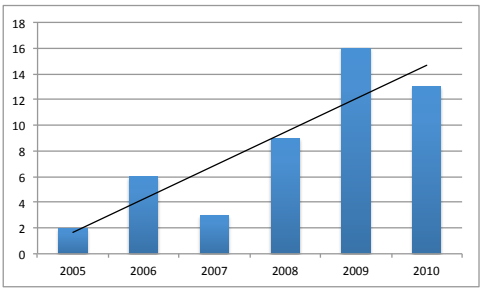
In August, 2009 Julien and Tavis came up with another Linux kernel vulnerability concerning the way Linux deals with unavailable operations for some protocols [327]. `sock_sendpage` and others don't check for NULL pointers before dereferencing operations in the ops structure. Instead the kernel relies on correct initialization of those `proto_ops` structures with stubs (such as `sock_no_sendpage`) instead of NULL pointers.

Because of the flurry of activity on these vulnerabilities, some security observers termed 2009 as the year of kernel NULL pointer dereference.

3 Attacks on Devices

3.1 Introduction

In this chapter, we look at attacks against different kinds of *devices* ranging from mobile phones to smart cards. People nowadays tend to use a wider range of such devices than just a decade ago. These devices, including the ones that have been with us for a long time, are furthermore equipped with some communication capability, be it through a short-range radio or through a connection to the Internet, often encouraging social interactions. This, in turn, means that they are more likely to be attacked.



We have on purpose adapted a broad use of the term *device* to show a survey of the many types of attacks that are possible. For example, below we discuss *smart phones and other PDA-like devices* including tablets, *consumer electronics and small office equipment*, such as printers, routers but also TV streaming devices and photo frames, *RFID and sensor nodes*, *vehicles* (and the devices therein), *medical devices*, and more. The group is not homogeneous, but different types of devices have different security issues. For example, leaking past locations of a stationary set-top box is less of a concern than it is for a mobile phone. A router always connected to a power outlet may run more advanced protection mechanisms than a cell phone that is being carried in a pocket for most of the day. On the other hand, the user may update the firmware of the cell phone regularly to get the latest features, while never updating the router and thus leaving it with old vulnerabilities.

Some of these devices, notably the mobile phone, are always with us, always on, and having a range of sensors including GPS, gyroscope, microphone, camera, compass, light sensor, etc. The cell phone can also communicate over GPRS, through SMS/MMS, over Bluetooth, using local wireless networks, as well as ANT. It also has large storage[1] and the capability to perform financial transactions, i.e. placing a call or sending an MMS. Many people also use it to access their banks or buying an *app* in the phone marketplace. On the other hand, it runs on battery and any protective mechanism needs to consider its power usage. The RFID tag, on the other hand, may also be carried with us by being attached to clothing or a driver license, but it is much more restrictive in its computing capabilities. Yet other devices, such as pacemakers, are often overlooked from a security perspective as they are developed by engineers from other disciplines who may not be aware of the security implications of wireless communication.

The attacks against these devices are sometimes similar in nature to a regular computer – after all, a modern advanced smart phone is like a portable computer. Such devices can be attacked through the built-in web server as described in Chapter 9. Others can be exploited in similar ways to the attacks described in Chapter 2, especially because they may lack the latest hardware-supported memory protection mechanisms. Several factors make the devices and their environment unique and thus we devote this chapter to describing the attacks and some of the more recent defenses developed in the research community.

These devices can often not support a complete security solution as found on a regular computer. They may lack proper operating security mechanisms [192], run with old and vulnerable firmware [297] with no mechanism for patching, or have a protocol stack that contains many vulnerabilities [151]. Sometimes the owner is not even given complete control (*root*) over the device, meaning that he cannot check all parameters of the device to investigate whether malware is running.

Many of the attacks and protection schemes we found described in recent academic conferences are focused on user privacy, often the inadvertent disclosure of the location of the user. This include attacks against cell phones, the use of RFID tags, but also toll devices mounted in cars. The tracking of devices actually highlights an interesting issue of conflicting goals – many times it is advantageous to track a device in some circumstances but not in others. For example, the owner may want to see the path taken by goods in a supply chain, but this information should not be leaked to the adversary. A user may not want his phone (and thus himself) to be tracked, unless it is stolen. A person may be willing to disclose his location automatically, but only in an emergency or to trusted friends.

[1] With the large storage in modern phones, it would be possible to record a voice conversation and later transmit it when the user connects to an open network.

Botnets of DSL routers have been created [67], and researchers have investigated the feasibility of worm propagation using only Bluetooth [308] or WiFi [22]. Such botnets can be used for denial of service (DoS) attacks and, somewhat surprisingly, it has been shown that in special circumstances the DoS attack does not depend so much on available bandwidth as with problems when a cellular network is connected to Internet [332, 331]. Another surprising fact concerning malware directed at mobile phones is that several versions have actually been installed by the user. They have been disguised as Trojans in the phone marketplace [304], or just repeatedly asked to be installed until the user gave up [120].

In the following, we give an overview of recent research and attacks. Most of the recent research has been focused on mobile phones, RFID tags and sensor networks. As later deliverables will summarize the state of the art of security for low-capability devices, this section is quite brief in this report. We also give an historical look backwards on mobile phone malware to set the scene for the discussion of the research results.

3.2 Historical attacks against the cell phone

With the evolution of cell phones into smart phones, with Internet connectivity and the capability to execute third-party software, it was just a question of when the first malware would appear. Kingpin et al. [192] pointed out early on that even basic OS security mechanisms were missing from PDAs. A massive worm was predicted to happen "soon", by several experts [168] but even though there have been many types of malware, no outbreak has reached a critical mass *yet*. A surprising fact is actually that early malware for the cell phone required the user's approval before it could be installed. This was achieved either through social engineering or by constantly asking the user for permissions [120, 168].

In June 2004, *Cabir* [120, 168] – a proof of concept worm – spread into the wild. It used Bluetooth connection on Symbian Series 60 to replicate itself and infect other victims, but it did not destroy or steal any information. However, variants soon appeared that were both more malicious and used other communication techniques, such as SMS and MMS, to spread faster. Later types of malware even appeared to be used for the financial benefit of the hackers themselves.

After Cabir, the first malware defined as a Trojan for smart phones appeared in August 2004. The Trojan, *Mquito*, was spreading in a cracked version of a mobile game. By late 2004, another Trojan was seen in the wild, named *Skull*. This Trojan was again hidden as a legitimate application. These two types of malware targeted Symbian phones, but the malware *WinCE.Duts* also appeared in 2004 targeting the Windows CE platform. It was followed by *WinCE.Brador*, which had more advanced features such

as the ability to receive commands from its attacker over an Internet connection.

It did not take long until cross-platform malware appeared, using the fact that the phones are often connected to regular computers [168]. The *Cardtrap* Trojan appeared in September 2005 and infected Symbian 60 series smart phones. However, it also installed a Windows malware on the memory card. When the user inserted the infected memory card in a computer, the Trojan could misuse system resources and access personal data.

Another Trojan named *RedBrowser* was discovered in 2006. It was disguised as a free WAP browser and used social engineering to be installed. It was unique in three ways, according to F-Secure [121]. It sent premium-rate SMS messages to a Russian number, thus using the capability of the phone to perform financial transactions to steal money. It was also the first malware using J2ME and it worked on many low-end phones. In 2006, Kaspersky measured the prevalence of malware in MMS message, and they found that about 0.5%–1.5% of all MMS were infected.

In 2008, *InfoJack*, another Trojan for the WinCE platform, appeared inside installer packages for smart phones [122]. After the initial infection, the malware downloads the rest of the code over an Internet connection. The Trojan then sends the user information available on the device to its home website.

For the Symbian 60 series, another worm called *Sexy Space* was detected in 2009 [123]. The infected phones sent SMS messages to the numbers in the address book with a link to install an application to access pornographic contents. The unique aspect of the Sexy Space is that it was the first malware *signed by Symbian*. A signed application could be installed into the systems without raising user warnings, thus making the malware less suspicious.

From being a proof-of-concept, the malware found in the wild has become much more sophisticated and adopted many of the functions of regular computer malware. McAfee states in its Q4 Threat Report, that they have seen an increase of new mobile malware in 2010 by 46 percent compared with 2009 [199]. Kaspersky Labs reports a doubling in the number of detected malicious programs targeting mobile devices [223]. Also, it is noted that 2010 was the year when the first Android and Apple's iPhone OS malware appeared [223, 360]. However, the J2ME platform is still the most targeted platform (57.67%) followed by Symbian (29.26%), Python[2] (5.64%), and Windows Mobile (5.08%) [223].

3.3 Mobile phones and other PDA-like devices

As described in the introduction, the mobile phone contains a large set of sensors and it is often carried with us. The information on the phone is

[2]Some platforms have a Python distribution, such as the Symbian Series 60.

also somewhat more structured than on a regular computer and can often be accessed through a well-defined API, such as the address book, the call history, etc. The phone contains sensitive information. It can be used to track our movements and record ambient voice, or make a video recording when two specific people meet. In the last couple of years, *tablets* have also been introduced on the consumer market. In many ways, the tablet shares similar properties with the smart phone and sometimes even the underlying operating system is the same. The discussion below can often apply to either tablets, mobile phones or ultraportable laptops but to simplify the presentation we use the term mobile phone.

There are several taxonomies describing attacks against mobile phones, considering the motivation of the attacker [115], the infection vector [71], or the resulting network attack [149]. Below, we have instead used the topics of recent categorized research papers as an outline.

3.3.1 Theft and the need for authentication

One of the largest risks with a portable small device is theft [132]. For that reason, early protective mechanisms focused on the authentication of the user, often through a simple password, and the ability to remotely delete all information on the phone. However, a recent attack demonstrated by Aviv et al. [29], shows that is is possible to guess the password pattern on a touch phone by studying the oily residue left by the owner's fingers. Furthermore, Backes et al. demonstrate how reflections of the screen in the eye can be used to read information from a distance with relatively cheap equipment [33, 32]. This attack may be quite easy to perform as the phone is often used in foreign environments away from the home or the office. Recent research by Zahid et al. [359] considers keystroke analysis as a more dynamic method to identify the user of the phone.

3.3.2 Privacy leaks

Given the personal information that the phone collects and stores, many research efforts and attacks target general privacy leaks. In a talk at Black hat 2010, Seriot discusses privacy leaks related to the iPhone [304]. There has been several applications in the *Apple App Store* that has been removed due to privacy concerns, such as the game *Aurora Feint* that tried to upload all the user's contacts, allegedly to find other players. A lawsuit was also filed against the developer Storm8 because of concerns of the collection of the users' phone numbers. A more scientific study of privacy that involved more than $1,400$ phone applications were performed by Egele et al. [111]. They found that most applications do not leak personal information, even if they are hosted on unofficial forums. However, more than half seemed to leak

the unique ID of the device, allowing the creation of user profiles. Balasubramaniyan et al. [38] have deviced a technique to identify the source and the path taken by the currently received call. With their method they can distinguish between Skype, Vonage, specific PSTNs, and cellular networks. Even though this is not an attack on the mobile phone itself, it identifies information of how and where the call is placed.

3.3.3 Problems with geolocation services

Looking more specifically at the location of the user and the ability to track his movement, there are several attacks and studies. One of the most recent events is the inadvertent leak by Apple [24]. Apple used an unencrypted cache to be able to faster locate a user than by only using GPS, but did not purge all entries. After a public outcry, the company responded and will reduce the time it saves the data.

Husted et al. [167] have investigated how a user can be tracked in a metropolitan area, if the phone is actively looking for a wifi network to use by broadcasting a unique identifier. They investigated how many tracking devices the adversary must control to be able to track a user efficiently. Similar techniques, albeit without binding the unique identifier to the actual user, is already used. Copenhagen airport has announced that they will use a similar technique to track (anonymous) passengers and use this data to build a better and more structured airport [248]. They claim no passenger information is tracked, only the phones.

Nevertheless, geolocation offers new opportunities and services. Many users do not mind sharing their location with a trusted third party and some friends if it leads to new services [143]. However, this information may be leaked e.g. if the third trusted party has rogue employees surreptiously using this information, or if the trusted party itself is attacked [271]. Manweller et al. [219] have developed a scheme were trust is only given to users if they have been at the same place before. In that way, there is no need to trust a central repository and new types of services are possible because you can share information with a wider range of people and not just with your closest friends. This subject is also investigated by Narayanan et al. [246]. Ristenpart et al. [287] describe a privacy-preserving device-tracking system, so that a device can only be tracked when it has been stolen (and the owner would like to locate it).

3.3.4 Exploits, worms and botnets

As described above, there have been several historical worms targeting the mobile phone. Some of these have required the permission of the user to be installed, depending on social engineering techniques or on flaws in the user interfaces. However, there are also exploits that do not require any (or

little user interaction). One very serious recent vulnerability is discussed by Miller and Mulliner [240]. By sending a specially crafted SMS to an iPhone or a phone running Android, it was possible to make it lose all network connectivity, resulting in a type of denial of service attack [241]. Further attempts allowed execution of arbitrary commands on the iPhone *without any user interaction* [88]. The service handling SMS messages, CommCenter, was running as root – in contrast to the web browser that was sandboxed. Speculating, many previous attacks have been web-related, meaning that developers are more aware of security risks over that medium and do not apply the same security mechanisms to other methods of communication. The research has since then been extended by Golde and Mulliner with attacks against a wide range of phones [139]. Previously, Mulliner and Vigna [239] looked at vulnerabilities in MMS User Agents.

Other attacks have targeted other programs on the phone. An earlier exploit discovered by Ormandy used a crafted TIFF image to execute arbitrary code [304]. A crafted MP3 file could trigger a crash on Android phones and possibly execution of arbitrary code [84]. Habib et al. looked at the network stack in smart phones and found that they could be vulnerable to "old" and well-known attacks [151]. Users of jailbroken phones have also been targeted by worms [65, 269]. The attack was very simple – the users had not changed the default password for SSH so the worm could simply connect using the default password `alpine`.

Singh et al. [308] have studied the propagation of worms that would almost exclusivley use Bluetooth. In metropolitan areas, such worms are a realistic threat and may have a large impact as the worm would jump through a "local" connection between two phones, and thus not go through the core cellular network where it could be detected and filtered. Yan and Eidenbenz [354] have also studied such propagation. Akritidis et al. [22] have looked at a similar problem; how a worm can spread using wifi networks in densely populated metropolitan areas. Even though their results is not particular to mobile phones, they show that under certain circumstances it is possible to reach 80% of all hosts within 20 minutes. They also discuss how it is possible to monitor the location of users. Fleizach et al. [130] study the propagation of malware by taking more parameters into account, such as constraints based on the entries in the address book of the users.

The latest phones have the ability to be remotely controlled by the user [91]. By connecting to the phone, one can send SMS, upload videos, view call logs, etc. Even though such functions make it easier for the user to administer his phone, it is another type of interface that can be attacked in the future.

3.3.5 Using the phone for DoS attacks

When it is possible to take command of a large number of hosts, these can easily be used for a traditional DoS attack. However, as Traynor et al. [330, 332, 331] explain, it is quite easy to attack the services in the cellular network from malicious devices on the network. They show that in special circumstances the DoS attack does not depend so much on available bandwidth as with problems with the connection of the cellular network to Internet. These two types of networks have fundamentally opposed design principles and sometimes these differences can create vulnerabilities. They demonstrate the attack principle by requesting the *home location register* but they also propose countermeasures to their attack. Mulliner and Seifert expand on the research, by demonstrating a practical cellular botnet for the iPhone [238]. Racic et al. [277] show how only a few rogue devices can dominate the time slots in a 3G network. They also give an overivew of other attacks that have been discovered elsewhere.

3.3.6 General protection strategies

Protecting the mobile phone from malware is a very active research topic and here we list recent research that offers general identification or protection strategies. One of the major difficulties is based on the limited amount of power available on the cell phone. Any new program that runs will use more power, meaning the phone needs to be charged more often. This is actually turned into a defense strategy as explained by Liu et al. [207]; also malicious programs will consume power. By modeling power consumption, their tool *VirusMeter* can detect malicious use. It runs in two modes. Lightweight analysis is performed when the phone is using the battery and a more heavyweight analysis is done when the phone is charging. Also others use the power consumption to detect malware, as in [190]. As Nash et al. [247] explain, the attack purpose of certain malware is actually to drain the battery and thus perform a sort of DoS attack against the phone.

There is also a tradeoff where to run the algorithms. The applications can be statically analyzed before they are offered for download in the marketplace, services in the cloud can be used to offload some of the work from the phone [256, 270], or the complete protective mechanism can be running only on the phone. Jeter et al. [179] focus on Android and discuss its security model with its virtual sandbox but claim that there are still weak spots. They used a “self-designed” malware to check the validity of their proposed solution. Schmidth et al. [300] use static analysis coupled with a collaborative effort to detect malware on Android phones. Becher et al. [42] use dynamic analysis to find malware. Jakobsson and Johansson suggest a practical method based on *memory-printing* and claim they can detect any active malware [178]. Others, notably Yan et al. [355], profile a user’s SMS

behavior in their *SMS-Watchdog* to discover spam, phising attempts, or denial of service attempts against the network. Also Bose and Shin consider protection mechanisms for SMS and instant messaging services, especially when they are connected [50].

Enck et al. [115] use their tool *Kirin* for lightweight certification of applications at install time. Their tool is again developed for Android. Barrera et al. [40] present a methodology to study the security model in Android, based on its permissions. Finally, if an installed application is deemed to be malicious, most vendors seem to have a kill switch to instantly remove it from the phones of the users [63].

As can be seen above, many of the research efforts to improve the protection of the mobile phone is geared towards Android, which might be because it is open source and for that reason easy to study and improve upon.

3.4 SoHo devices and consumer electronic devices

Other devices, usually overlooked by security experts, are so-called *SoHo* devices, i.e. devices that can be found in a Small Office or a Home Office. Consumer electronics, such as set-top boxes or music players also run the risk of being compromised even though most people would not consider them to be targets for attacks.

Just for completeness, we start to mention one of the most prominent devices that are used for propagation of attacks – the ubiquitous USB drive. Even though its problems are well known, it may cause problems even with users well aware of security [146] and it has been used to attack high-security facilities that have been airgapped [124].

Connor [257] describes how a modern printer works and how it can be attacked. These machines are similar in nature to a computer and they run a range of services. For example, Connor describes the web interface that is protected by a password, which is unfortunately seldom changed from the default one. However, even when it is changed there are still a range of vulnerabilities that can be used for an attack. Even though attacks using a printer may seem like a remote possibility, they have already been exploited [74]. In 1999, a printer located at the Space and Naval Warfare System command was hacked and the routing tables were changed. Files in the printing queue were thus directed to Russia and then back again. This way the attacker could keep a copy or even change the contents of the files being printed. According to Connor [257], other types of equipments, such as cash registers, ATMs, access control doors, and cctv systems, may also be vulnerable to similar attacks.

Vuagnoux and Pasini [339] show a weakness in another ubiquitous device in the office, the keyboard. Both wireless and wired keyboards generate compromising radiation that can be used to recover the keystrokes. They ar-

gue the flaw is there because of cost pressures in this market. The pairing of wireless keyboards, or any wireless device, is also fraught with peril as explained by Halevi and Saxena [152].

Celeda et al. [67] have done an analysis of the *Chuck Norris botnet*, which targets vulnerable ADSL modems and routers by using a brute force attack. They comment on the difficulty of the owner of the router to discover the attack attempts and the resulting compromise, as these devices run with older firmware, are not actively monitored and never patched.

Similarly, Saponas et al. [297] describe problems in a wide range of consumer electronic devices. They looked at Slingbox, Nike+iPod, and the music player Zune. With Slingbox, they could determine remotely what movie a user was watching. In Nike+iPod, a unique global identifier was found that could in turn be used to track the owner.

Looking at other consumer devices, also the ebook readers can be infected [77], as well as a digital frame for photos [134]. The malware can then propagate when the photo frame is connected to a regular computer. Bojinov et al. [48] have further studied the embedded management interfaces in a range of consumer products. All the devices they have examined have been vulnerable to a range of the web attacks further described in Chapter 9. We expect that many of the attacks described in the other chapters, even if they are older and already patched for regular systems, may work against consumer devices. The web interface, for example, is easy to implement and use but inherently complex so it is often attacked.

Finally, devices may not be directly targeted but still influenced by attacks against the regular (supporting) network. A recent example of such an activity is the attack against Sony's PlayStation Network [46], that may both have released privacy-sensitive information of its users as well as disconnecting the devices. It is even speculated that this attack was an attempt to control the network that issues updates to the actual consoles, thus being able to compromise and control millions of such devices [141].

3.5 Sensors and RFID

Of the topics covered in this chapter, security for sensor networks and RFID tags is one of the most developed. For that reason, another deliverable, *Review of the State-of-the-Art in Low Capacity Devices*, will describe the state of the art of key management, encryption, authentication, integrity, routing, aggregation, clock synchronization, self-stabilization, etc. In this section, we give a very brief overview on recent results regarding RFID tags, with only a few references to the most recent results.

RFID tags contain a tiny, miniaturized chip that is powered by means of inductions. Due to their size as well as their low cost, they can be attached to almost anything. They can be found on key cards, fare passes for public

transport, passports, driver licenses, pets, clothing, blood bags, and they are often used in supply chain management.

However, there is a host of security threats related to the RFID tags. For example, RFID may carry malware as is somewhat ironically outlined in a research paper named "Is Your Cat Infected with a Computer Virus?" [285].

One of the areas that has even reached general press is attacks against the Mifare classic, a contactless smartcard slightly more powerful than a classical RFID chip. It is being used in fare passes for public transport in London, Boston, the Netherlands, Australia, Taiwan, etc. One of the more powerful attacks allow the adversary to extract all cryptographic keys over wireless communication with access only to the card and no reader [135]. Thus, it expands on a previous attack described by de Koning Gans et al. [93].

The cloning of other types of cards, such as the enhanced driver licenses used in Washington, is investigated by Koscher et al. [195]. However, Danev et al. [87] show that many RFID smart cards actually have slightly different signatures if one look at the signal on the physical layer. They suggest that this fingerprint, together with the ID, may be a defense against certain cloning attacks.

Given that RFID tags emit a unique identifier, they can also be used as means to track people. Blass et al. [47] and Berbain et al. [44] suggest two different methods to protect the location data.

Finally, smartcards are also used in payment systems with the English "Chip & Pin" being one such example. Drimer and Murdoch discuss the problem if the reader is not trusted but relays the information you enter. They suggest a practical solution based on a distance bounding protocol [104].

3.6 Other types of devices

Going from "regular" computing systems to other areas of society, there are still risks for attacks. In modern society, also traditional devices are being extended with networking capabilities to increase their functions or to cut cost. *Smart meters*, measuring the electrical use of consumers, have been shown to be vulnerable to attacks [142] and Davis has simulated a worm outbreak by changing the firmware on such devices [92]. McLaughlin et al. [225] discuss how diversity techniques can be used to avoid large-scale attacks targeting a single vulnerability.

Vehicules, such as cars and airplanes, are also heavily networked. In fact, a modern car may contain between 50 to 90 computers [254], and the car network thus consists of quite a number of devices. The security of the network and the devices has been investigated experimentally by Koscher et al. [194]. They found that they can override the driver's input, even remotely [221]. Others have studied the passive keyless entry system [131]

and security and privacy vulnerabilities related to the tire pressure monitoring system [292]. Cars, as mobile phones, can be used to track the owner's location. For toll roads, traffic congestion monitoring, and other such services, there is a valid reason for tracking the movements of the vehicle. Popa et al. [268] explore a system to limit the information sent to the central server. Similarly, Balasch et al. [37] suggest a privacy-preserving implementation of toll pricing based on a cryptographic protocol. The security of the connected car will be expanded on in a later deliverable.

Medical devices has also been shown to be vulnerable to attacks. For example, Halpering et al. [154] describe attacks against pacemakers and other *implantable medical devices*. Such devices often contain valuable information for emergency personnel, and being able to read the information easily might save lives. On the other hand, such information should not be available to others to browse. Rasmussen et al. [280] discuss how access can be restricted and only given if the requester and the owner (wearer of the implant) is in close proximity.

Similarly to the PlayStation incident described above in Section 3.4, these devices do not exist in a vacuum. They often need communication with other equipment to function correctly. For example, it was speculated that one reason for a plane crash in Spain 2008 was due to a central computer system being infected by malware [226] and thus unable to detect technical problems with the aircraft, but in the final report this was no longer listed as a reason.

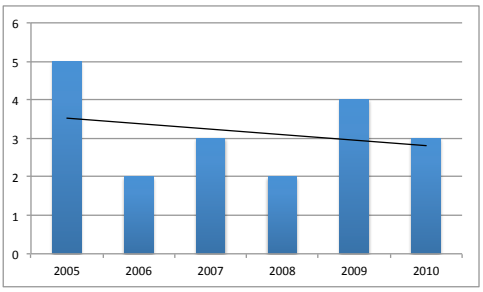
3.7 Conclusion

In a modern society, we use a range of devices to simplify life, both on a consumer level and of a more critical nature. To provide more and better services, these devices are often networked together and they may be attacked and controled remotely by a malicious adversary. We have listed several types of attacks, some seen in the wild but others only demonstrated in the laboratory, to show the range of existing problems. We also discussed some of the recent research efforts mitigating these threats. The malware targeting devices has so far followed similar development compared with malware targeting regular computers, and it will most likely continue to do so. Large-scale outbreaks have been predicted several times, especially for mobile phones, but has not yet reached a critical mass [168]. However, there are numerous incidents, among the latest the attack against the Sony PlayStation network, showing how millions of users may be affected even by attacks not directed at the devices themselves. The Chuck Norris Botnet, even though smaller in size, show that the step from attacking the infrastructure to the devices themselves is rather small.

4 Denial of Service

4.1 Introduction

According to the news on August 6, 2009, Twitter was shut down for hours, silencing millions of Tweeters [162]. From the view of a user, the first indication of this outage was "site is down"; actually, not only was the site down, but all client applications that depended on the Twitter API could also not connect to the service, creating a complete Twitter blackout. According to the Twitter's admission, this outage was caused by a *Denial of Service Attack*.



A Denial of Service (DoS) attack is an attempt by the attacker to prevent legitimate users of a service from using that service. Generally speaking, any attack that can saturate or exhaust system resources or get the system into an erroneous state, sometimes even crashing the system, should be identified as a DoS attack. The concept behind DoS attacks is not new, the general attack idea has been with us for more than twenty years and still keeps evolving. The first infamous DoS attack is the Morris Worm [319], an Internet worm developed by a Cornell graduate student. This worm can exploit and infect vulnerable systems automatically and then replicate itself. The worm could infect the same host multiple times, thus overloading it and creating a denial of service of the machine in question, even though it is claimed the Morris worm was developed without any malicious intent. From that point in time, more and more DoS incidents have been observed and reported. The period between 2000–2004 had the highest known frequency of DoS attacks seen so far. For example, in 2001 researchers from CAIDA

observed 12,000 attacks against more than 5,000 distinct targets from a three week-long dataset [235] using *backscatter analysis*.

The motives for launching a DoS attack vary. Some people may just want to show off their skills or prove that they have found some system vulnerabilities, as in the case of the Morris Worm. Economic incentive is another strong reason for launching a DoS attack. A company can benefit from launching attacks against its competitors, or more likely, the company in question will hire third party attackers to launch the actual attack [159]. On the other hand, when the attacker has the ability to launch a potential DoS attack, they can instead blackmail the victim into paying for the attack to not take place [164]. Apart from one's own gain, there have also been several attacks with a political motive. The incidents in Estonia [203] and in Georgia [222] are such examples of politically motivated attacks.

Regardless of the reason behind the attack, the resulting loss can be very significant. When the targets are websites or services of big companies or governments, the loss can even be billions of US dollars. For more information about the history of DoS attacks and other infamous incidents, one may refer to [209].

As DoS attacks have been so frequent and because the resulting damage can be large from an economic standpoint, the mitigation of these attacks is an active research area for both industry and academia. However, as explained below, preventing or mitigating DoS attacks is not an easy. Some types of attacks exploit a vulnerability in a software implementation, which can be patched. Others may use a design flaw, or even just a brute force technique to overload the recipient.

So how do then these attacks work? The attacker can study the flaws of communication protocols (or their implementations) and insert malformed or bogus packets to subvert the legitimate communications. This kind of attacks can be called *semantic attacks*. However, the attacker does not need to inspect the implementation of protocols, as it is possible to just flood seemingly legitimate traffic to congest the victim's network or keep the victim's host busy processing the packets – either way, legitimate clients will not be served. This kind of attacks is often referred to as *brute force attacks*. To successfully flood packets to overwhelm the victim's network, the attacker often uses many compromised machines or *zombies* to send traffic simultaneously, a form of attack called *Distributed Denial of Service (DDoS)* attack.

In the literature, there is furthermore a category of DoS attacks called *permanent* or *non-recoverable* DoS attacks. Non-recoverable attacks inflict permanent damage to the hardware of the victim [229], the replacement or reinstallation of the hardware is then needed to restore function. Since this kind of attack is not common, it is not discussed further in this survey.

4.2 Semantic attacks

Teardrop

The attacker sends incorrect IP fragments to the target. The target machine may crash if it does not implement TCP/IP fragmentation re-assembly code properly [68]. This kind of attack can be prevented by fixing the IP implementation bugs in operating systems.

Ping of Death

A ping of death is an attack where the attacker sends the victim a ping packet which is larger than $65,535$ bytes. Previously, many operating systems could not handle such large ping packets, and this attack led to a system crash.

SYN Attack

In this attack, the attacker takes advantage of an asymmetry in the TCP protocol. The receiver is required to keep state when a connection is about to be established, which a malicious sender does not need to do. When receiving a TCP/SYN packet, the receiver stores the state in memory waiting for the completion of the handshake (or a timeout). If the attacker continues sending TCP/SYN packets without sending back the final ACK packet for the TCP handshake, the server's resource can be quickly depleted by maintaining many half-open sessions. Even though the asymmetry still exists, it was a much bigger problem in the past as the table allocated for these half-open connections in many operating systems was of a limited size.

ICMP Flood (Smurf Attack)

In this attack, the attacker floods ICMP echo packets to the network, which broadcast these messages to all hosts present in this particular network. These ICMP echo packets have a spoofed IP source address, i.e. the the victim's address. This means that all the hosts who receive the echo packet will send echo reply packets to the victim, exhausting the bandwidth of the victim. This kind of attack is a mixture of a semantic attack with a brute force technique. The way the attack works is based on the response mechanism in ICMP. However, from the perspective of the victim, it is a brute force attack as the victim is flooded by packets from many machines.

Similar to the ICMP flooding attack, the attacker can take advantages of any reply-based protocol to launch a *reflected attack*, by spoofing requests from the victim to a large set of Internet servers, which will send all the replies to the victim. Common protocols used in the attacks include DNS queries, ICMP and TCP. For more information, one may refer to an analysis about reflected attacks [264].

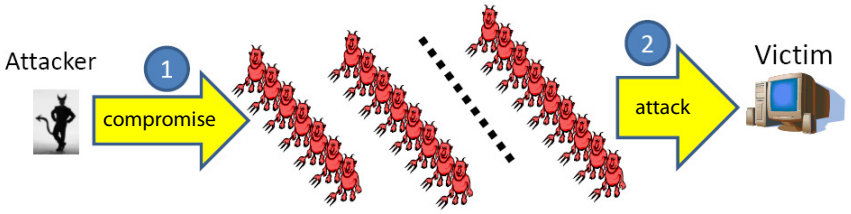


Figure 4.1: Phases in DDoS attacks

BGP Poisoning

The BGP protocol is used to establish routing paths between networks in Autonomous System level. The routing information is updated by exchanging the BGP advertisement between routers. Usually, the routers update their routing tables without verification of the BGP advertisements. The attacker can subvert the network communication by announcing a better route to some destinations, meaning that then all the packets to these destinations are routed to the attacker instead. Furthermore, the attacker can disturb the BGP routing by announcing fake BGP advertisements with addresses of other routers, meaning that the corresponding traffic will be routed to those routers which do not have optimal routes to the actual destination. These weak points in BGP are the basic motivation of designing secure BGP (S–BGP) and other revised versions of BGP [189].

4.3 Brute Force attacks

There are several variants of the brute force attack, and we use bandwidth DDoS attacks as an example where many machines flood packets to the victim simultaneously, thus achieving an amplifications effect by the use of many hosts.

The target of DDoS attacks can be hosts and Internet infrastructures (e.g. DNS servers, core routers). A typical procedure of a DDoS attack is shown in Figure 4.1. Basically, there are two phases in a DDoS attack. First the attacker exploits some vulnerability to recruit machines, which then can be used as attacking agents. The results is often referred to as a botnet where the compromised hosts are called zombies. The procedure of finding vulnerable machines and turning them into attacking agents can be done in several ways. The attacker may use Trojans or a worm that exploit a new vulnerability, such as the Code Red worm [161]. After the recruiting phase, the attacker can launch an attack by sending an attack command to the attacking agents through handler machines (masters) or via IRC communication channels [309].

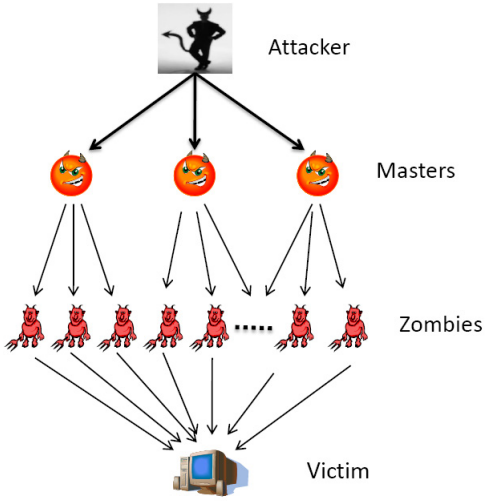


Figure 4.2: A typical DDoS network

To cover his/her real identity, the attacker usually recruits zombie machines with the help from handler (master) machines. The attacker first compromises and infects one or more masters, each of the masters can then further compromise many zombies. A typical DDoS network is shown in Figure 4.2.

Over time, many DDoS tools have been developed to maintain the DDoS network and launch DDoS attacks. Some of the more infamous include Trinoo [102], TFN [101], TFN2K [75], stacheldraht [100], Shaft [320] and mstream [321]. All of these tools are based on the architecture shown in Figure 4.2. The differences among them are basically the communication patterns between the attacker and the masters, and also between the masters and the zombies. The types of packets are also different from different tools. For a brief explanation about these attacking tools, one may refer to the computer security handbook, Chapter 11, by SeymourBosworth, Michel E. Kabay [51].

A recent interesting trend is that the attacker can actually assemble the botnet through the voluntary participation of the "victims," if there is a common cause. One such example is the orchestrated attack on organizations such as Mastercard.com, PayPal, Visa.com and PostFinance in 2010. This attack was organized by a group called "anonymous" to show their anger against the injustice against Wikileaks [20]. The attack is launched using an attack tool called LOIC [160], which can direct the volunteers to attack the indicated websites. The attack tool is easy to download and use. The attack was even further developed and in the end just required that the volunteers visited a web site to join in the attack. Thus, no skill is needed except knowing the right web site and clicking on a button [163].

As can be seen, a DDoS attack is more powerful than a DoS attack, since it can easily aggregate big volume of malicious traffic and quickly deplete the victim's network resources. Many of the more traditional DoS attacks can easily be turned into more powerful DDoS attacks. A simple type of DDoS attack is *ping flood*. A ping flood attack can be launched by a single attacking machine. However, it is more powerful if the attack is used in a DDoS attack, with many machine flooding ICMP echo (ping) packets to the victim simultaneously. Since the attacker can get very big aggregated bandwidth from many compromised machine, it is very easy to overwhelm the victim's network due the asymmetry of the bandwidth resources, even if the victim is a large service provider.

Basically, the attacker can flood any type of packets in a DDoS attack. For example, the attacker can combine TCP SYN/ACK, ICMP echo, ICMP echo reply, UDP packets together to flood the victim. Filtering out some specific types of packets cannot solve the problem completely. Furthermore, there are several other strategies that the attacker usually adopts to make the attacks difficult to defend against.

IP Spoofing

The attacker can use spoofed IP addresses in the malicious packets to cover the real identity of the attacking hosts. IP spoofing may affect the accuracy of the countermeasures of DDoS attacks. Additionally, since most attack prevention mechanisms use IP addresses to identify the source of attacks, innocent hosts may find themselves blocked because their addresses were spoofed by some attacker.

There are many solutions against IP spoofing, however any solution that relies on message authentication is potentially vulnerable to DoS attacks [114]. Ingress-filtering [263] or router based filtering [204] also cannot solve the problem completely. The attacker can spoof an address in an attacking agent's subnet, or spoof an address on the packet forwarding path to the destination.

Attack Rate

In DDoS attacks, usually the attacking traffic consists of many malicious flows originating from hosts which are scattered throughout the network. The attacker can adjust the attack rate, which makes it more difficult to identify malicious traffic. For example, the attacker can increase the attack rate very slowly to delay the attack detection. The attacker can also conduct *on-off attacks* or *pulsing attacks*, flooding packets intermittently and thus making the attack sources more difficult to be traced. If the attacker can recruit many zombies, he/she can easily launch a big volume of malicious traffic by having each of the zombie machines contribute only a small

volume of legitimate packet flows. In this case, it is hard to distinguish the attack from a *flash crowd*.

4.4 Other attacks

Peer-to-Peer

Peer-to-peer networks may be used in a DDoS attacks, where two common ones are index poisoning and routing table poisoning [350]. In the index poisoning attack, the aim of the attacker is to make several peers believe that some popular file is present at the the victim's host. To achieve this, the attacker sends a false index record with the victim's IP address and port number to all the other nodes. When a node wants to download this particular file, it will make a connection to the victim. During a short moment, there may be many connections to the victim, consuming the victim's resources dramatically. In the routing table poisoning attack, the aim of the attacker is to make the peers add the victim as their neighbor. The attacker sends node joining announcement message with the victim's IP address to every peer. Then some peers may adjust their routing table according to this information. So the victim may be selected as the forwarding node of many messages. If the attacker poisons the routing table of a large number of peers, the victim may receive a flood of search queries and maintenance messages, saturating its bandwidth.

Application Denial of Service

Some web application servers may suffer DoS attacks due to lack of control over their users' behaviors. For example the application server may not limit the resources that a specific user can use in the system. The attacker can then keep requiring resources (e.g. space in hard disk, memory, CPU time) until the resources are exhausted. The so-called *fork bomb* is a specific example.

The attacker can also launch attacks based on the fact that some systems cannot handle some errors properly [259]. For example some systems may lock out an account if it observes several failed attempts to login. Thus, the legitimate users cannot login when their accounts are locked. This latter is a good example of how a defense mechanism (a limited number of tries to login to avoid password guessing) can in turn become a new attack vector.

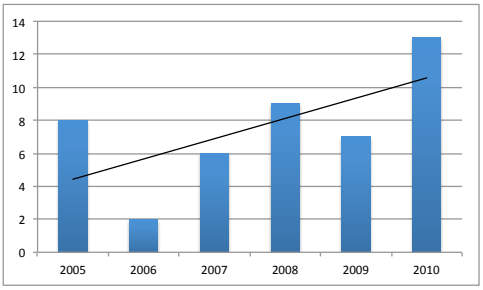
There are also many attacks due to the improper design or bugs in the codes of the server, such as deadlock in the allocation of resources, buffer overflow when copying files of invalid length. Since these attacks are not based on network protocols, we do not include them into the semantic attack category.

4.5 Conclusion

Since it first occurrence, the DoS attack has changed form and sophistication. As mentioned above, there are many kinds of DoS attacks. Improper implementation of network protocols and poor design or implementation of application servers may be exploited by the attacker, leading to DoS attacks. These vulnerabilities can many times be fixed through software patching. However, the risk of DoS attacks is not eliminated even if the web site is well implemented and updated regularly. It can still be successfully attacked by a brute force attack. It is said that regardless of how well secured the victim system may be, its susceptibility to DDoS attacks depends on the state of security in the rest of the global Internet [66]. Even worse now, the bots army can be formed by volunteers. Their participation in the attack is not because their machine is compromised, but because they want to and they find the cause worth the attack. For that reason, the DoS problem is not only a technical security problem any longer, but also a type of attack where issues and large conflicts in society will play a role.

5 Critical Infrastructure Attacks

5.1 Introduction



According to the definition in "Directive on the identification and designation of European Critical Infrastructure and the assessment of the need to improve their protection" [8] the term "Critical Infrastructure" means: "... those assets, systems or parts there of located in the EU Member States which are essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions...".

The above quoted directive is a consequence of the "European Programme for Critical Infrastructure Protection" [4] which refers to the doctrine and programmes created to identify and protect critical infrastructure that, in case of fault, incident or attack, could seriously impact both the country where it is hosted and at least one other European Member State (European Programme for Critical Infrastructure Protection).

In the Deliverable D 1.2.1 "Scenario analysis" [6] of the IST Project IRRIS (Integrated Risk Reduction of Information-based Infrastructure Systems) [6] the following definitions, which we have accepted for further use in this survey, were done:

Infrastructure (I) An infrastructure forms a framework of (inter) dependent networks and systems comprising identifiable industries, institutions (including people and procedures), and/or distribution capabilities

that provide a reliable flow of products, supplies and/or services, for the smooth functioning of governments at all levels, economy and society as a whole and of other infrastructures.

Critical Infrastructure (CI) A critical infrastructure is a large scale infrastructure which, if degraded, disrupted or destroyed, would have a serious impact on health, safety, security or well-being of citizens or the effective functioning of governments and/or economy.

Critical Information Infrastructure (CII) Information processes supported by information and communication technology (ICT) that are critical infrastructures for themselves or that are critical for the operation of other critical infrastructures.

Critical Infrastructure Protection (CIP) The programs and activities of infrastructure owners, manufacturers, users, operators and regulatory authorities which aim at keeping the performance of critical infrastructures in case of natural disasters, failures, human error, attacks or accidents above a defined minimum level of services and aim at minimising the recovery time and damage.

Critical Information Infrastructure Protection (CIIP) The programs and activities of infrastructure owners, manufacturers, users, operators and regulatory authorities which aim at keeping the performance of critical information infrastructures in case of failures, attacks or accidents above a defined minimum level of services and aim at minimising the recovery time and damage. CIP is more than CIIP, which is only a subset of a comprehensive protection effort, as it focuses on the critical information infrastructure. But in official publications, both terms are often used inconsistently.

The cyberattacks are possible only to and through the ICT components of CI that means to the CII. The Critical Infrastructure and especially CII have become very vulnerable to terrorists, criminals and fun seeking persons like hackers, crackers, etc. The tools used to attack CII are as any other tools for cyberattacks in general.

According to [108] CII attacks include:

- Unauthorized access to sensitive or confidential information;
- Destruction, modification or substitution of software needed by critical infrastructures;
- Limited access for the agents able to prevent or mitigate the results of the attacks.

The possible consequences from critical infrastructure attacks include [108]:

- Blocked transportation, electricity and water supply, communications, data transmission, nuclear power plants, air-traffic control;

- Bankruptcy of commercial structures and financial systems, failure of international business transactions, destabilization of markets and financial institutions, money and information theft;
- Loss of intellectual property or reputation (due to a worm attack the company for on-line payments PayPal was facing a bankruptcy in 2002);
- Human victims or material losses, provoked by the destructive use of critical infrastructure elements (cyber sabotage in the food industry, air or railway traffic);
- Unauthorized access and/or modification of personal information;
- Possibility for imputing terrorist acts to other country/government and aggravation of the tension in international relations.

While the actual restoration of the CII could be a quick and easy task, the indirect effects of even the shortest failures can be felt for a while. CII attacks can seriously undermine public and business confidence in electronic commerce, government initiatives, etc. The human and economic costs associated with recovery or mitigation strategies are enormous.

5.2 Analysis Critical Information Infrastructure security issues

Generally, the analysis of the Critical Information Infrastructure (CII) is a tough mission. The most appropriate ways for the implementation of the analysis of the CII security issues are:

- Model based analysis;
- Scenario based analysis of the Critical Infrastructure Attacks.

5.2.1 Model Based Analysis of Critical Information Infrastructure

Model based analysis of CII relies on the assumption of ICT as an element of the Critical Infrastructure and its consideration as a Complex Adaptive System (CAS) that can be classified as socio-technical systems [214], [296], noting the facts that modern ICT and the flows interrelated to them could be revealed as cyberphysical systems [275].

CAS analysis in itself could be performed by utilizing the methodological framework proposed in [106]. This allows studying of both static and dynamic aspects of the infrastructure and at the same time provides the

necessary level of details. Briefly, the framework encompasses: system analysis, behavior analysis, knowledge discovery, visualization and information sharing.

Additionally, the process could also be supported by a number of analysis schemes, amongst of which as most researched could be reckoned the following ten: AIMS, Aspen, CASCADE, CISIA, GoRAF, HHM, IIM, IRAM, OGG CIPI and UML-CI.

AIMS (Agent-based Interdependency Modeling and Simulation) was proposed in [18]. It provides a multiagent based approach for modeling and simulation that allows both modeling and behavior monitoring during the simulation, including unexpected events injection in the context of a scenario of interest. Additionally, the system allows usage of predefined component templates.

Aspen [242], [58] is also an agent based micro-simulation model especially designed for the US economy by Sandia National Laboratories. The model uses evolutionary learning techniques like genetic algorithm within the agent architecture and is capable on simulation of simple decision making agents (households, banks, governments, and companies).

CASCADE is a probabilistic and dynamic complex system model for cascading failure analysis. It allows analysis and behavior monitoring of the load transfer in between interconnected systems under extreme failure conditions. The basis of CASCADE application is a scenario of identical components from similar or different infrastructures that are cascaded in a model system and a failure in some of its components is monitored to appear [169].

CISIA (Critical Infrastructure Simulator by Interdependent Agents) is designed for analyzing the short term effects of a failure both in terms of faults propagation and with respect to performance degradations. CISIA is based on Repast [15] software framework that provides a library of classes for creating, running, displaying and collecting data from an agent based simulations. The modeling process is describing the infrastructure behavior as each agent is representing a macro component of the modeled system [295].

GoRAF integrates engineering and business perspectives for risks identification related to the interdependences in between enterprises. Criticality of the elements is calculated on the basis of risk and business values. The dynamics of the model utilizes CISA simulator and shows the probability of the interoperability of infrastructure resources [261].

HHM (Hierarchical Holographic Modeling) is a modeling scheme for multiple perspectives representation based on scenarios that is utilized for complex systems such as infrastructure modeling. This approach is based on the assumption for integrated, hierarchical multilevel complementary decompositions of a system. By using HHM, sixteen different perspectives were identified: Physical, Scope, Temporal, Maintenance, Institutional, Organizational, Management, Resource Allocation, Supervisory Control and

Data Acquisition (SCADA), Systems Configuration, Hydrology, Geography, External Factors, Buffers, Contaminants, and Quality of Surface and Ground Water [351], [352].

IIM (Inoperability Input/output Model) is based on the Leontiefs input/output model. It illustrates the interdependency between different sectors of the economy by focusing the investigation on the degree of direct or indirect financial reliance between the sectors [352]. Based on a set of initial disruptions in a sector, the model is able to characterize the cascading effect of failure. A dynamic extension to IIM allows a temporal analysis of the recovery mode.

IRAM (Infrastructure Risk Analysis Model) is a four phase methodology for identifying, ranking, assessing and managing the extreme risks threatening an infrastructure system. The identification phase is supported by HHM for studying the system decomposition followed by ranking of the vulnerabilities and threats of the sub-components. During the assessment phase a set of suitable scenarios of the sub-component operations are generated and shown with the help of event trees. The created scenarios are also ranked. The assessment of infrastructure surety in the third phase is undertaken through five different risk measures. These measures support the decision maker in the damage assessment, understanding the behavior of low probability-high impact events, and assessment of the surety of the system. At the final managing phase - based on the analysis of the previous phases, a decision maker is able to perform a qualitative and/or quantitative judgments of the current situation and to take appropriate steps to reach an ideal (desired) state [30].

OGC CIPI (Open Geospatial Consortium Critical Infrastructure Protection Initiative), [258] is a program coordinated by OGC for pilot studying of the emergency management through data sharing at different governmental levels. The study provides a basis for sharing of geographical information and standards interoperability developments. It also concerns the integration in the emergency alert systems based on draft interface specifications. The main goal of CIPI is to improve the interoperability between deferent sources of information (telecommunication, water resources, oil and gas, government, transportation, emergency response, electric power and health services infrastructure) for collaborative detection, prevention, planning, responding and recovering from natural vulnerabilities and human threats.

UML-CI is a Critical Infrastructure UML driven reference model [85] that provides a capability for high level infrastructure metamodels creation and profiling based on six dimensions of critical infrastructure specification. This profiling allows initial insight for infrastructure analysis and system identification, providing a solid basis for common understanding, communication, and knowledge transfer, allowing at the same time documentation of best practices and infrastructure metamodels. The models created with UML-CI

metamodel can be re-used as a base or guideline for the creation of new infrastructure systems [107].

5.2.2 Scenario based analysis of Critical Infrastructure Attacks

Generally, the scenario is a basic concept of description of an event or series of actions and events for future situation that intends to realize what would be the real-world impact of appropriate cyberattack. The most common way for the scenarios building are brainstorming sessions and Delphi results conversions. A good example for a scenario list related to CI attacks could be found in [11] which resulted after FORWARD project activity.

Very popular in the modern society is the group of scenarios named **Cyber-doom scenarios** that are generally a reflection of long-held, but ultimately incorrect, assumptions and fears about the fragility of modern societies and infrastructure systems as a result of a cyberattack. The paper [294] examines the cyber-doom scenarios and place them into a larger historical context, assess how realistic they are and draw out the policy implications of relying upon such tales, as well as alternative principles for the formulation of cybersecurity policy. It draws from research in the history of technology, military history, and disaster sociology. The paper argues that infrastructural collapse leading to social and/or civilizational collapse is not supported by the current body of empirical research on the subject and cyber-doom scenarios encourages the adoption of counter-productive policies focused on control, militarization, and centralization. This paper also accentuates on the fact that cybersecurity policy should be based on more realistic understandings of what is possible on the base of empirical research rather than hypothetical scenarios and should be guided by principles of resilience, decentralization, and self-organization.

5.3 Recent Research on Critical Infrastructure Cyberattacks

We discuss now recent developments in research related to critical infrastructures.

E-voting, Crypto and DoS attacks are presented in **"Cryptographic Voting Protocols: A Systems Perspective"** [69]. The cryptographic protocols are one part of a voting system which consists of voting machines, software implementations, and election formal procedures. In this paper, the security features of two cryptographic protocols are analyzed, one proposed by A. Neff and another by D. Chaum. Several potential weaknesses are discovered in these voting protocols. These weaknesses include: subliminal channels in the encrypted ballots, problems resulting from human unreliability in cryptographic protocols, and denial of service. These attacks could

compromise election integrity, erode voter privacy, and enable vote coercion. Whether our attacks succeed or not will depend on implementation of a voting system as a whole.

Attacks against E-voting and malware insertion are presented in the next two short talks. **"Electronic Voting in the United States: An Update"** [19] This invited talk is a review of the issues in Diebold's Accuvote TS and TSX voting machines, used in public elections in USA. The author's team found serious security vulnerabilities in the machines, and lack of understanding of software and hardware.

"Computerized Voting Machines: A View from the Trenches" [31] This is a short talk concerning problems with introducing paperless computerized voting systems, DRE based (Direct Recording Electronic) in USA. There is a miscommunication between cybersecurity and software experts and policy makers. The main risks while taking fast political decision in this manner are:

- Every program has bugs;
- Last minute updates are dangerous;
- It is possible to insert malicious code in such a system;
- It is very difficult to detect and fix a malware threat, especially in a large program code.

Policies and Assessement are discussed in **"Homeland Security: Networking, Security, and Policy"** [2] This invited talk presented an overview of the recently created Department of Homeland Security, its Science and Technology Directorate, and some of the research initiatives started in the Department. Many of these initiatives provide examples where networking, security, and policy come together in interesting ways as the Department works with critical infrastructure providers to secure the nation's infrastructures.

Crypto, Trust and Languages categories are presented in the next two papers. **"Sanitizable Signatures"** [133] The described sanitizable signatures offer many security features for critical applications. A sanitizable signature allows authorized semi-trusted censors to modify - in a limited and controlled fashion - parts of a signed message without interacting with the original signer. A common model is presented for this new primitive, based on standard signature schemes and secure under common cryptographic assumptions. Some experimental measurements for the implementation of a sanitizable signature scheme are provided.

"Security-Typed Languages for Implementation of Cryptographic Protocols: A Case Study" [16] Every modern critical infrastructure relies on a trusty communication. Security protocols are an essential part of that

system. Cyberattacks are directed in two ways - against the protocol itself and against the protocol's particular implementation. According to CERTs (Computer Emergency Response Team) most of the exploited vulnerabilities come from bad protocol implementations. The paper presents security assurance provided by security-typed languages when implementing cryptographic protocols. As a demonstration, this case study uses Jif, a Java-based security-typed language, for implementing a non-trivial cryptographic protocol that allows playing online poker with untrusted third parties in a fair manner. The conclusion is that security-typed languages are very useful tools for analyzing and exploring weaknesses of the protocol implementations.

Remote E-voting issues again and a solution in **"Civitas, Toward a Secure Voting System"**, [213] This paper describes Civitas, a remote voting system. Special attention is paid on security part of implemetation. Civitas is compared to previous systems of this kind and its advanteages are explained. The system provides stronger security. The new technical achievments are: secure registration protocol and a scalable vote storage system.In fact, the article does not concern cyberattacks directly but it presents a good example of implementation of a reliable voting system.

RFID, Attacks and Security of Devices, Access control, Malware, Reverse engineering and debugging, and Cross-domain attacks are concerned in the next four papers. Different types of smart cards are used in some CI environments. **"Wirelessly Pickpocketing a Mifare Classic Card"** [119] discusses four atacks against the widely used smart card Mifare Classic. The attacks are based on the fact that the stream cipher CRYPTO1 has been recently reverese engineered. The attacker only needs a wirelles access to a card. but not to the reader. The strongest of the described attack may recover a secret key in less than one second not using a high performance hardware. This is an example of a very easy and effective attack. Another weakness while performing nested authentications provides enough plaintext for a speedy known-plaintext attack.

"Chip and PIN is Broken" [117] focuses on the EMV protocol (named after Europay, MasterCard, and Visa), which is used for smart card payments worldwide. Therefore it is considered to be a fundamental engine of a large critical infrastructure such as banking operations. EMV secures credit and debit card transactions by authenticating both the card and the customer presenting it through a combination of cryptographic authentication codes, digital signatures, and the entry of a PIN. This paper demonstrates a protocol flaw which allows criminals to use a genuine card to make a payment without knowing the card's PIN, and to remain undetected even when the merchant has an online connection to the banking network. This is a form of a man-in-the-middle attack to trick the terminal into believing the PIN verified correctly, while telling the card that no PIN was entered at all. A practical example of an attack against EMV is given and discussed in details.

The security vulnerabilities are clearly defined. The conclusion is that the EMV protocol is broken by design, which necessitates further research for the next version of EMV to be developed properly.

"Analysis of a Modern Automobile", [183] Modern automobiles are complex systems which are controlled by several computers interconnected via internal networks, which introduces potential risks. The paper experimentally evaluates the issues on a modern automobile and demonstrates the vulnerability of the underlying highly computerized system. It is possible that an attacker who is able to infiltrate virtually any Electronic Control Unit (ECU) can leverage this ability to completely overcome internal protections. Several practical experiments are conducted. It is found that it is possible to bypass primitive network security protections within the car, such as maliciously bridging between car's internal subnets. Composite attacks are also presented in cases that leverage individual weaknesses, including an attack that embeds malicious code in a car's telematics unit and that will completely erase any evidence of its presence after a crash.

"False Data Injection Attacks against State Estimation in Electric Power Grids" [353] The paper describes attacks directed to system monitoring of the electric power grids. System monitoring has an important role for the reliable operation of power grids. It produces measurement and status data of the grid. There are several methods for analyzing and detecting bad or incorrect measurement values including malicious ones. In fact, an attacker could inject false measurement data and this will not be detected by the monitoring system. For this purpose, the attacker needs physical access to a measurement device such as power grid sub-station. The attack can introduces arbitrary errors into certain state variables without being detected by existing monitoring algorithms. Two scenarios are presented: the attacker is either constrained to specific meters or limited in the resources required to compromise meters. It is demonstrated that the attacker can construct attack vectors to change the results of state estimation in both scenarios. Validation of the attacks is performed by simulation based on IEEE bus test systems, using MATPOWER simulation package for MATLAB.

Network IDS as a defence against the attacks is proposed in **"Detecting Network Anomalies in Backbone Networks"** [61] All of the critical infrastructures nowadays depend on a complicated telecommunication structure with high performance backbone networks as a basis. Detecting anomalous traffic is of primary interest in IP networks monitoring and management. The problem is even more challenging when talking about high speed and high availability backbone networks that may serve critical infrastructure. This paper is focused on the development of an anomaly based Net-Intrusion Detection System (IDS) based on Principal Component Analysis (PCA). PCA is a technique that allows the reduction of the number of variables, while retaining most of the original variability the data. It is possible to separate the traffic into normal and anomalous components, thus revealing an anomaly.

Therefore revealing the traffic anomalies is a good base for timely detecting and analyzing the cyberattacks.

5.4 Industrial Critical Information Infrastructure

The industrial CII security is an interesting and modern research field [5]. Their security is basically related to Supervisory Control And Data Acquisition [184] used in the industry. Within this context the North American Electric Regulatory Commission [249] has spearheaded an effort to define cyber security standards intended to provide a cybersecurity framework to identify risks, help secure critical cyber assets ensuring operational reliably.

A good high-level analysis of the possible threats to a power plant system, a categorization of the typical hardware devices involved, and some high level discussion about the intrinsic vulnerabilities of common power plant architectures could be found in [17]. Another work related to SCADA security, is presented in [276]. What however should be noted is that communication protocols used in such systems (e.g. Modbus, DNP3, etc.) are not prepared for ICT typical threats, because at the time of their design the industrial networks were closed for Internet. Some work within the direction of improving these protocols security has been published in [216], [180], [322]. Those papers underline how SCADA systems, and generally speaking industrial process control systems, are at the moment exposed to ICT threats. Regarding this context several security assessment methodologies for complex ICT based systems and infrastructures like: CORAS, EBIOS, INSAW and OCTAVE together with a set of six scenarios against CIP (the power plant attacks have been studied [170]:

Scenario 1: RADIUS (Remote Authentication Dial In User Service) Denial of Service;

Scenario 2: Domain Credential Stolen/External Connection DoS: this attack scenario has a twofold goal: (a) to obtain the credential of a user who tries to log on the RADIUS Server; (b) to cause a DoS by rejecting any remote log-in request;

Scenario 3: Intranet Virus Infection;

Scenario 4: Data Network Worm Infection;

Scenario 5: Process Network Malware Infection;

Scenario 6: Phishing Attacks and Local DNS Poisoning.

5.5 Some recent Examples for Cyberattacks on Critical Information Infrastructure

Nowadays even the basic services of a modern information society like: water, electricity and telecommunications are now computerized and often

connected to the Internet [182]. In the same publication a short survey of Critical ICT related to the earliest days of the WWW, are noted, mentioning: Chechen terrorists demonstrating the power of Internet-enabled propaganda [262]; the United States and China on-state, "patriotic" hacker war, with uncertain consequences for national security leadership; Syrian air defense disabling by a cyberattack moments. The Kyrgyzstan domestic political crisis and Iranian voters, in "open war" with state security forces, used peer-to-peer social-networking websites to avoid government restrictions on dialogue with the outside world [59]. In this context the surveys on cybersecurity presented in HORIZON 2015 should also be marked [10].

The asymmetric nature of information technology and cyber warfare manifests itself in many ways: e.g. Smurf and botnet based attacks including criminal results like the one of the "bumbling hacker" (Briton Gary McKinnon) considered by the Pentagon as "the biggest military computer hack of all time" [215].

In terms of financial damages a "mafiaboy" case for a 15-year-old kid from Montreal in 2001 have to be noted for him beeing able to deny Internet service to some of the world's biggest online companies, causing an estimated $1.7 billion in damage [86].

In 2010 a number of hackers attack in support of the WikiLeaks phenomena were produced against VISA, Master card, Facebook and even Twitter [345].

Additionally, the worm Stuxnet attack [243] that covered several countries (Iran, Indonesia,India, United States, Australia, United Kingdom, Malaysia and Pakistan) but mostly dangerous for a nuclear power plant in Iran is also another example of practical CIP vulnerabilities based on SCADA wholes.

Finally, we should not also omit and the number of attacks against giants like Google, Yahoo, Microsoft, Mozila and Skype most claimed to be related with Iran and China [144] [205].

5.6 Conclusions

As a result of the present literature survey it can be concluded that the attacks related to Critical Information Infrastructure (CII) are basically connected to the presences of weak points in its architecture, communication protocols (e.g. gaps in SCADA system protocols, financial system protocols and voting systems), DoS attacks (implementing Botnets, Smurf and other worms), intrusions from outsiders and insiders for good or evil social impact (e.g. phishing and frauds in social networks like Facebook, Twitter, Skype and social influence in websites like WiliLeaks). However, studying of cyberattacks on CII could benefit from the modelling with different concepts, software environments and scenarios that allow both static/dynamic behavior and nature exploration.

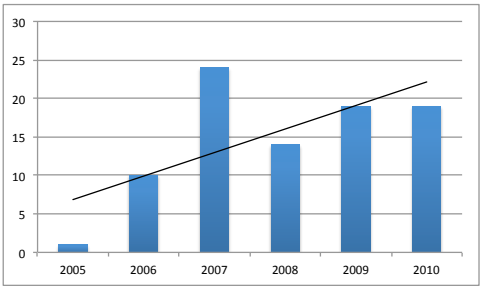
Apart of this, there are not so many scientific publications directly related to the Critical Infrastructure attacks, as a whole that are publicly available for the last 5 years. The main reason is the confidentiality as one of the key principles for the realization of Critical Infrastructure Protection described in "European Programme for Critical Infrastructure Protection" [4]. According to this principle the access to the information about CI security should be granted only on a need-to-know basis and the related information sharing should be classified and done in an environment of trust and security. Our opinion is that this principle should be referred to the CI details but not to all security issues.

Another reason for the relative lack of information about CI security issues is that most CI operators hold a monopoly in their market. This contributes to a tendency to reduce their communication with the outer world, especially when it comes to sensitive information.

6 Social Network and Privacy Attacks

6.1 Introduction

Hundreds of millions of users are registered to social networking sites and regularly use their features to stay in touch with friends, communicate, do online commerce, and share multimedia artifacts with other users.



Unfortunately, social networking sites are also a very attractive target for attackers because of the nature of the sensitive information that they contain about registered users. Typically, users enter their real e-mail addresses and provide information on their education, friends, professional background, activities they are involved in, their current relationship status and their sexual preferences. Hence, from the attacker's point of view, access to this type of detailed, personal information would be ideal for launching targeted, social engineering attacks, now often referred to as spear phishing [7, 3]. Furthermore, the collected e-mail addresses and personal information would be invaluable for spammers as they would 1) have access to e-mail addresses that belong to real people (i.e., one problem spammers face is that they often do not know if the e-mail addresses that they collect are indeed being used by real people or they are just secondary addresses that are not regularly read) and 2) have information about the people using these e-mail addresses allowing them to efficiently personalize their marketing activities, tailored according to the knowledge from the target's profile.

In addition, the ability to associate personal information with an e-mail address is important to be able to successfully bypass spam filters [186].

Such filters usually generate a list of spam tokens versus good tokens after training with a large set of previously received e-mails. As a result, e-mails that contain the name of the user receiving the e-mail, or names of people that he is acquainted with tend to receive lower spam ratings than e-mails that are less personal. As a result, if the spammer is able to include some personal information in the spam that he is sending, he would be able to improve his chances of reaching the targeted user.

Several publications have analyzed and measured features of social networks that are privacy-related. For example, Mislove et al. present a measurement study on social networks [231] while Bonneau and Preibusch evaluate the privacy settings and policies of a large number of social networks [49], finding strong evidence that the social networking market is failing to provide users with adequate privacy control.

In the rest of this chapter, we will analyze different kind of attacks against user's privacy on online social networks. We will also provide a survey of the corresponding countermeasures that have been proposed by researchers to mitigate the threats.

6.2 Attacks against the Social Network

The most well-known attack to compromise the trust relationship in a social network that employs a reputation system is the *sybil attack* [103]. In this attack, the attacker creates multiple fake identities and use them to gain a disproportionately large influence on the reputation system. The attack relies on the assumption that it is relatively easy to create fake profiles on most of the existing social networks.

To defend against sybil attacks, many approaches have been proposed. In particular, SybilGuard [357] and SybilLimit [358] are based on the fact that real-world social networks are fast mixing [129] and this insight is used to distinguish the sybil nodes from normal nodes. Fast mixing means that subsets of honest nodes have good connectivity to the rest of the social network, while it is very hard to achieve the same level of connectivity with fake nodes. If this assumption is true, both SybilGuard and SybilLimit are good solutions for detecting Sybil nodes. However, many attacks we present in this chapter result in the attacker gaining legitimate friendship connections and, therefore, would not be detected by current sybil-detection approaches.

6.3 Impersonation and Profile Cloning Attacks

The main prerequisite for being able to access personal user information in a social networking site is to have a confirmed personal relationship with the target. For example, the default setting in Facebook is to allow all confirmed friends to have access to the personal information (e-mail address,

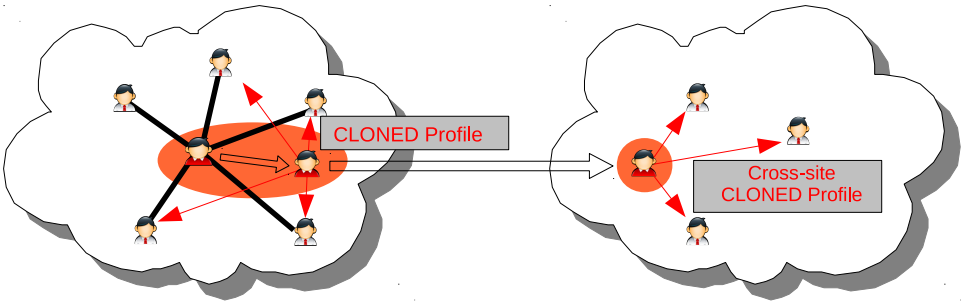


Figure 6.1: Single-site and Cross-site Profile Cloning

photographs, etc.), but not to provide it to unconfirmed third parties. Similarly, in LinkedIn, the contacts of a person can only be accessed if it is a confirmed business contact, and therefore he/she has already accepted a request and confirmed the relationship. Moreover, as we already explained in the previous section, fake nodes need to get connected to real users in order to look more realistic and avoid detection techniques based on the "fast mixing" property.

One way to achieve this result consists in impersonating a real user, instead of creating fake account for a non-existing person. Hamiel and Moyer conducted an impersonation experiment in which they created a fake profile on LinkedIn for the well-known security expert Marcus Ranum. The authors obtained the information to create a plausible profile by manually surfing the web, visiting Ranum's personal web page, and his entry in Wikipedia [236]. By impersonating a high-profile person, the authors showed how effective an impersonation attack can be. The forged profile received many friend requests, even from one of the target's immediate family members.

The best example of identity fraud and impersonating attack in social networks has been presented by Bilge et al. [45]. In the paper, the authors present and experimentally evaluate two identity theft attacks (see Figure 6.1 that would allow an attacker to establish a friendship connection with the victims and, hence, access their personal information. The first presented attack consists of the cloning of existing user accounts inside the same social network. In the second, more advanced attack, the paper show that it is feasible to launch an automated, cross-site profile cloning attack where the victim's contacts are retrieved from one network and re-established in a different social network where the target is not registered yet.

Unfortunately, impersonation attacks are not just a possibility. For example, in 2009 Nature published an article [202] on a number of scientists linked to stem-cell research, that have been impersonated on Facebook in a convincing, but bogus, network of apparent friends.

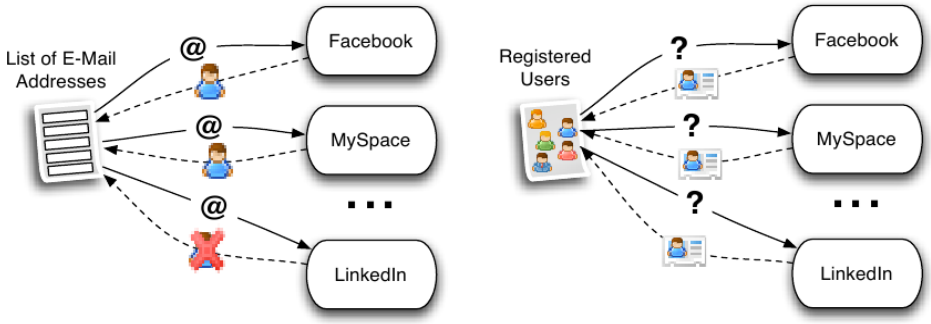


Figure 6.2: Automated user profiling based on information collected on social networks.

6.4 De-anonymization Attacks

De-anonymization of privacy-sensitive data is not a new concept. Researchers initially focused on anonymization and de-anonymization of network level data. However, due to the popularity of social networks and the large amounts of sensitive data they store, the focus of de-anonymization research has recently extended to this area. Several publications have shown that seemingly non-sensitive data from publicly available sources can be used to recover private information about individuals.

For example, Griffith and Jakobsson [148] use public records to infer individuals' mothers' maiden names, and Heatherly et al. [156], as well as Zheleva and Getoor [364], show how public data provided by social networks can be used to infer private information. Zheleva's paper propose a technique to predict the private attributes of users, reducing the problem to a relational classification problem based on friendship and group membership information.

Narayanan and Shmatikov have shown that statistical methods can be applied to de-anonymize micro-data by cross-correlating multiple datasets [245]. They extend their approach to social networks in [244], and prove that it is possible to de-anonymize members by mapping known, auxiliary information on the (social) network topology. Diaz et al. present a de-anonymization approach that uses information gained from observing communication patterns between social network members [99]. Finally, Backstrom et al. showed how to deanonymize a single social network [34].

Another example of cross-network de-anonymization is presented by Balduzzi et al. [39]. In this case, the authors toke advantage of a common weakness, namely the fact that an attacker can query popular social networks for registered e-mail addresses on a large scale. This allows an attacker to correlate public information of users that are registered on multiple social networking web sites with the same e-mail address (Figure 6.2).

The paper experiments demonstrate how it is possible to automatically extract information about users that may actually wish to hide certain online behavior. For example, the authors were able to identify users who are potentially using a different name on a dating web site, and are pretending to be younger than they really are. Obviously, this kind of cross-site correlation has a significant privacy impact.

Finally, a different approach to de-anonymize users has recently been proposed by Wondracek et al. [348]. The paper presents a novel technique that leverages well-known web browser history stealing attacks and group membership information available on social networking sites. The authors show that the combination of these two pieces of information is sufficient to uniquely identify users, or, at least, to significantly reduce the set of possible candidates. Thus, whenever a social network user visits a malicious website, this website can launch a de-anonymization attack and learn the identity of its visitors.

The main lesson is that a user cannot rely on anonymization to ensure individual privacy in social network and many successful attacks exists that are able to re-identify and de-anonymize also the more careful users.

6.5 Social Engineering Attacks

Social engineering attacks are well-known in practice as well as in literature (e.g., [232, 13, 343, 176, 317]). Social engineering targets human weaknesses instead of vulnerabilities in technical systems. Automated Social Engineering (ASE) is the process of automatically executing social engineering attacks. For example, spamming and phishing can be seen as a very simple form of social engineering (i.e., making users click on links). In the Scientific American Mind [116], Robert Epstein reports how he was fooled by a computer program that pretended to be a Russian woman.

A general problem on social networks is that it is difficult for users to judge if a friend request is trustworthy or not. Thus, users are often quick in accepting invitations from people they do not know. For example, an experiment conducted by Sophos in 2007 showed that 41% of Facebook users acknowledged a friend request from a random person [9]. More cautions users can be tricked by requests from adversaries that impersonate friends [45]. Unfortunately, once a connection is established, the attacker typically has full access to all information on the victim's profile. Moreover, users who receive messages from alleged friends are much more likely to act upon such message, for example, by clicking on links. A similar result was reported by Jagatic et al. [177]. The authors found that phishing attempts are more likely to succeed if the attacker uses stolen information from victims' friends in social networks to craft their phishing e-mails.

In contrast to active social engineering that requires the attacker to establish contact with the victim, in a reverse social engineering attack, it is the victim that contacts the attacker.

Irani et al. [175] conducted a study on reverse social engineering attacks in online social networks. The paper shows that reverse social engineering attacks are a feasible threat in real-life, and that attackers may be able to attract a large numbers of legitimate users without actively sending any friend request. The authors conducted some experiments that demonstrated how suggestions and friend-finding features (e.g., demographic-based searches) made by social networking sites may provide an incentive for the victims to contact a user if the right setting is created (e.g., an attractive photograph, an attack profile with similar interests, etc.).

6.6 Attacks through Malicious Applications

Many social networks offer the possibility to create additional applications that extend the functionality of the network (e.g., the Facebook application platform [14], and Open Social [12]). To seamlessly embed an application into the social network, the platforms provide libraries to third-party developers. Those libraries contain the bindings for different programming languages to easily access the functionality and data of the social network.

With the tight integration between the social network and third-party applications, privacy issues arise, especially when it comes to the handling of sensitive profile data. Once an application obtains access to profile data, it is impossible for the social network to further enforce or asses how this data is used by the application, lacking technical means to enforce.

For example, in an incident involving the "Top Friends" Facebook application [109], everybody could access the birthday, relationship status, and gender of all Top Friends users, even in cases where this information was set to be private by those users. A more recent incident [314] involved some of the most popular Facebook applications transmitting user information to advertising and Internet tracking companies.

Privacy concerns with regard to online social networks applications attracted also the attention of the research community. In [126], Felt et al. evaluated the requirements of personal data for 150 popular Facebook applications. They conclude that only 9% of the evaluated applications need to access personal profile data to work correctly. The application framework introduced in [307] is designed to keep all personal profile data confined. To this end, the xBook framework provides a restricted JavaScript environment based on ADSafe, extended with data storage capabilities.

Shehab et al. [306] introduce an three step approach for Facebook application access control. First, upon registration, each application has to submit a so-called application sheet, specifying the data needs for this ap-

plications. The second step consists of a so-called user sheet, reflecting the access control decisions the user made for each element of the application sheet. Finally, the third step covers the necessary modifications the application has to undergo to cope with data that it cannot read because access is denied by the user sheet.

Egele et al. [110] presented PoX, an extension for Facebook that makes all requests for private data explicit to the user and allows her to exert fine-grained access control over what profile data can be accessed by individual applications. By leveraging a client-side proxy that executes in the user's web browser, data requests can be relayed to Facebook without forcing the user to trust additional third parties.

6.7 Conclusions

In this chapter we presented a number of serious attacks against the user privacy on social networking websites. Sometimes, countermeasures are available, sometimes they are not.

One possible way to protect the user's privacy against malicious application, service vulnerabilities, and misused functionalities, consists in applying cryptographic techniques "on top" or to replace existing social networks.

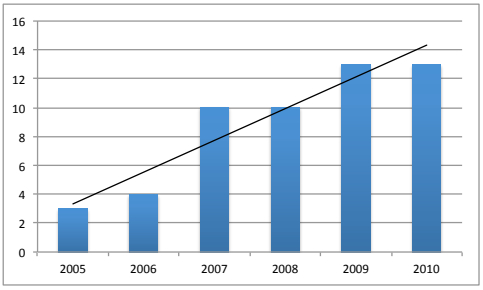
For example, Lucas et al. [211] propose FlyByNight, a cryptographic system that encrypts all communication between users on the Facebook Platform. The authors implemented a proof of concept Facebook application that relies on asymmetric key cryptographic methods to encrypt messages with their respective receiver's public keys. The purpose of FlyByNight is to make communication in the social network unaccessible to the social network operator. The system does not, however, protect the data stored in a user's profile.

Another method to protect the data of Facebook users was presented by Lou et al. [212]. In their approach, they store fake information on the Facebook site, but keep the real data encrypted on a separate server. Finally, researchers have gone as far as proposing new, completely distributed, peer-to-peer social network [83] to protect the users' privacy.

7 Web Attacks

7.1 Introduction

The purpose of this section is to give an overview on web attacks and dedicated research. Per definition, web attacks incorporate a very broad spectrum of security related topics, including parts of the Chapters 6,4 and 8. While these chapters discuss some major problems in detail, this section deals with the following topics:



- Drive-by downloads: Strongly related to malware and its distribution channels, drive-by downloads are still an important distribution technique to infect computer systems with malware. The exploited vulnerability is always targeted at the web Browser and delivered by arbitrary web pages with bogus payload. Therefore, it is the most common form of web attacks.

- Directly exploiting web pages and their infrastructure is still possible by various means. SQL injection (SQLI), Cross-site-scripting (XSS) and Cross-site request forgery (CSRF) are examples for directed attack techniques. Even though these topics received quite some attention from researchers in the past ten years, the underlying problems are still not completely solved.

- Web spam is neither directed at a specific web site nor at the clients themselves. Instead, an attacker tries to influence the results returned by search engines, inject messages to forums or use other means of

distributing advertisement to web resources. Although not a direct attack per se, web spam is a huge annoyance and has a negative impact on both, the browsing experience of the user and the provider of the web resource itself.

An all-embracing discussion of the work published in these areas would clearly go beyond the scope of this chapter. Therefore, we will discuss the most recent approaches and advances in some selected areas of the above iteration in the following sections.

7.2 Drive-by Downloads

Drive-by-download attacks [272], [273] exploit vulnerabilities in web browsers and web browser components. These attacks are triggered when a victim uses her browser to load a web page that contains malicious code. To attract potential victims, attackers can prepare malicious web pages and distribute their URLs (e.g., by including links in spam mails). Alternatively, attackers can compromise existing web sites and inject malicious code into search engine results or legitimate pages (e.g., by embedding iframes). Typically, the exploit code used in a drive-by-download attack is written in a client-side scripting language, such as JavaScript or ActionScript (as part of an Adobe Flash file). This code is directly run by the browser or executed with the help of a browser extension. When the exploit is successful, it downloads and installs malware on the victims machine, frequently in an effort to recruit additional members for a botnet [266]. Over the last few years, drive-by-download attacks have become the most popular means used by cyber-criminals to compromise and infect hosts. The reason for the popularity of this type of attack is that direct attacks against hosts and their operating systems have become more difficult. This is in part due to the growing efforts by Microsoft to improve the security of its Windows products, but also due to the fact that users are increasingly shielded by Firewalls and NAT devices (such as home routers). Given the importance of drive-by-download attacks, researchers have recently proposed first steps to harden browsers against these exploits [113], [281]. Moreover, a number of papers have presented approaches to detect malicious web pages [217], [342] and to study their prevalence by crawling large portions of the Internet [273], [272]. Finally, follow-up work has studied the behavior of web-based malware once a machine is compromised [266]. However, malicious web pages and web-based malware do not live in isolation. Someone has to develop the code used for drive-by-download attacks, infect web sites, and set up exploit servers to which victims are redirected, so that they can download the latest malware instances. Also, malicious sites rely on visitors with vulnerable software components to carry out successful attacks. These aspects are all important pieces of a drive-by campaign, which are characterized as

a coordinated effort of a group of cyber-criminals to propagate malware via drive-by-download attacks.

7.2.1 Campaigns

So far, researchers have focused on malicious web sites and their prevalence, mostly neglecting other aspects of drive-by-download campaigns. However, a more comprehensive study of these campaigns is important and has the potential of revealing a number of interesting facts about the various parties that are involved. In particular, an understanding of the modus operandi of cyber-criminals, the ways in which they craft and update their exploit code, and the infrastructure they use to host their attacks is desirable. In addition, it is interesting to see how the exploit code is distributed and which web sites are targeted as part of a single campaign. In [316], the results of an in-depth study of the Mebroot drive-by-download campaign are presented. The study was carried out over one year, covering different time periods from May to September 2009, and one week in April 2010. The approach was based on infiltrating a running campaign by sinkholing the web sites used by the Mebroot authors to launch exploits against their victims, and by directly monitoring the network traffic on an exploit server. From these internal vantage points, direct visibility of the victims of the campaign and of the infected web sites that were unwittingly redirecting their visitors to the exploit sites was available. Apart from the campaign-specific results, the study showed how browser update capabilities influence the impact of a campaign.

Figure 7.1 shows that Google Chrome is by far the browser with the most up-to-date clients, followed by Firefox. Internet Explorer, on the other hand, is still a popular target for attacks directed at the browser or its plugins. Besides, not all attacks are targeted at a specific browser vulnerability. In a lot of cases, the exploit is targeted at a plug-in like the flash player or the pdf reader. Summed up, drive-by-download campaigns affect a significant number of web users and an alarming fraction of these users rely on vulnerable systems, such as old versions of the browser or vulnerable additional components. Similarly, on the server-side, web masters of infected sites are slow at removing malicious code injected in their pages and often, when they do, they are prone to be infected again. Therefore, an efficient defense against large drive-by campaigns is hard to devise and implement.

7.2.2 Defenses

Depending on the attacked vulnerability, several approaches to mitigate drive-by download attacks exist[113]. For malware analysis, two different approaches exist. While dynamic analysis actually executes the malware,

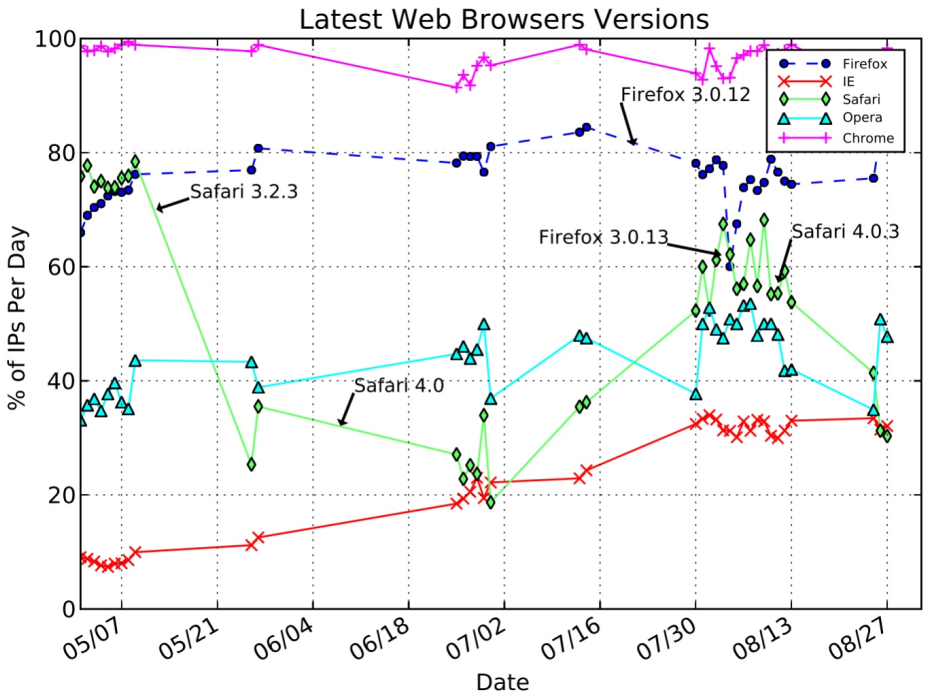


Figure 7.1: Browser version

static analysis is performed without running the software in question. Dynamic approaches execute the malware in a controlled environment, and observe the interaction of the malicious component with the environment. Hooking API function calls results in detailed information of the behavior of a program. CWSandbox [346] uses hooking to log the invocations of Windows API function. A mixture of static and dynamic techniques is applied by Kirda et al. [193] to detect malicious browser plug-ins. Egele et al. performed information flow analysis on browser plug-ins [112] to identify spyware components that leak sensitive information. Information flow analysis is also the key idea of Panorama [356], where Yin et al. implemented a system to discover rootkits.

While powerful, existing analysis techniques are typically too heavyweight to be used for detection on a client machine. In contrast to that, current techniques detect drive-by download attacks by monitoring potentially malicious scripts directly in the browser. Previous studies have shown that drive-by download attacks pose a real threat to the Internet and its users. The mechanisms used by attackers to mount their attacks are investigated by Provos et al. in [273]. The life cycle of an infected machine is analyzed by Polychronakis in[273]. In [272], Provos et al. present a measurement study that reports that the results for 1.3% of all Google search queries contain at least one link pointing to a page that performs a drive-by

attack. Also, Frei et al. analyzed the vulnerability landscape of web browsers in the Internet. Apparently, only 60% of the users that navigate the Internet everyday use the latest, most secure version of their web browser. Based on a Secunia report, the authors argue that many browser plug-ins commonly in use have known vulnerabilities. The fact that many users only reluctantly update their web browsers and plug-ins makes it feasible for attackers to distribute attacks that target old vulnerabilities. As many of the vulnerabilities leading to control flow hijacking are present in ActiveX components, Dormann and Plakosh [1] propose fuzzy testing as a means of detecting such flaws before distributing a component. Detecting shellcode in network traffic has a long standing history. Network intrusion detection systems, such as Snort [290] or Bro, rely on signatures to identify malicious network streams. While signature detection works well for known static threats, advanced polymorphic shellcode and engines that can automatically produce such shellcode can sometimes evade these detection techniques. Based on abstract payload execution, Toth and Kruegel have proposed a mechanism to detect buffer overflow attacks [329]. More precisely, their prototype implementation identifies long valid sequences of instructions in HTTP requests, thus detecting the NOP sledge that commonly accompanies shellcode. Continuing this work, Polychronakis et al. [266] proposed to apply lightweight emulation on network data to identify polymorphic shellcode. This approach relies on the so-called GetPC heuristic. That is, a shellcode is only identified if a sequence of instructions is emulated that reads the current program counter value. The class of non-self-contained shellcode, however, contains code that reaches its goal without showing such behavior. While network-traffic-based techniques are useful, they typically cannot be used to detect drive-by downloads. The reason is that, although JavaScript contents of a web page are transmitted over the network, this code is often obfuscated. Furthermore, the shellcode contained in the JavaScript scripts are not transmitted in binary form. Instead, the ASCII representation of the individual bytes is transmitted. This sequence does not yield a valid instruction sequence in general. Analyzing malicious JavaScript has recently gained more attention by the scientific community. Hallaraker and Vigna [153] present an approach to audit the execution of JavaScript code. These audit logs can be compared to high-level policies to detect potential attacks. Similarly, Feinstein and Peck introduced Caffeine Monkey [125], a tool that supports the collection and analysis of malicious JavaScript. To this end, they extended the Mozilla SpiderMonkey JavaScript engine by adding run-time logging facilities. Chenette et al. aim at automatically reversing the obfuscation of malicious JavaScripts. Their approach relies on hooking techniques to monitor calls to relevant JavaScript functions, such as eval or document.write. Vogt et al. propose a system that prevents cross-site script-

[1] http://www.cert.org/archive/pdf/dranzer.pdf

ing attacks performed by malicious JavaScript code [338]. To protect a user from JavaScript that tries to steal sensitive information, the propagation of such information through the JavaScript engine is tracked. Requests to a domain containing information originating from another domain raise an alert, and allow the user to stop further execution of the script.

With all these approaches, drive-by downloads are still one of the primary distribution channels for malware. The most important requirement has to be met by browser vendors. In the end, they have to make sure, that security patches are applied automatically and without necessary user interaction to keep at least old vulnerabilities from being exploited in a malware campaign.

7.3 Direct attacks

While drive-by downloads are targeted at a huge amount of computers to infect them with malicious software, several attack scenarios exist, where the web hosting environment is specifically chosen as a target.

7.3.1 Cross-site request forgery (CSRF)

Cross-Site Request Forgery (CSRF) is a web application attack vector with which an attacker forces an unwitting user's browser to perform actions on a third party website, possibly reusing all cached authentication credentials of that user. In 2007, CSRF was listed as one of the most serious web application vulnerabilities in the OWASP Top Ten. In 2008, Zeller and Felten documented a number of serious CSRF vulnerabilities in high-profile websites, among which a vulnerability on the home banking website of ING Direct, which allowed an attacker to transfer funds from any user account to an account chosen by the attacker. One of the root causes of CSRF is the abuse of cached credentials in cross-domain requests. Web applications can easily trigger new requests to web applications in a different trust domain without any user intervention. This results in the browser sending out cross-domain requests, while implicitly using credentials cached in the browser (such as cookies, SSL certificates or login/password pairs). From a server point of view, these implicitly authenticated requests are legitimate and are requested on behalf of the user. The user, however, is not aware that he sent out those requests, nor that he approved them. Currently, a whole range of techniques exist to mitigate CSRF, either protecting the server application or protecting the end-user (e.g. via a browser plugin or a client-side proxy). However, the server-side protection mechanisms are not yet widely adopted, and most of the client-side mitigation techniques only provide limited protection, or do not scale well to web 2.0 applications. As a result, even the most cautious web user is not always able to appropri-

ately protect himself against CSRF. Combinations of client- and server-side techniques [218], [41] are often employed to mitigate impact and severity of CSRF attacks.

7.3.2 Cross-site scripting(XSS)

Closely related to CSRF, Cross-site scripting (XSS) attacks are also deemed among the number-one security threats on the Internet today. These attacks breach confidentiality of sensitive data, undermine authorization schemes, defraud users, defame web sites, and more. The web site www.xssed.com documents recently successful XSS attacks on major blog and social networking sites. Notably Facebook, LiveJournal, MySpace and Orkut have all been hit by these attacks. XSS attacks can be self-propagating, and have the potential to rapidly victimize millions of people. Broadly speaking, XSS is injection of unauthorized script code into a web page. As a web application processes input from untrusted users, it generates some low-integrity output web content which we term untrusted HTML. The goal of an XSS attack is to embed malicious script code in untrusted HTML, causing the script to be executed on a victims web browser within the context of the conduit web application. We say the attack script is unauthorized because the application does not intend to allow scripts in untrusted HTML. Defenses for XSS aim to prevent unauthorized script execution by enforcing a no-script policy on untrusted HTML.

To disallow script execution in untrusted web content, a web application might possibly take one of the following approaches.

- Content Filtering: The application may attempt to de- tect and remove all scripts from untrusted HTML before sending it to the browser.
- Browser Collaboration: The application may collaborate with the browser by indicating which scripts in the web page are authorized, leaving the browser to ensure the authorization policy is upheld.

Content filtering. Content filtering is otherwise known as sanitization. This defense technique uses filter functions to remove potentially malicious data or instructions from user input. Filter functions are applied after user input is read by a web application, but before the input is employed in a sensitive operation or output to the web browser.Removal of scripts from untrusted content is a difficult problem for web applications that permit HTML markup in user input such as blog, wiki and social networking applications. These applications are expanding and proliferating rapidly, thus the growing need for robust XSS defenses. The WordPress blog platform is one popular application that empowers anonymous users to control the presentation of their blog comments. It does so by permitting input of structured HTML elements for text formatting (e.g., <b> for bold, <i> for ital-

ics). Content filtering based defenses for this type of application face a difficult challenge: allowing all benign HTML user input, while simultaneously blocking all potentially harmful scripts in the untrusted output. Simply disallowing HTML syntax control characters is not a practical filtering solution for these applications because every control character that can be used to introduce attack code also has a legitimate use in some benign, non-script context. For example, the < character needs to be present in hyperlinks and text formatting, and the " character needs to be present in generic text content. Both are legitimate and allowed user inputs, but can be abused to mount XSS attacks. Advanced content filters try to anticipate how untrusted content will be interpreted by the client web browser's parser, as it is the browser parser that makes crucial decisions about script execution. To be completely effective in eliminating XSS, a filter function must necessarily model the full range of parsing behaviors pertaining to script execution for several browsers. This is a very difficult problem, as diligently documented in the XSS Cheat Sheet, which describes a wide variety of parsing quirks exhibited by different browsers. Quirks are essentially anomalous browser parser behaviors that either contradict language standards or account for conditions not well defined by these standards (such as how to parse malformed HTML). They are sometimes intentionally introduced and retained in a browser's code base to correctly render existing web sites that depend on the quirks of older browsers. Quirks vary by browser, are complex to model, not entirely understood and not all known (especially for closed-source browsers). Therefore, from a web application perspective, the task of implementing correct and complete content filter functions is very difficult, if not impossible.

Browser collaboration. Robust prevention of XSS attacks can be achieved if web browsers are made capable of distinguishing authorized from unauthorized scripts. This vision was first espoused in BEEP [181], wherein this approach was implemented by (a) creating a server-browser collaboration protocol to communicate the set of authorized scripts, then (b) modifying the browser to understand this protocol and enforce a policy denying unauthorized script execution. Although the defense strategy envisioned by the authors of BEEP is a compelling and effective long-term solution, their implementation approach leaves a large void in near-term protection. This is because web applications adopting this approach require their users to employ custom BEEP-enabled browsers for protection from XSS attacks. Amendments to these shortcomings were introduced by Louw et.al [210], where the authors introduce a system that provides robust XSS protection while being compatible with existing browsers and still allowing benign HTML content.

7.3.3 SQL injections

SQL injection is a code injection technique that exploits a security vulnerability occurring in the database layer of web applications. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another. Curiously, this form of attack is known for more than 10 years, and bulletproof [52] defenses have been devised to protect companies and services from attacks to their databases. By using prepared statements or object-relational mappings, secure queries can be guaranteed. Nevertheless, SQL injections are still possible in several web pages. The most famous incident occured in March 2011, when Mysql.com was compromised by a blind SQL injection attack[2]. Such incidents show, that even major players are prone to vulnerabilities where a mitigation solution already exists. In the end, there will always be programmers under time constraints who decide to implement a quick solution without caring about security implications.

7.4 Web spam

The third topic discussed in this Chapter deals with a very common nuisance that every far-reaching medium has to deal with: Advertisement. The term web spam refers to hyperlinked pages on the WorldWideWeb that are created with the intention of misleading search engines. For example, a pornography site may spam the web by adding thousands of keywords to its home page, often making the text invisible to humans through ingenious use of color schemes. A search engine will then index the extra keywords, and return the pornography page as an answer to queries that contain some of the keywords. As the added keywords are typically not of strictly adult nature, people searching for other topics will be led to the page. Another web spamming technique is the creation of a large number of bogus web pages, all pointing to a single target page. Since many search engines take into account the number of incoming links in ranking pages, the rank of the target page is likely to increase, and appear earlier in query result sets. Just as with email spam, determining if a page or group of pages is spam is subjective. For instance, consider a cluster of web sites that link to each other pages repeatedly. These links may represent useful relationships between the sites, or they may have been created with the express intention of boosting the rank of each others' pages. In general, it is hard to distinguish between these two scenarios. However, just as with email spam, most

[2] http://blog.sucuri.net/2011/03/mysql-com-compromised.html

people can easily identify the blatant and brazen instances of web spam. For example, most would agree that if much of the text on a page is made invisible to humans (as noted above), and is irrelevant to the main topic of the page, then it was added with the intention to mislead. Similarly, if one finds a page with thousands of URLs referring to hosts like

```
buy-canon-rebel-300d-lens-case.camerasx.com,
buy-nikon-d100-d70-lens-case.camerasx.com,
...,
```

and notices that all host names map to the same IP address, then one would conclude that the page was created to mislead search engines. (The motivation behind URL spamming is that many search engines pay special attention to words in host names and give these words a higher weight than if they had occurred in plain text.) While most humans would agree on the blatant web spam cases, this does not mean that it is easy for a computer to detect such instances. Search engine companies typically employ staff members who specialize in the detection of web spam, constantly scanning the web looking for offenders. When a spam page is identified, a search engine stops crawling it, and its content is no longer indexed. Several approaches try to tackle this problem by introducing automated techniques that detect sites that are likely to be spam or that are likely to be reputable. Amitay et al.[25], for instance feed connectivity features of pages into a rule-based classifier, in order to identify link spam. Baeza-Yates et al. [35] present a study of collusion topologies designed to boost PageRank while Gyoengyi et al. [150] introduce TrustRank which finds non-spam pages by following links from an initial seed set of trusted pages. Wu and Davison [349] and Gyoengyi and Garcia-Molina [150] study how to detect link farms (i.e. sites exchanging links for mutual benefit). Finally, Mishne et al. [230] present a probabilistic method operating on word frequencies, which identifies the special case of link spam within blog comments.

Recently, new forms of web spam emerge, where the attacker takes advantage of meta-information like location or spoken language to create even better fitting spam links. The main goal still is to blur the line between legitimate search results and artificially injected links.

7.5 Conclusion

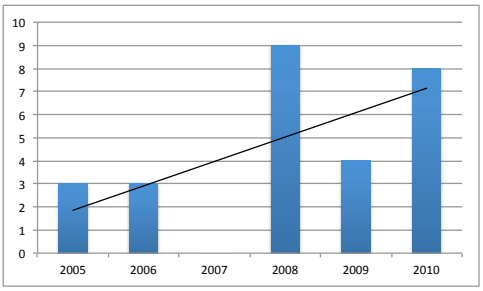
Web attacks, in a broader sense, are still gaining momentum. With the huge amount of participating individuals, where most only have a sparse security awareness, a multitude of attack vectors become practicable. While attacks targeted at the server infrastructure try to exploit very specific vulnerabilities, client-side attacks are now predominant. The reason is simply that most users tend to wait with applying security patches for their systems,

which can in turn be exploited fairly easy. Raising security awareness, on the other hand, has always been deemed a brawny task which cannot be done overnight. Therefore, this form of attacks will also be seen in the future, leaving researchers with the task to devise appropriate countermeasures and mitigation techniques.

8 Network-level Attacks

8.1 Introduction

In this section we will be covering two major classes of network-level attacks. Namely, attacks on the Border Gateway Protocol (BGP) and the Domain Name System (DNS). We focus on these protocols, as they form the backbone of today's Internet, and any compromise on their integrity or operation can lead to significant security problems. In the rest of the section we go over the protocols, the corresponding attacks as well as the proposed defenses.



8.2 BGP Overview

The Internet is a global system of interconnected computer networks. Networks are comprised of end systems, called hosts, and routers, each with one or more IP addresses. For the Internet to be connected there must be a way for the packets of any host to reach any other host even if they are not on the same network. This requires that data flows from one network to another until it reaches its final destination. Routing protocols are used by the routers to discover paths to each destination. On the Internet, networks and hence routers belong to different heterogeneous organizations like universities, private corporations, government agencies, etc. These entities are reluctant to share information about the structure of their networks. Moreover, they wish to be able to freely administer and organize their network according to their needs and not based on a common standard. This makes

connectivity among them a challenging task since each entity tries to balance between connectivity and reachability while on the same time provide as few information as possible.

The Internet is comprised of Autonomous Systems (AS), an autonomous system is an organizational unit which controls a number of networks and has a clearly defined routing policy. Each AS can control many networks but each network is controlled by exactly one AS. Also, each AS is responsible for delivering packets to its hosts and forward packets it receives in one of its border routers to the appropriate border router to continue to the next AS. The AS can choose its internal structure freely and it does not need to publish any information about it. The only information it needs to publish is its border routers. A border router is a router that connects an AS with another AS. For two ASes to communicate they must configure their border routers as having each other as neighbors. An AS can have many border routers each with different neighbors. This is possible because the network of an AS can span across a large geographic area and hence connect with different ASes in different areas. Usually, the organizations decide to peer border routers that have a direct physical connection even if this is not necessary. IP address assignment is directly connected with the notion of AS and consequently to routing. The Internet Assigned Numbers Authority (IANA) through its local registries (ARIN, RIPE, etc) assigns to each AS a unique number (ASN) from 1 to 64511. Also through the same process IANA assigns to each AS blocks of IP addresses, called IP *prefixes*. These IP addresses are contiguous and are represented by the first address and a mask length. The delegation of address blocks is hierarchical, so an AS can in turn delegate a part or the whole of its prefix to another AS without notifying IANA. In practice, IANA delegates large IP prefixes to the local registries which then further delegate them to national registries, that finally assign prefixes to ASes. The delegations can cascade even more levels. E.g. an AS can further delegate part of its prefix to another AS.

The Border Gateway Protocol (BGP) is designed to help ASes decide the best path for a given destination IP. The protocol works as follows: The border routers of the AS advertise to their neighbors that they have a path for some IP prefix. The AS that introduces a prefix in the global routing system is called the *originating* AS of this prefix. The advertisement lets ASes neighboring the originating AS know that traffic destined to that IP prefix should be forwarded to the latter, which knows how to handle it. Additionally, the neighboring ASes forward the advertisement to their own neighbors, adding the information that the path for the given IP prefix passes through them to reach the originating AS. So, the neighbors of the neighbors also have a route for the particular IP prefix. The ASes continue to propagate the advertisement, adding themselves to the path, until all ASes have a route to that prefix.

For instance, in Figure 8.1 let's assume that the AS 1 is in charge of the IP prefix 128.192.0.0/16. The border router of AS 1 advertises to the border routers of AS 2 and AS 3 that it can handle packet destined to that IP block. AS 2 and AS 3 now know how to route packets destined to 128.92.0.0/16. But the other ASes still have no route for those packets. That is why AS 2 propagates the advertisement to AS 4 and AS 3 to AS 5 and AS 6. These advertisements include the ASN of the originating AS, AS 1, and also the ASN for the ASes that the packet must traverse to reach their destination, AS 2 and AS 3 respectively. So now, AS 4, AS 5 and AS 6 know that packets destined to 128.92.0.0/7 the originating AS is AS 1 and to reach that AS packet must pass though AS 2 or AS 3 accordingly. AS 5 will receive two advertisements, one from AS 4 and one from AS 3, based on policies defined by the administrator and on metrics defined on the BGP it will choose one to forward the packets but it will continue to remember the other. This is important for redundancy, if for example the link between AS 1 and AS 2 goes down then AS 5 will advertise the route through AS 3 to AS 4 and AS 3 so that they can still connect to AS 1.

In the previous example, each AS forwarded all the advertisements it received to all its neighbors. But this is not always the case, as each AS may choose which routes wishes to advertise to each neighbor. The main reason for choosing not to forward an advertisement is to avoid the creation of circular routes. This would happen if some AS forwarded a route advertisement which already contains the AS itself in the routing path. Additionally, the choice of which of the multiple route advertisements for a given prefix an AS propagates may be based on economical criteria and/or existing policies and agreements. E.g. if AS A receives route advertisements for 128.192.0.0/16 from both ASes B, C and it cheaper for A to forward traffic through B, it will (usually) choose to further propagate only that route advertisement. Another common case is an AS choosing to advertise routes through other ASes controlled by its parent organization even though they are not the shortest.

ASes are able to delegate a subset of their prefix to other ASes. For instance the 210.0.0/7 is assigned to APNIC which in turn delegates the 211.120.0.0/12 to JPNIC. This means that the small prefix will have a different path than the /7 prefix. Therefore APNIC cannot advertise a single prefix, /7, without stealing the traffic from JPNIC. The one way to solve with this problem is to partition the /7 network to many /12 and assign the 211.120.0.0/12 to JPNIC and the all the other to APNIC but that would increase the size of routing tables prohibitively, since each border would have to hold 256 different routes. So the BGP support the option, called *longest prefix match*, for choosing the most specific prefix. This means that it has a generic route 210.0.0/7 with destination to APNIC and a more specific route 211.120.0.0/12 to JPNIC. Packets are forwarded towards the most specific

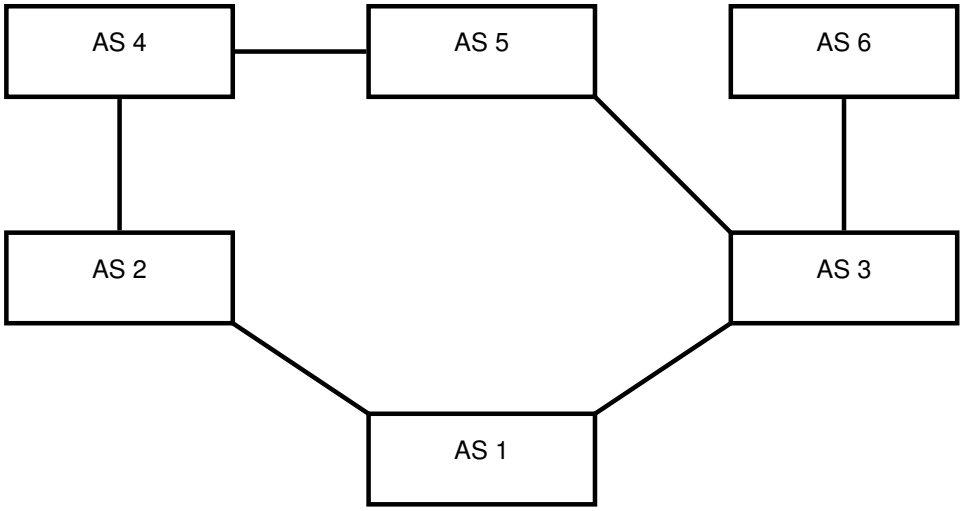


Figure 8.1: Connectivity graph of Autonomous Systems

destination that matches their address. This greatly reduces the size of the address tables but it is also an attack vector, as we will see later.

One of the BGP weakness is that it only provides weak security guarantees. BGP does not ensure that BGP-speaking routers use the ASN they have been allocated or that they hold the IP prefixes they advertise. The neighboring ASes will accept these advertisements, if not configured otherwise, and propagate to the rest of the global routing systems. This makes BGP vulnerable to misconfiguration and malicious attacks.

The rest of the chapter is organized as follows. In section 8.3 we discuss possible attack vectors on the BGP protocol, the section draws upon info from [255]. In section 8.4, we provide an overview of the security solutions as discussed in [57].

8.3 BGP Attacks

Before discussing the various attack mechanisms we will focus on the objectives of the attackers. When an attacker attacks the global routing systems aims in one of the following outcomes.

Make a destination unreachable for a portion or the entire Internet. Usually, a malicious BGP router makes false advertisements to attract traffic destined to a prefix and then just drops the packets it receives. As a result the advertised prefix becomes unavailable since packets cannot reach their destination. The extent of the attack depends on how far the malicious advertisements will propagate. This effect is called *Blackholing*.

Redirect traffic for eavesdropping or attacking. The attacker tries to alter the legitimate path with another path that servers its malicious purposes.

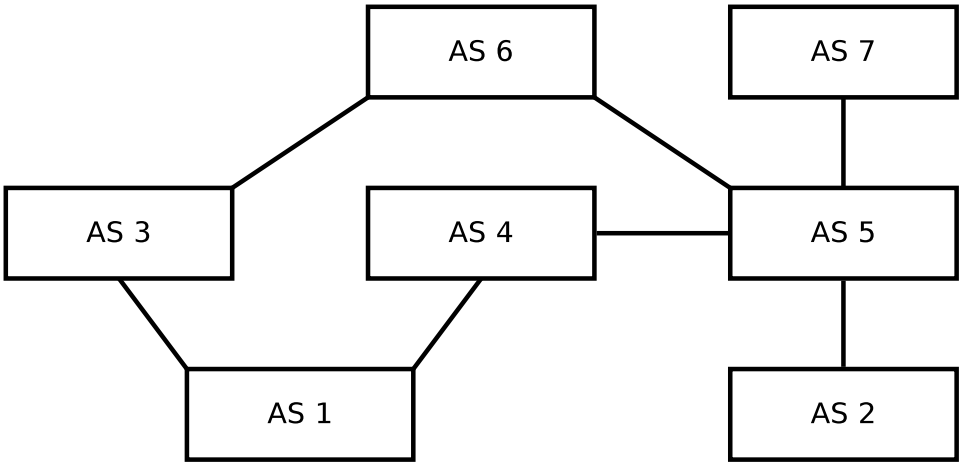


Figure 8.2: Internet topology

The forged path can pass through a router controlled by the attacker. In this case, the attacker can perform a *man-in-the-middle* attack. Or the attacker can change the destination of the path to target in a malicious host. The malicious host can *impersonate* the legitimate one so as to extract sensitive information. Furthermore, the attacker can include in the path a benign router to overwhelm it with the extra traffic, essentially performing a *denial-of-service* attack.

Lastly the attacker may wish to cause *instability* in the global routing system. This involves announcing the same route with different originating AS or with different attributes. The benign border routers usually block routes that experience instability, consequently the prefix loses its connectivity. The methods to achieve the previous objectives are described below as well as in [255].

8.3.1 Attack Mechanisms

For the attacker to achieve his goal, he usually needs to use a compromised BGP router. He can use this router to create, modify or drop BGP updates. The effectiveness of each attack can be affected by the AS topology. As we see in Figure 8.2 some routers are in more critical positions than others. For instance in the event that $AS\ 5$ fails $AS\ 7$ and $AS\ 2$ become unreachable. On the other hand a failure in $AS\ 3$ or $AS\ 4$ has no effect on the reachability of the other ASes.

8.3.1.1 Prefix Hijacking

The BGP protocol provides no means to validate that a BGP-speaking router belongs to the AS it claims it belongs or that it actually has ownership over

the IP prefixes it advertises. A malicious router can advertise prefixes unassigned or assigned to any AS to accomplish its malicious objectives. Such advertisements would propagate through its neighbors and reach all other ASes. The other ASes have no way of knowing that such advertisements are legitimate. In Figure 8.2, AS 2 is the legitimate owner of the prefix $120.20.0.0/16$. A malicious AS, in this case AS 1, can advertise the exact same prefix. Now there are two possible routes in the system for the given prefix. The other ASes will consider both of them legit and choose the best according to their policy. The most common policy dictates the use of the path with the fewer intermediate ASes. According to that policy AS 3, AS 4 and probably AS 6 will start forwarding traffic for $120.20.0.0/16$ to the malicious AS 1. This attack is known as *prefix hijacking*.

The attack can be used to create the blackholing effect discussed previously. AS 1 will receive the traffic destined to $120.20.0.0/16$ and just drops it, making hosts in the $120.20.0.0/16$ seems unreachable for parts of the Internet, in this case for AS 3 and AS 4. A more sophisticated attacker could try to *impersonate* the legitimate destination. Assume that a host in the $120.20.0.0/16$ services a web-banking application. Since AS 1 will receive the traffic destined to the web-banking application it can answer back immitating the legitimate service. Thus, making it possible to steal sensitive information like credit-card numbers and passwords.

An even more elaborate attacker can try can exploit the *longest prefix match* attribute of the BGP. The malicious AS advertises a subset of the prefix advertised by the legitimate AS. As discussed in the Section 8.2 routers tend to prefer routes that match the destination address as much as possible. So, if a malicious AS advertise a prefix that is more specific than the legitimate advertsiment, for instance the legitimate AS advertises the $120.20.0.0/16$ whereas the malicious AS the $120.20.0.0/17$. All other ASes will prefer to route traffic destined to $120.20.0.0/17$ to the malicious AS even if they are closer to the legitimate one. In figure 8.2, the legitimate AS 2 advertises $120.20.0.0/16$ whereas the malicious AS 1 advertises the $120.20.10.0/24$. The advertisement of AS 1 is more specific then the advertisement of AS 2 as a consequences all ASes will prefer to send traffic for $120.20.10.0/24$ to AS 1 instead of AS 2. If AS 1 wishes to hijack all traffic of AS 2 then it can just *de-aggregate* the $120.20.0.0/16$ and advertise 256 /24 prefixes.

The problem of multiple ASes advertising the same prefix have been studied by Zhao et al in [362] which conclude that they are very common on the Internet. This kind of conflict does not always indicate a malicious attack, they may be router misconfiguration or anomalies after a path change and until the BGP converges.

8.3.1.2 Contradictory Advertisements

ASes usually announce more than one route per prefix for redundancy reasons. Among these routes the originating AS usually wishes only one to be used whereas the others are for backup only. One way to accomplish that is to artificially make one of the paths longer by prepending the originating routers' ASN in the path multiple times. This creates the illusion that the other advertised path is shorter, so it's more likely for other ASes to prefer it. For instance, let us assume the $AS\ 1$ wishes to advertise the route through $AS\ 3$ as its main route and the route through $AS\ 4$ as a secondary backup route. The advertisement to $AS\ 3$ would look something like $[AS1]$, where the advertisement to $AS\ 4$ will look like $[AS1, AS1, AS1]$ making sure that the other path, through $AS\ 3$, is shorter. Because the route through $AS\ 3$ is shorter then all ASes should prefer it from the route through $AS\ 4$. A malicious attacker controlling $AS\ 5$ can drop the advertisement through $AS\ 3$ and propagate only the advertisement though $AS\ 4$. Since, $AS\ 2$ and $AS\ 7$ have no way of knowing about the route through $AS\ 3$ they will forward their traffic through the $AS\ 4$. The traffic may congest $AS\ 4$ or according to the agreement between $AS\ 4$ and $AS\ 1$, $AS\ 1$ may be charged for the traffic it receives through $AS\ 4$.

8.3.2 Route Manipulation

Route manipulation includes route dumping and route instability. Route dumping is an attack that forces a legitimate router to stop using, dump, a valid route. This can be done by announcing and withdrawing the target route at a sufficient high rate. The router that receives the advertisements will react to the fluctuations by dropping the route. This is done for the protection of itself and the whole system's. The use of a new route requires the update of the routing table and the propagation of the advertisement to the neighboring routers. This process is quite expensive to be performed continuously. Additionally, when the router propagates the advertisement will trigger similar update operation to the neighboring routers and maybe to the system as a whole. That is why routers dump routes that exhibit high instability.

Suppose $AS\ 6$ is a malicious AS and wishes to perform denial-of-service attack to $AS\ 1$. Using route dumping $AS\ 6$ can make all routes to $AS\ 1$ being dumped from $AS\ 5$. To do this $AS\ 6$ announces the route $[AS\ 1, AS\ 3, AS\ 6]$ to $AS\ 5$ and then quickly withdraw it and announce a different route to $AS\ 1$ like $[AS\ 1, AS\ 4]$. Then repeat the process at a high rate. As discussed earlier, the $AS\ 5$ will dump any route to $AS\ 1$. It has been shown with only one withdraw and re-announcement it is possible for some routers to dump a route for almost an hour [220].

An attacker may want to avoid causing route dumping since this is a very drastic solution and at a best case it will perform a DoS attack. Another approach it to make a route fluctuate in a rate slow enough to prevent route dumping but still high enough to cause route instability. This kind of attacks can be the cause of a lot of disturbance in the routing system, even temporarily. The victims AS can become unavailable, unreachable, traffic can be forced through specific ASes for eavesdropping or to cause congestion [73]. In other words it can degrade the routing performance in the whole Internet. Another way to cause instability is presented in [43]. It is based on link cuts. In this model the attacker knowns the topology of the links and routers. He then calculates which links must be disabled to force the traffic though a compromised AS. Then the compromised routers can launch attacks that can eavesdrop and blackhole the victim's traffic.

8.3.2.1 Congestion-induced BGP session failures

The BGP protocol is based on TCP for the communication between the routers. TCP has been chosen for its reliability which ensures that the updates will be delivered to their destination. One drawback of TCP is that its throughput dramatically falls when operating on congested links. Effectively, during heavy congestion, the TCP session between two routers may become so slow that is dropped, causing the routers to stop receiving and propagating updates. When the TCP session finally is established, the routers need to exchange entire routing tables multiplying the need for bandwidth, also the convergence time increases significantly. An adversary who wishes to degrade the quality of the BGP system can exploit this defensive property of the TCP by flooding carefully selected links. Making the communication between the border routers difficult, in the short-term routers end up with inconsistent views of the network, which someone can take advantage of. In the long-term routers will have to exchange a lot of information, maybe entire routing tables, to converge on a consistent view multiplying the need for bandwidth and increasing the convergence time dramatically.

8.4 BGP Defenses

In this section, we will discuss some of the techniques implemented or proposed for securing the Internet routing system from malicious attacks. In Section 8.4.1 we discuss the currently deployed techniques for BGP security. In section 8.4.2, we present some comprehensive security architectures aiming to secure all aspects of the BGP protocol. Finally section 8.4.3 discusses efficient solutions addressing specific vulnerabilities of the BGP and some detection mechanisms as proposed by [57].

8.4.1 Current BGP Security Techniques

8.4.1.1 Route filtering

One of the most widely adopted defensive mechanism is the filtering of all ingress and outgress routes based on organizational policies. Most routers block advertisements for addresses known to be private or special, like the loopback address. Filtering also occurs for ASes using private ASNs. The CIDR report publishes daily a list of *bogons*, advertisements of address blocks and AS numbers not yet allocated by IANA. Filtering may also apply based on the length of the AS-path, extremely long paths may be rejected, to defend against contradictory advertisements and route flapping attacks. Moreover, routers filter small prefixes, eg /24, both to avoid increasing the size of routing tables and defend against de-aggregation attacks.

Route filtering is best performed close to the origin of the advertisement. ASes can easily filter traffic from *stub* ASes. Stub ASes are only connected to one AS which connects them to the rest of the Internet. For instance, in Figure 8.2 AS 2 and AS 7 are stub ASes. AS 5 can easily perform filtering for the advertisements of those two ASes, since it has complete knowledge of their topology. It is impossible for these ASes to advertise paths to some origin other than themselves. Additionally, ASes can filter routes from their internal networks. Since AS knows which IP prefixes have been assigned to it and can filter any network announcing unassigned addresses, such action would indicate a compromised router inside the AS or a misconfiguration error. The ability to perform effective filtering decreases as you move away from the origin of the announcement. As there is less information about the topology of the originating AS and because of route aggregation from the intermediate routers. The difficulty also increases from the fact that the IP assignments can change dynamically just as the routes. This dynamic environment makes the maintenance of up-to-date and strict filters for all the participants of the system a dawning task.

The strategy of careful route filtering greatly increase the security of the AS and its neighbors. Currently, it is one of the most commonly deployed security mechanism. However, the technique is limited by the effectiveness of heuristics rules. If not widely deployed it is difficult for a border router to be protected since attacks originating away from the router cannot be detected [60, 255]

8.4.1.2 Routing registries

As discussed earlier route filtering is ineffective for attacks originating several hops away. This is inherently difficult since each router has only a partial view of the Internet topology. Having a consistent global view shared by all the routers could solve this problem. This solution uses a registry service where each AS will publish information about its topology, its assigned

prefixes and its policies. Other ASes would query that service to validate the information they are receiving from the BGP messages. Additionally, ASes could create effective filters based on the information from the registry. An AS would filter all announcements originating from an AS except the ones about its assigned prefixes. However, this approach raises a few privacy concerns. There is no guaranty that the information in the registry will be valid. Malicious ASes could register address prefixes as easily as valid ones. With over 40000 ASes currently active it would be infeasible to check each one's registration for validity. Even the use of the official records of IANA cannot help in this task. Most of the assignments are so old that corporations have bankrupt or change their name or delegate their prefix to other corporations, making the identification of the legitimate owner of a prefix extremely difficult. Another concern has to do with the trust of the registry. A malicious registry can manipulate the route information at will with dramatic consequences for the routing system. Also a poorly secured registry can be hacked giving the attacker a wealth of sensitive information, making attacks on the BGP far easier. Lastly, organization usually consider the information about their topology and their policies as sensitive and are reluctant to share them, especially with competitors [312].

To eliminate the risk of malicious users polluting the routing registry with false information a Public Key Infrastructure(PKI) along with the use of certificates must be used. Each valid owner of a prefix has its certificate signed by the authority, upstream AS or registry, which delegate this prefix to him. Each AS who wishes to authenticate a route would follow the chain of signed certificates (chain-of-trust) from IANA to the valid owner of the IP prefix.

8.4.2 Routing Architectures

This section discusses full scale architectures that provide origin and topology authentication. This solutions provide a comprehensive model for securing BGP against a wide range of attacks

8.4.2.1 s-BGP

s-BGP is the first comprehensive security solution for the BGP [188]. It consists of a *Public Key Infrastructure* (PKI) and an extra attribute in the advertisement messages. s-BGP propose the use of a PKI for pairing prefixes to AS numbers and network elements, routers and hosts, to their respective AS [303]. Each certificate is signed by the authority, upstream AS or regional registry, which delegate the prefix to the AS. At the top of the chain is the IANA, which self signs its certificate. Following the chain of certificates an AS can verify the validity of an advertisement. This technique requires the exchange of public keys which is managed by the PKI.

s-BGP is designed around the notion of *attestations*, more specifically *address and route attestations*. An address attestation is a claim of being the rightful originator of a prefix. Address attestations are distributed *out-of-band*, using a communication channel different of the BGP. Address attestations are issued and signed by the owner of a prefix. To validate as address attestation one need to follow the chain of delegation from IANA to the advertising AS. On the other hand, route attestations are distributed as attributes in the BGP advertisements. The originating AS includes its ASN in a message and then signs it and propagates it to its neighbors. Each neighbor after validating that the originating AS is the rightful owner of the prefix, using the address attestation, it includes its own AS number and signs again the message before propagating it again. Since the message is signed by all passing ASes, using an onion style signing where its signature is applied over the whole message, it is impossible for a malicious AS to tamper in any way the route without breaking the signature chain.

A major drawback of the s-BGP is the processing cost. The extra processing overhead is incurred by the requirement to use cryptographic signatures for all operations. Another drawback is the high convergence time, since for all routes each hop needs to be authenticated. Studies have shown that the convergence time almost doubles [253] and considerable storage is needed.

8.4.2.2 soBGP

Secure origin BGP was proposed as a lightweight alternative to s-BGP by Cisco engineers. It tries to balance between security and performance. It allows the administrator to make choices about the level of security wishes to implement and the performance penalty he is willing to accept. Like the s-BGP it uses a PKI for managing certificates. Three types of certificates are used. The first one binds public keys to all soBGP-speaking routers. The second one contains details about the topology of each router, which are used to create a topology database. The third type of certificate binds prefixes to organizations. All communication is performed as part of the BGP protocol using a new type of message in contrast to the s-BGP.

The soBGP protocol only validates static parts of the BGP protocol, like the routers and the topology graph, not transient, like the paths. So a soBGP-speaking router can only validate that a path is plausible given the global topology created by the topology certificates. A malicious user can tamper paths as long as the tampered path is still valid. Also when the topology graph changes the view of the topology will stay out-of-date the AS that made the change issue a new topology certificate.

soBGP gives the administrator the ability to change configuration to tip the balance between security and performance [252]. An administrator can choose to validate a route before or after start using it. Can decide to validate all the routers though the path or only the originator or none. More-

over, since soBGP only validates static elements all certificates can be issued before the router entering the system and not on-the-fly.

8.4.2.3 IRV

The Interdomain Router Validation (IRV) [140] provides the most distributed of the three approaches discussed in this section. It is based on the idea that each route can be validated by querying the originating router. Each router has an IRV server installed, upon receiving an advertisement the router asks from the local server to validate it. The IRV server communicates with the IRV server in the originating router to confirm the route. The type of validation depends on the router. The IRV server can choose to validate only the origin or the whole path depending on each policy, making the protocol more performance-aware. Additionally, path from trusted ASes may not be validated at all.

Secure connection between the IRV servers can implemented, use of IPsec and TLS, to ensure authentication, integrity and confidentiality. IRV servers can respond to requests depending on their policy making the architecture more privacy-aware than s-BGP and so-BGP since each AS can choose the information it is willing to reveal to identify itself.

A major drawback of this technique is that it needs a functioning network. This can be a problem in the initialization phase or when the network is recovering from an outage. A way to mitigate the problem is the use of static routes for some or all of the IRV server, AS collaboration, exchange paths using goship-like protocols [36], and optimistic routing which involves using the route even before it is validated.

8.4.3 Other Solutions

This section is about solutions that address some, usually not all, security vulnerabilities and concerns of the BGP. On the same time they provide better performance then the methods discussed on the previous section.

Some efforts have been proposed that guard only against prefix hijacking. These methods, called *Origin Authentication* are designed to validate only address ownership.

Aiello et al. [21] formalize the semantics of address delegation and explore different cryptographic methods for binding address prefixes to ASes. Hu et al. in [166] propose the *Secure Path Vector* (SPV) protocol. The SVP uses one-time cryptographic keys to authenticate the origin and the distance from the origin. The originating AS produces off-line enough keys to support even the longest path in the system. For each advertisement it appends a MAC computed using the first key. And sends the message along with all the keys to each neighbor. The neighbors validate the message using the first key and then compute a new MAC using the second key before forwarding

the message to their neighbors. The process is repeated recursively in each router. Since all signatures are pre-computed, there is only a small computational overhead in each operation. The drawback is that SPV protocol is quite complicate since a lot of cryptographic data needs to be exchanged between the routers. In practice it has been shown that the protocol is vulnerable to forgery [278] . Zhao et al. [361] propose some *aggregated path authentication* techniques. These techniques combine the signature amortization and aggregate signatures methods to reduce the computations cost of s-BGP

The *pretty secure BGP* (psBGP) [196] is a protocol which uses a central infrastructure for authenticating AS numbers and a decentralized algorithm for validating prefix ownership. The authors of the papers argue that the population of ASes remain fairly stable over time making ideal to keep track of it in a centralized fashion, reason for which they use a PKI to keep track of the ASes. The prefix ownership on the other hand changes rapidly and additionally ASes have no incentive to announce their delegation decision, making it impossible to keep track of it through a centralized system. In their approach each AS holds a list with the prefix that each neighbor advertises. When an AS A wants to validate a route originating from AS B it asks the neighbors of B whether B has advertise this route to them also. Then it can use an algorithm for example *n out of m* to decide whether to trust this advertisement or not.

Another area of research is the detection rather than the prevention of BGP attacks in the wild. Some systems have been proposed which try to identify prefix hijacking attacks while happening. Most prefix hijacking attacks trigger a *Multiple Origin AS* (MOAS) anomaly. A MOAS anomaly can be benign for instance when a multi-homed AS changes between preferred route, a simple misconfiguration error or an indication of a prefix hijacking attack. Zhao et al. [363] propose the use of the community attribute of the BGP protocol to distinguish between valid and invalid MOAS. Each announcement will contain in the community attribute the ASNs that are eligible to also announce this prefix. Other ASes can perform route filtering based on this attribute. One drawback of this method is that announcements can be tampered before reaching their neighbors. If tampering introduces a new ASN in the announcement then the defence against prefix hijacking has been rendered useless. On the other hand, if the announcement removes some of the valid ASNs then neighboring ASes will reject valid routes.

Kruegel et al. [198] propose using an intrusion detection system which is not based on BGP attributes to identify anomalies, but rather it identifies departure from normal behavior. For instance, MOAS conflicts that have not been seen in the past are considered malicious or weird route aggregations. Generally the metrics they proposed are based on the study of common BGP configurations.

A system named *Prefix Hijacking Alert System* (PHAS) [200] provides a mechanism for anomaly detection but leaves the final decision of whether the anomaly consists an attack to the user. The PHAS provide a central server where the owner of a prefix registers himself and the prefix he owns. PHAS uses Route Views and RIPE for MOAS on the given prefix. Route views and RIPE provide real time access to the announcements that happen on the global routing system. The PHAS server check for MOAS conflicts for the prefixes that are registered and notify the owner of the prefix for a possible attack. This centralized approach makes the server a single point of failure, also there are no guaranties that the legitimate owner of a prefix will make the registration and not a malicious attacker.

Pretty Good BGP (PSBGP) [187] make the assumptions that new ASes and new route announcements are inherently suspicious. That router that receives an announcement checks whether it is a new announcements or one that has been previously seen. Newly advertised routes are not preferred for a period of time as long as there are alternative routes. Providing an effective defense against short-lived prefix hijacking attacks. The system can be extended to defend against longer-lived attacks by asking for human confirmation before accepting suspicious routes. PSBGP in vulnerable in the link-cutting attacks [43] discussed previously which can leave no other route than the malicious one. PSBGP cannot prevent these attacks since they exhibit the same behavior as multihomed ASes that change their preferred route.

The system proposed in [165] defends against prefix hijacking by identifying the network that advertises the request. In this paper, information about the network that advertises a prefix is gathered using various probes. For such a network the hosts are probed to gather information like Roundtrip-time, OS used and the id field of the TCP packets. These information is used to create a unique fingerprint of the network. When a MOAS conflict occur both destinations are probed. If there is differenciation between the known fingerprint and the network that answers the probes than probably a prefix hijacking attack as occured.

The *Whisper* protocol [318] is another mechanism to detect prefix hijacking attacks that is designed to operate without the need of a PKI. Whisper only raises alarms and does not pinpoint the source of the problem. It operates by generating for each prefix a random (secret) value at its originator. This value is hashed and propagated with the prefix advertisement. On each subsequent propagation of the route advertisement the propagated hashed value is re-hashed. This allows the receiver AS of two different advertisements for the same prefix to identify if the advertised routes are *consistent to each other* and raise an alarm if they aren't. The exact verification method is dependent on the hash function used.

Name	Topologibal Auth.	Path Auth.	Origin Auth
Route filtering	weak	weak	weak
Routing registries	weak	weak	weak
s-BGP	strong	strong	strong
soBGP	strong	none	strong
IRV	strong	strong	strong
Origin Authentication	none	none	strong
SPV	strong	strong	none
Signature Amortization	strong	strong	none
Reference Locality	strong	strong	none
psBGP	weak	strong	weak
MOAS Detection	none	none	weak
Intrusion Detection	none	none	weak
PHAS	none	none	weak
PSBGP	weak	weak	weak
Real-Time Monitoring	none	none	weak
Whisper	none	weak	none

Figure 8.3: Comparative array for all the security solutions

8.5 DNS Overview

The *Domain Name System* (DNS) is one of the critical components of the Internet. Its main function is to translate human-readable domain names to machine-readable IP address. Even though it is not essential for the functionality of the Internet it is considered critical since many applications like e-mail, the web, instant messaging services cannot operate without it.

The need for translating names to numerical addresses existed since the beginning of the ARPANET. At first it was only a file named ”HOSTS.TXT” that contained the mappings between names and numerical addresses. As the Internet was expanding keeping this file updated in every host became unsustainable. In 1983, Paul Mockapetris invented the Domain Name System as a scalable solution to the naming to IP address problem [233]. It was a major step forward to its previous system in terms of efficiency and availability but it does not provide any resilience against malicious attacks.

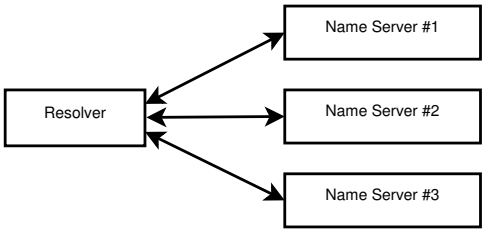
DNS consists of three components. First, the *domain name space* and the *resource records* which create a tree structure with names and associated data. Second, *name servers* which store the domain name space name tree in a distributed fashion, each name server holds only a part of the tree and pointer to other name servers. A name server can also cache other parts of the tree for performance reasons, mirrors of the name servers can also exist for redundancy. The name server holding the root of the name space tree is called the *root name server*. The root name server holds information about

the name server of the next level which are the *top level domain server*. These are the servers authorized to serve the top level domains like .com, .edu, .org, .uk, .it, .gr etc. Lower in the tree are the organization and corporation name servers. Finally, the *resolvers* are programs for extracting information from the DNS infrastructure. The resolver receives queries from humans or other programs, (mail clients, web browsers) for domain names they wish to translate and communicates with the name server to fetch the answer. Each resolver must me able to access at least one name server and traverse between them to find the answer for a query.

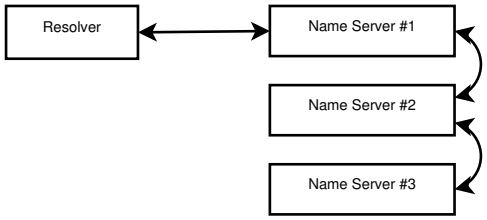
The process of translating a domain name to an IP address is referred to as *resolving*. Since the domain space is hierarchical and decentralized the resolution process involves querying a name server near the top of the tree and then querying other name servers while traversing the tree towards the name server with the necessary information. There are two methods for resolving a domain name. The first one is the *iterative* method, shown in Figure 8.4a. In this method the DNS resolver asks the first name server, which is usually the default name server of the ISP, if the server is authoritative or it has cached the answer it returns it, otherwise it return the address of another name server better qualified to answer. In the second method named *recursive resolution* a DNS name server takes the responsibility to obtain the answer from the other name servers and return it to the resolver.

For example, let's assume that the web browser wants to translate the domain name www.example.com. In the first step of the resolving process the application sends a query to the local DNS resolver with the domain name it wants to translate. The resolver knows the IP address of at least one name server, usually the default DNS server provided by the ISP. The resolver queries the name server who will either return the answer or the address of another name server, usually one of the root name server. The resolver queries the root server to learn the name server authoritative for "com" Top Level Domain. Then resolver asks the "com" name server who is the authoritative name server for the "example" name space. This would probably be a name server controlled by an ISP or a corporation. The resolver will ask that name server the IP address of the host serving the "www" protocol. Conceptually, the resolver traverses down a path of the name space tree until it finds a leaf which will have the IP address of the host the resolver looks for. The second method is the *recursive method* shown in Figure 8.4b. In this mode a DNS name server, usually the one of the ISP, undertakes the task of querying the other name servers to obtain the answer and return it to the resolver. The second method may seem more resource intensive from the side of the name server but it allows for better caching. In case of a second user asking the same domain name resolution the name server can serve him from its cache.

Since, multiple DNS servers can be authoritative for some name space the DNS protocol employs a *zone transfer* mechanism for synchronization



(a) Iterative Lookup



(b) Recursive Lookup

between these servers. Usually, the secondary name server (slave) queries the primary one (master) periodically in order to stay up-to-date. The response to that query includes all the information associated with name resolution like name servers, host names, authority records and Time-To-Live information. Regular hosts can also request a zone transfer if they want to collect information about an entire domain. This can be a security risk since network administrators tend to name hosts according to the service they provide so the DNS records can give an abundant of information about the network topology and the services of each host. Another attack vector is that the query is just a few bytes in size whereas the response is much bigger.

8.6 DNS Attacks

8.6.1 Denial of Service

A *Denial of Service attack* (DoS) is an attempt to render a computer resource unavailable to the legitimate users. DNS can be the target but also the tool for such attacks. A malevolent wishing to render the Naming System unavailable would target the name servers. One kind of attack would focus on crashing the servers. Specially crafted packets can exploit software vulnerabilities, like buffer overflows, which can cause the server to crash. It is not necessary that the attacker targets the DNS software, all services running in the same host as the DNS server, even the OS itself can be the target of the attack. Another kind of DoS attack in the name servers aims to consume all the available resources of the server, thus making the server response slowly or even unresponsive to legitimate requests [174, 171]. DNS name

servers are particularly vulnerable in this attack because they're intended to be publicly accessible making host filtering impossible, even traffic-shaping can have a prohibitive impact to legitimate users. An attacker can send request for domain guaranteed not to be cached, non-existent domains are the simplest choice since it is improbable to have been requested in the past. Such queries could only by served by the authoritative server of the domain forcing the DNS server under attack to make at least one look up to another server. A coordinated attack (DDoS) where all of the attackers request domain names that are not cached can impose an unbearable burden to the server who has to recursively look up those domains.

DNS can also be used to perform a type of DoS called *amplification attack* to other hosts. An attacker can take advantage of the huge difference in sizes between a zone transfer request and the response. Assuming that the DNS server accepts requests for zone transfers from everyone an attacker can send such request using the source address of the target. The DNS name server will respond to the request flooding the unsuspected target with data. If the target is a network rather than a single host than the attacker can ask for zone transfers spoofing multiple IPs on the target network.

8.6.2 Man-In-The-Middle Attack

DNS queries are usually transmitted using the UDP protocol for low overhead. Most of the times both the query and the response fit in a single datagram which travels through the network unsigned and unencrypted. Since the DNS protocol does not authenticate the origin of the requests, an attacker can intercept a query as it travel from the resolver to the DNS name server and respond with a malicious translation before the real name server. The odds usually favor the attackers because he frequently resides in the same LAN as the victim whereas the DNS server usually not. Moreover, the DNS server receiving the query may have to recursively look it up which would increase the response time even further. The resolver requesting the translation will accept as valid any response it receives for two reasons. First, it cannot verify that the responder is who he claims he is and second it cannot verify that the responder is delegated to control that part of the domain space. This type of attack, called *Man In The Middle* can be used to redirect benign user to sites different from what they requested for phishing and other malicious activities. It is also possible that attacker stands between two name servers and in this case it can cause a *Cache Poisoning* which we discuss in 8.6.4.

8.6.3 Query prediction

Since the DNS uses a connection-less protocol of communication questions are associated with answers based on a Transaction Id. This means that

the answer to a query would have the same Transaction Id, which is a 16-bit integer, and it would also include the query. For an attacker to exploit this restriction he need to guess both the Transaction Id and the query. For the first there are 2^{16} combinations, and with today's computers this can be easily brute forced. But this may not even be necessary, there are known implementations in the DNS resolver and servers that can make the Transaction Id predictable. Until recently some of the most widely used server implementations were simply incrementing the value of the Id. The query is harder to predict but in practice if the state of the resolver is known then it's easy to predict as well. Knowing these two the attacker can send a forged response to the resolver. This attack has advantages and disadvantages compared to the Man In The Middle attack 8.6.2. The main disadvantage is that the ID and query may not be easy to predict, on the other hand the attacker does not have to eavesdrop the information as it travels. If the attacker successfully predict the query can redirect the benign user to a site of his choice. Additionally, if the attacker performs this attack targeting a name server who recursively looks up queries, then it can serve the legitimate translation but use a property of the DNS protocol that allows the response to contain other translation not directly related to the query. These other data are treated as trusted and cached by the name server.

8.6.4 Cache poisoning

Cache Poisoning is a more sophisticated attack aiming to cache in a legitimate server resolutions that are not valid. There are two variations of the attack as we will discuss. In both of them the attacker owns a domain name, for instance www.malicous.com and also controls the name server that provides the resolution for that domain, for example ns.malicous.com. The attacker issues a request for this domain name to a benign name server, *foo.com*. Assuming that *foo.com* works recursively, it will have to query the malicious DNS server to resolve the domain name. In the first approach the malicious attacker take advantage of a DNS property that allows a response packet to contain information not directly related to that query. The attacker can exploit that to include information about translations about other domain that he does not control. The benign server will cache that information so subsequent queries will return the malicious resolution. In the other approach, the attacker, since it control the malicious server, learns the Transaction Id. Then, he makes queries about legitimate sites while at the same time the malicious server sends to the benign server responses to that query using Transaction Ids chosen based on the Transaction Id it has just received. It is essentially tries to perform the Query Prediction discussed in Section 8.6.3, only this time it uses the knowledge it has previously obtained. Most implementations of the name servers are vulnerable in choosing their Transaction Id based on a predictable manner. This attack

has high rate of success since the query is known and the Transaction Id can be guessed based on previous information. Once the benign server accepts the response of the malicious one as valid it caches it and servers all client of that domain using the rogue translation.

8.6.5 Betrayal By Trusted Server

Between the resolver and the authoritative DNS server reside many DNS servers. The authenticity of the response depends on each hop of the process being trustworthy. If one of the hops gets compromised the system becomes untrustworthy. As all systems on the Internet a name server is vulnerable in all kinds of attacks as all the other servers on the Internet.

8.6.6 Social engineering

The evil user can try to exploit the characteristics of users and the relations among them to control the naming system. These attacks do not target the protocol but can cause great frustration and economic loss to the victim. *Domain Hijacking* is the transfer of ownership of a domain name from the rightful owner to a malicious one. The attacker tries to convince the domain registrar to modify the registration information of the domain name or transfer it to another registrar. To do this the attacker uses all sort of social engineering techniques to impersonate the owner of the domain. If the attack succeeds the attacker usually initiates a transfer to another registrar to a different country to make the return to the legit owner even more difficult. This kind of attack can have huge financial impact to the owner since, any income generated by the site in his domain is directed to the attacker.

A similar attack named, *typosquatting*, relies on the typographical mistakes of the users. The attacker uses domain names similar to the legitimate ones with some small change for instance instead of example.com the attacker owns the examples.com, the malicious domain is designed to be confusingly similar to the legitimate one. Subsequently, a user who accidental enters the malicious site does not notice the difference and can give away personal information.

The *Internation Domain Name* abuse takes a advantage of the international domain names that have been recently used in the Internet. It is possible to register a domain name using all the Unicode character set. An attacker can imitate the legitimate page by using a letter that look similar in different languages. Again the attacker aims to trick the user into believing that he is in the legitimate page, in order to steal personal information.

8.7 DNS Defenses

DNS was designed to be scalable but not secure. This stems from the fact that when DNS was designed Internet consisted of only a few thousands hosts located mainly at universities and government agencies. Security was not a major consideration. Since then emergence of threats has led the Internet Engineering Task Force (IETF) to propose a series of specifications to secure DNS against malicious attacks, this suite of specifications is called Domain Name System Security Extensions (DNSSEC).

8.7.1 DNSSEC

DNSSEC [172] is designed to protect DNS resolvers from fraudulent responses. When a resolver receives a reply he must be able to identify that the answer originated from the authoritative DNS server of the domain and that the data was not tampered in the way. In DNSSEC all authoritative servers have a pair of public/private key for signing their response to the resolvers. The resolver use the public key to authenticate that the response was not tampered by a third party. To validate that the name server is authoritative for a domain there exists a chain-of-trust. DNSSEC uses the hierarchy of the naming tree, more specifically each time a name server delegate a part of its naming space it signs a certificate to the new authoritative name server to this name space. For the resolver to validate the authority of a name server needs at least a key that is correct from sources other than the DNS. This starting point is called *Trust Anchor*, usually it is a key of one of the root server. Using this key the resolver can follow the authentication chain up to the authoritative server responsible for a domain name.

Despite the improvements in security, DNSSEC hasn't yet been widely deployed on the Internet. Some aspects of the protocol are still under debate, for instance how to manage the keys, how to distribute the keys or what happens if a key gets compromised. There are also performance concerns: the size of the DNS packets would increase significantly if DNSSEC was to be deployed because of the size of the signature which is much bigger than the actual data in the packet. Moreover the incurred overhead for signing all the responses could be prohibitive for busy DNS servers.

8.7.2 Defensive Architectures

Another way to increase security, without deploying new technologies, is to partition the resources and limit access to them. One technique referred as *Split-Horizon DNS* recommends the use of one public and one private DNS. This approach tries to limit the exposure of the organizational structure to the outliers. The outside world has access only to the public DNS which holds information only about hosts that should be publicly available like

web and mail servers. The private DNS server, which is accessible only from inside the company holds information about private resources like file servers and NFS clients. This design limits the exposure of our network structure to the outside world.

A similar design splits the DNS server according to their functionality. This strategy named *Split-Split* uses two name servers, the Advertising Name Server and the Resolving Name Server. The first one is responsible for answering queries of external entities, resolvers and other name servers, for our domain. Since it is the authoritative server for our domain it must be accessible from both inside and outside the organization. But, since it only answers queries for our domain and does not try to resolve queries for other domains, there is no need to have cache hence eliminating DNS poisoning attacks. We can eliminate the exposure of the advertising name server even further by disabling recursive resolution, there is no incentive for having this feature which can be an attack vector since the hosts in our organization will use the second server for resolution.

The Resolving Name Server is responsible for resolving the requests of the internal hosts. This approach has a number of benefits both from performance and security perspectives. Since, the server is transparent to the outside world it is very hard for an outsider to perform DDOS and poisoning attacks. Moreover, since the server is dedicated for internal use the performance requirements are smaller, and also the employees enjoy better performance and reliability.

8.7.3 IP filtering and Signatures

DNS records are one of the most valuable source of information for the attackers of an organization. Zone transfers are one of the ways that the attackers can obtain that information. Zone transfers are used primarily for mirroring the information about an authoritative zone for reliability and security between the ”master” and the ”slave” nameserver. But, anyone can initiate this process with the server. To mitigate the problem the master nameserver can use IP filtering, this involves not accepting zone transfer requests except from specific IP addresses, *Access Lists*. This technique has been used extensively in the Internet but again is vulnerable to IP spoofing attacks. The RCF 2930 proposes the use of cryptographic signatures for authenticating the identity of the sender and the integrity of the message. The technique named *Transaction Signature* (TSIG) assumes that the master and the slave server share a secret key. When the slave wants to send a request for zone transfer it hashes[1] the message and the secret key to obtain an HMAC signature for the request. The master server that receives the request and the signature is then able to verify that the request originates

[1] Currently the only supported hash function is MD5

from the slave server and hasn't been tampered with. This is done by regenerating the HMAC signature and checking that it matches the one it received with the request. To secure the slave from cache poisoning the server also signs its response.

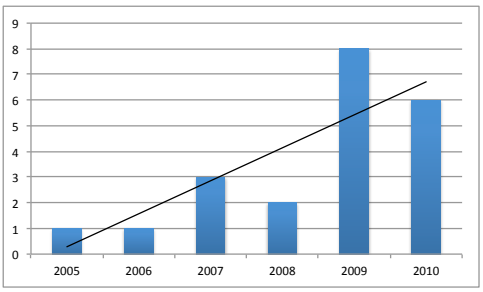
8.8 Conclusion

Our presentation in this section focused on attacks and defenses on two very important services for the operation of the Internet, the Border Gateway Protocol and the Domain Name System. Specifically, we started by introducing how each service operates, to give the reader suffecient information to understand, first, the importance of each service, and second, how the service can be attacked. We the expanded on actual attacks that have been, or can be, carried against these services, as well as a comprehensive discussion on possible defense. It is our belief that these core network services will remain under threat in the near future, due to their importance in the operation of the Internet.

9 Virtualization and Cloud Computing Attacks

9.1 Introduction

In this chapter, we discuss attacks and threats against two upcoming technologies, namely virtualization and cloud computing.



We first present Virtualized environments in Section 9.2, as they are the enabling, underlying technology for most cloud computing environments. In particular, we divide the threats presented in attacks against the virtualization systems themselves (Section 9.2.1), and abuses of virtualization, in particular to enable the creation of resilient malware strains (Section 9.2.2).

9.2 Virtualization environments

Virtualization technology has been widely accepted and integrated fastly in the enterprise ecosystem: Gartner estimates that, in spite of licensing and deployment issues, by the end of 2010, 80% of enterprises had a virtualization program, plan or project ongoing, and 25% of current server workloads were running on VMs[1]. Perceived economic advantages have been the driver, and consequently implementation has cared more about optimization and, in the best cases, resiliency than about security. However, as pointed out in [27], virtualization technology falls short of delivering on the ideal of a robust, trustworthy virtual computer system that is “exactly alike” or even better than bare metal hardware.

In the following sections we will outline both how virtualized environments can fail under certain attacks (Section 9.2.1), and how the concepts of virtualization can be abused to create extremely resilient malware (Section 9.2.2).

9.2.1 Attacks against virtualized environments

Attacks against virtualized environments are even older than the widespread adoption of the technology itself. In [145] the authors observe that, at the time of writing (2003), virtualization technologies (at application-level, in their example) exhibited weaknesses that could be exploited. In particular, they exploited memory errors in order to take control of a running JVM environment.

More recently, several authors and independent researchers in the hacker community assessed the security of existing virtualization technologies. In [260] the author assesses Bochs, QEMU, VMware, Xen on the x86 platform via both source code auditing and blackbox testing. Almost all the tested versions fell under attacks. In [127] more attacks are described and several practical protection measures are proposed.

This has led to an enormous growth in vulnerabilities. In fact, vulnerabilities in VMware have grown 35 times between 1999 and 2007 [201].

One of the few scientific papers outlining attacks against virtualization environments is [289], where the authors show that low-entropy randomization algorithms of RNGs, which fail to reuse information found in VMs' snapshots, can be exploited using the so-called "reset vulns" to compromise encrypted sessions, or even to expose a server's DSA signing key. A mitigation strategy is proposed for hedging routine cryptographic operations against bad randomness.

9.2.2 Abuses of virtualization

Researchers have also shown that virtualization technology can actually be used as a tool for creating malware. In [191] a particularly nasty malware is designed, which installs a virtual-machine monitor underneath an existing operating system and hoists the original operating system into a virtual machine.

Further research has been done in the applied security/hacking community, producing for instance the Blue Pill prototype [293].

9.3 The emerging paradigm of cloud computing

The emerging concept of cloud computing is both widespread and puzzling [337]. According to the definition given in a recent, comprehensive

and sound analysis [28], cloud computing enables organizations to run applications (often referred to as services) on a pay-as-you-go basis on top of reliable, highly-available, scalable software and hardware infrastructures referred to as *clouds*. In some sense, a cloud can be seen as a modern, large mainframe [224] with virtually infinite resources, and the term cloud *computing* refers to the use of such resources to deliver web services.

Cloud security is also, unsurprisingly, an ill-defined and vague field. Community-driven initiatives [**?**] and organizations such as the Cloud Security Alliance [228] are pursuing the common objective of gathering knowledge and joining efforts to devise security measures appropriate for cloud computing. The rapid growth of Cloud Security Alliance [138] suggests that the community is indeed concerned about security. While cloud computing certainly poses new challenges, however, it may be difficult to distinguish between issues *specifically* caused by the emerging computing paradigm, and issues that, by coincidence, occur on a system deployed on a cloud [70]. On one hand, the community of cloud users is worried about the fragility of cloud computing [54, 155, 298, 90, 291, 185] and concerns have arisen recently regarding the significant outages of the major cloud providers. On the other hand, it is quite easy to observe that security issues are mainly caused, as usual, by programming errors. In other words, although a cloud unquestionably offers a sophisticated and flexible platform, it still runs pieces of software, which can be just as insecure as any piece of software running on traditional environments. For instance, Amazon's S3 experienced two outages in the 2008 due to an overload of the authentication service [315] and an error in a single bit [325]. Also, Google AppEngine suffered a "blackout" because of a programming error [347]. Clearly, such vulnerabilities have nothing to do with cloud computing itself.

9.3.1 Cloud computing: challenges and attacks

As discussed thoroughly in [28], the cloud computing paradigm presents some novel challenges to computer scientists.

Cloud providers try to sell "cloud security" as just another instance of the problem of virtualization security, offering security-as-a-service introspection monitoring solutions. However, in [72] it is shown that none of them monitors guests on the semantic level required to effectively support both white- and black-listing of kernel functions, or allows to start monitoring VMs without the assumptions of a secure start state.

Actually, cloud security challenges go beyond the challenges of virtualized environments in at least three ways.

9.3.1.1 Data confidentiality

The obvious and, in general, effective measure to protect data confidentiality is encryption. However, encryption is not always a feasible solution, especially for data-intensive applications that require high I/O throughput. Although a scheme to compute arbitrary circuits over encrypted data (so to avoid encryption/decryption) has been proposed recently [137], in its current stage this solution requires significant efforts to be adopted in the real world. In addition, encryption is not straightforward when data is distributed. As most of the users refrain from encrypting their laptop harddrive because of the technical and computational overhead, in a similar vein, would users bother encrypting their virtual, remote storage? Moreover, if a remote storage is transparently encrypted (i.e., by the provider), whom the data belongs to? The user, the provider? And, is this fact provable? How?

9.3.1.2 Resource sharing

The security issues typical of shared hosting environments are magnified in the case of clouds, because the additional, *unperceived* complexity due to dynamic resource slicing, allocation, replication and optimization, gives each user the illusion of being unique. Users may behave maliciously (e.g., by violating the underlying virtualization systems as outlined in 9.2.1), affecting other users and their reputation. A recent incident [197] that affected the reputation of a whole, shared Amazon EC2 cloud is discussed in [70] as a noteworthy example of this specific issue. To what extent users sharing the same cloud are isolated? Is it feasible to employ simple fail-over mechanisms to transparently "move" a mis-behaving user or process onto another cloud? Would this offer an adequate degree of protection?

In [288] the authors use the Amazon EC2 service as a case study, and show that it is possible to map the internal cloud infrastructure, identify where a particular target VM is likely to reside, and then instantiate new VMs until one is placed co-resident with the target. They proceed to show how then this can be used to mount cross-VM side-channel attacks.

9.3.1.3 Debugging and auditability

Programmers know how to pinpoint and solve software flaws using debuggers, which allow to track the execution of complex, multi-threaded processes and inspect the memory content. This routine task turns out to be a challenging research problem in the case of distributed applications [89, 136, 283]. Besides the intrinsic difficulties that programmers have to face, i.e., understanding what is "the memory", or the "process state", debugging tools devised for large-scale distributed systems are quite obtrusive (e.g., they require code annotation). In addition, bugs are difficult to re-

produce in local, smaller configurations because testing and development environments might differ significantly from deployment conditions.

From a purely forensic point of view, monitoring and keeping track of a system's activity is as important as debugging. Unfortunately, this might in turn be very difficult in large-scale systems, since data and processes are distributed rather than contained within well-defined boundaries. Even simple tasks such as collecting logs are inherently more challenging when applications are distributed.

9.3.2 The Cloud as a new paradigm for malware

The wave of crime that we see on the Web today is quite different from the more traditional network attacks. Our adversaries changed tactics moving away from noisy scanning to more stealthy attacks, because their motivations changed. The modern cyber attackers look for economic incentives, as opposed to exhibitions of technical superiority [274]. The fast-growing underground economy has already embraced the cloud model.

In fact, it can be argued that botnets are an embryonic cloud-like, distributed malicious infrastructures [64, 274]. Some researchers [26] even quantified the threat of browsers being controlled by malicious websites that instruct them to remotely attack third parties (e.g., denial of service attacks, worm propagation and reconnaissance scans), creating a powerful, cloud-like, client attack infrastructure.

10 Conclusions

In this document we presented a survey of cyberattacks as they related to System Security. We categorized cyberattacks in eight major categories and discussed the major issues, research directions, attacks and defenses in each category. The categories in this document do not cover non-systems security classes such as trust, crypto, measurement, etc., and also categories such as malware or sensor networks that will be covered in Deliverables 5.1 and 6.1 of the *SysSec* Project. Finally, we conducted an analysis looking at the major publications of the area and measuring the trends of the research community in terms of papers published in the above categories.

Bibliography

[1] Virtualization adoption thwarted by software licensing issues, report. available online.

[2] Homeland Security: Networking Security and Policy. In *Proceedings of 14th USENIX Security Symposium*, invited talk, 2005.

[3] Modeling and Preventing Phishing Attacks. http://www.informatics.indiana.edu/markus/papers/phishing_jakobsson.pdf, 2005.

[4] Communication from the commission on a european programme for critical infrastructure protection. In *Commission of the European Communities*, Brussels, December 2006.

[5] Crutial project. 2006. Available at http://crutial.rse-web.it.

[6] Integrated Risk Reduction of Information-based Infrastructure Systems. In *IRRIIS project, Deliverable D 1.2.1 "Scenario analysis"*, Jun 2006. Available at http://www.irriis.org/File.aspx?lang=2\&oiid=8661\&pid=572.

[7] Spear phishing: Highly targeted phishing scams. http://www.microsoft.com/protect/yourself/phishing/spear.mspx, 2006.

[8] Directive on the identification and designation of european critical infrastructure and the assessment of the need to improve their protection. In *Commission of the European Communities*, Brussels, Belgium, May 2008.

[9] Sophos Facebook ID Probe. http://www.sophos.com/pressoffice/news/articles/2007/08/facebook.html, 2008.

[10] Dcaf horizon 2015. 2010. Available at http://www.dcaf.ch/dcaf/Projects/Publications?lng=en\&id=123098.

[11] Deliverable d3.1: White book: Emerging ict threats. In *FORWARD project. Managing Emerging Threats in ICT Infrastructures*, 2010. Available at http://www.ict-forward.eu/media/publications/forward-whitebook.pdf.

[12] OpenSocial, the Web is better when it's social. http://code.google.com/apis/opensocial, 2010.

[13] Sophos Security Threat 2010. http://www.sophos.com/sophos/docs/eng/papers/sophos-security-threat-report-jan-2010-wpna.pdf, 2010.

[14] Facebook Application Platform. http://developers.facebook.com, 2011.

[15] The Repast Suite. 2011. Available at http://repast.sourceforge.net.

[16] Askarov A. and Sabelfeld A. A.security-typed languages for implementation of cryptographic protocols: A case study. In *10th European Symposium on Research in Computer Security, Lecture Notes in Computer Science*, volume 3679, pages 197–221, 2005. DOI: 10.1007/11555827.

[17] Creery A. and Byres E. Industrial cybersecurity for power system and scada networks. In *IEEE Industry Application Magazine*, July-August 2007.

[18] Ghorbani A., Bagheri E., Zafarani R., Baghi H., Noye G., and Onut I. Agent-based interdependencies modeling and simulation (aims). In *Technical Rep. No. IAS-TR01-06, Intelligent and Adaptive Systems Research Group, Faculty of Computer Science, UNB*, 2006.

[19] Rubin A. Electronic Voting in the United States: An Update. In *14th USENIX Security Symposium, invited talk*, 2005.

[20] Esther Addley and Josh Halliday. Operation payback cripples mastercard site in revenge for wikileaks ban, Dec. 2010 http://www.guardian.co.uk/media/2010/dec/08/operation-payback-mastercard-website-wikileaks.

[21] William Aiello, John Ioannidis, and Patrick McDaniel. Origin authentication in interdomain routing. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 165–178, New York, NY, USA, 2003. ACM.

[22] P. Akritidis, W. Y. Chin, V. T. Lam, S. Sidiroglou, and K. G. Anagnostakis. Proximity breeds danger: emerging threats in metro-area wireless networks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 22:1–22:16, Berkeley, CA, USA, 2007. USENIX Association.

[23] Aleph1. Smashing The Stack For Fun And Profit. http://www.phrack.com/issues.html?issue=49&id=14, November 1996.

[24] Alasdair Allan. Got an iphone or 3g ipad? apple is recording your moves. http://radar.oreilly.com/2011/04/apple-location-tracking.html, April 2011.

[25] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 38–47. ACM, 2003.

[26] Spiros Antonatos, Periklis Akritidis, Vinh The Lam, and Kostas G. Anagnostakis. Puppetnets: Misusing web browsers as a distributed attack infrastructure. *ACM Trans. Inf. Syst. Secur.*, 12:12:1–12:38, December 2008.

[27] I. Arce. Ghost in the Virtual Machine. *Security & Privacy, IEEE*, 5(4):68–71, 2007.

[28] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[29] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge attacks on smartphone touch screens. In *Proceedings of the 4th USENIX conference on Offensive technologies*, WOOT'10, pages 1–7, Berkeley, CA, USA, 2010. USENIX Association.

[30] Ezell B., Farr J., and Wiese I. Infrastructure risk analysis model. In *Journal of Infrastructure Systems*, volume 6 of *3*, pages 114–117, 2000.

[31] Simons B. Computerized Voting Machines: A View from the Trenches. In *Proceedings of 10th European Symposium on Research in Computer Security. Lecture Notes in Computer Science*, volume 3679, pages 1–2, 2005. DOI: 10.1007/11555827.

[32] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik P. A. Lensch, and Martin Welk. Tempest in a teapot: Compromising reflections revisited. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, pages 315–327, Washington, DC, USA, 2009. IEEE Computer Society.

[33] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising reflections-or-how to read lcd monitors around the corner. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 158–169, Washington, DC, USA, 2008. IEEE Computer Society.

[34] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM, 2007.

[35] R. Baeza-Yates, C. Castillo, V. López, and C. Telefónica. PageRank increase under different collusion topologies. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 17–24. Citeseer, 2005.

[36] Brenda Baker, Robert Shostak, Independently R. T. Bumby, A. Hajnal, E. C. Milner, and E. Szemerdi. Gossips and telephones. *Discrete Mathematics*, 2:191–193, 1972.

[37] J. Balasch, A. Rial, C. Troncoso, C. Geuens, B. Preneel, and I. Verbauwhede. PrETP: Privacy-preserving electronic toll pricing. In *USENIX Security*, 2010.

[38] Vijay A. Balasubramaniyan, Aamir Poonawalla, Mustaque Ahamad, Michael T. Hunter, and Patrick Traynor. Pindr0p: using single-ended audio features to determine call provenance. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 109–120, New York, NY, USA, 2010. ACM.

[39] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel. Abusing social networks for automated user profiling. In *Recent Advances in Intrusion Detection*, pages 422–441. Springer, 2011.

[40] David Barrera, H. Güneş Kayacik, Paul C. van Oorschot, and Anil Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 73–84, New York, NY, USA, 2010. ACM.

[41] A. Barth, C. Jackson, and J.C. Mitchell. Robust defenses for cross-site request forgery. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 75–88. ACM, 2008.

[42] Michael Becher and Felix C. Freiling. Towards dynamic malware analysis to increase mobile device security. In *Proc. of SICHERHEIT*, 2008.

[43] S.M. Bellovin and E.R. Gansner. Using link cuts to attack Internet routing. *AT&T Labs Research Technical Report, http://www. research. att. com/smb/papers/reroute. pdf*.

[44] Côme Berbain, Olivier Billet, Jonathan Etrog, and Henri Gilbert. An efficient forward private rfid protocol. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 43–53, New York, NY, USA, 2009. ACM.

[45] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *18th International Conference on World Wide Web (WWW)*, 2009.

[46] Nick Bilton. Congress has questions for sony about attack. http://bits.blogs.nytimes.com/2011/04/29/house-of-representatives-letter-questions-sony-over-attack/?partner=rss&emc=rss, April 2011. New York Times.

[47] Erik-Oliver Blass, Kaoutar Elkhiyaoui, and Refik Molva. Tracker : security and privacy for RFID-based supply chains. Technical Report EURECOM+3087, Institut Eurecom, France, 04 2010.

[48] Hristo Bojinov, Elie Bursztein, and Dan Boneh. Xcs: cross channel scripting and its impact on web applications. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 420–431, New York, NY, USA, 2009. ACM.

[49] J. Bonneau and S. Preibusch. The privacy jungle: On the market for data protection in social networks. *Economics of Information Security and Privacy*, pages 121–167, 2010.

[50] Abhijit Bose and Kang G. Shin. Proactive security for mobile messaging networks. In *Proceedings of the 5th ACM workshop on Wireless security*, WiSe '06, pages 95–104, New York, NY, USA, 2006. ACM.

[51] S. Bosworth and M.E. Kabay. *Computer security handbook, Chapter 11*. John Wiley & Sons, 2002.

[52] S.W. Boyd and A.D. Keromytis. SQLrand: Preventing SQL injection attacks. In *Applied Cryptography and Network Security*, pages 292–302. Springer, 2004.

[53] Brandon Bray. Compiler Security Checks In Depth. http://msdn.microsoft.com/en-us/library/aa290051.aspx, February 2002.

[54] Jon Brodkin. Gartner: Seven cloud-computing security risks. http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853, 2008.

[55] BugTraq. Wu-Ftpd Remote Format String Stack Overwrite Vulnerability. http://www.securityfocus.com/bid/1387/info, June 2000.

[56] Bulba and Kil3r. Bypassing StackGuard and StackShield. *Phrack Magazine*, 56(5), January 2000.

[57] K. Butler, T.R. Farley, P. McDaniel, and J. Rexford. A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122, 2010.

[58] Barton D. C. and Stamber K. L. An agent-based microsimulation of critical infrastructure systems. In *Tech. Rep. SAND2000-0808C*, 2000. Available at http://www.osti.gov/bridge/servlets/purl/753426-Y0Xgs5/webviewable/753426.pdf.

[59] Stöcker C., Neumann C., and Dörting T. Iran's twitter revolution: Ahmadinejad's fear of the Internet. In *Spiegel*, Jun 2009.

[60] M. Caesar and J. Rexford. Bgp routing policies in isp networks. *Network, IEEE*, 19(6):5 – 11, nov.-dec. 2005.

[61] Ch. Callegari, Gazzarrini L., Giordano S., Pagano M., and Pepe T. Detecting network anomalies in backbone networks, recent advances in intrusion detection. In *Lecture Notes in Computer Science*, volume 6307, pages 490–491, 2010. DOI: 10.1007/978-3-642-15512-3_28.

[62] Diego Calleja. Linux 2.6.25. http://kernelnewbies.org/Linux_2_6_25#head-fc71477174376b69ac7c3cd153ed513d25bf2ca May 2008.

[63] Rich Cannings. Exercising our remote application removal feature. http://android-developers.blogspot.com/2010/06/exercising-our-remote-application.html, June 2010. Android Security Leader, Google.

[64] Jeffrey Carr. *Inside Cyber Warfare: Mapping the Cyber Underworld*. O'Reilly Media, Inc., 2009.

[65] Josh Carr. Worm rickrolls unsecured jailbroken iphones via ssh. http://www.tuaw.com/2009/11/07/jailbreak-worm-rickrolls-the-unsecured/, November 2009.

[66] CERT CC. Trends in denial of service attack technology, OCT. 2001 $www.cert.org/archive/pdf/DoS_trends.pdf$.

[67] Pavel Celeda, Radek Krejci, Jan Vykopal, and Martin Drasar. Embedded malware - an analysis of the chuck norris botnet. In *Proceedings of the 2010 European Conference on Computer Network Defense*, EC2ND '10, pages 3–10, Washington, DC, USA, 2010. IEEE Computer Society.

[68] CERT Advisory CA-1997-28 IP Denial-of-Service Attacks. http://www.cert.org/advisories/ca-1997-28.html, 2010.

[69] Karlof Ch., Sastry N., and Wagner D. Cryptographic Voting Protocols: A Systems Perspective. In *Proceedings of 14th USENIX Security Symposium*, pages 33–50, 2005.

[70] Yanpei Chen, Vern Paxson, and Randy H. Katz. What's new about cloud computing security? Technical Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010.

[71] Jerry Cheng, Starsky H.Y. Wong, Hao Yang, and Songwu Lu. Smartsiren: virus detection and alert for smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, MobiSys '07, pages 258–271, New York, NY, USA, 2007. ACM.

[72] Mihai Christodorescu, Reiner Sailer, Douglas Lee Schales, Daniele Sgandurra, and Diego Zamboni. Cloud security is not (just) virtualization security: a short paper. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, pages 97–102, New York, NY, USA, 2009. ACM.

[73] EG Coffman Jr, Z. Ge, V. Misra, and D. Towsley. Network resilience: exploring cascading failures within BGP. In *Proc. 40th Annual Allerton Conference on Communications, Computing and Control*. Citeseer, 2002.

[74] Hewlett-Packard Company. Print security and identity authorization. http://www.hp.com/united-states/public_slg/spy-museum-security-presentation.pdf, 2007. Presentation.

[75] Computer Emergency Response Team (CERT). Cert advisory ca-1999-17 denial-of-service tools, Mar. 2000 http://www.cert.org/advisories/CA-1999-17.html.

[76] Matt Conover and w00w00 Security Team. w00w00 on Heap Overflows. http://www.cgsecurity.org/exploit/heaptut.txt, January 1999.

[77] Tracy Lynn Cook. Ebook reader virus in 'twilight' pdf files. http://technology.gather.com/viewArticle.action?articleId=281474978284194, June 2010.

[78] Jonathan Corbet. Fun with null pointers. http://lwn.net/Articles/342330/, July 2009.

[79] Crispin Cowan. StackGuard: Automatic Protection From Stack-smashing Attacks. http://seclists.org/bugtraq/1997/Dec/120, December 1997.

[80] Crispin Cowan. FormatGuard. http://lwn.net/2001/0531/a/formatguard.php3, May 2001.

[81] Crispin Cowan, Matt Barringer, Steve Beattie, and Greg Kroah-Hartman. FormatGuard: Automatic Protection From printf Format String Vulnerabilities. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.

[82] Crispin Cowan, Calton Pu, Dave Maier, Heather Hintongif, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In *Proceedings of the 7th USENIX Security Symposium*, pages 63–78, January 1998.

[83] L.A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101, 2009.

[84] CVE. CVE-2009-0475:Integer underflow in the Huffman decoding functionality. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0475, February 2009.

[85] Frankel D. Model driven architecture applying mda to enterprise computing. In *Wiley Publishing*, USA, 2003.

[86] Verton D. In *The hacker diaries: confessions of teenage hackers*, New York: McGraw-Hill/Osborne, USA, 2002.

[87] Boris Danev, Thomas S. Heydt-Benjamin, and Srdjan Čapkun. Physical-layer identification of rfid devices. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 199–214, Berkeley, CA, USA, 2009. USENIX Association.

[88] Alan Dang. Explaining the vulnerability. http://www.tomshardware.com/reviews/hacking-iphone-security,2384-2.html, August 2009. Interview with Charlie Miller.

[89] Darren Dao, Jeannie Albrecht, Charles Killian, and Amin Vahdat. Live debugging of distributed systems. In *CC '09: Proceedings of the 18th International Conference on Compiler Construction*, pages 94–108, Berlin, Heidelberg, 2009. Springer-Verlag.

[90] Satish Das. Privacy and security top cloud concerns. http://information-security-resources.com/2009/09/29/privacy-and-security-top-cloud-concerns/, 2009.

[91] Chris Davies. Samsung Kies Air Adds Wireless Sync & Access to Galaxy S. http://androidcommunity.com/samsung-kies-air-adds-wireless-sync-access-to-galaxy-s-20110103/, January 2011. Android Community.

[92] Mike Davis. SmartGrid Device Security: Adventures in a new medium. *Black Hat, USA*, 2009.

[93] G. de Koning Gans, J.H. Hoepman, and F. Garcia. A practical attack on the MIFARE Classic. *Smart Card Research and Advanced Applications*, pages 267–282, 2008.

[94] Theo de Raadt. The OpenBSD 3.3 Release. http://www.openbsd.org/33.html, May 2003.

[95] Theo de Raadt. The OpenBSD 3.4 Release. http://www.openbsd.org/34.html, November 2003.

[96] Solar Designer. Linux kernel patch from the Openwall Project. http://www.openwall.com/linux/README.shtml.

[97] Solar Designer. Getting around non-executable stack (and fix). http://seclists.org/bugtraq/1997/Aug/63, August 1997.

[98] Solar Designer. Non-executable stack patch. http://lkml.indiana.edu/hypermail/linux/kernel/9706.0/0341.html, June 1997.

[99] Claudia Díaz, Carmela Troncoso, and Andrei Serjantov. On the impact of social network profiling on anonymity. In *Privacy Enhancing Technologies*, pages 44–62, 2008.

[100] David Dittrich. The "stacheldraht" distributed denial of service attack tool, Dec. 1999 http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.

[101] David Dittrich. The "tribe flood network" distributed denial of service attack tool, Oct. 1999 http://staff.washington.edu/dittrich/misc/tfn.analysis.

[102] David Dittrich. The dos project's "trinoo" distributed denial of service attack tool, Oct. 1999 http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[103] John R. Douceur. The Sybil Attack. In *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002.

[104] Saar Drimer and Steven J. Murdoch. Keep your enemies close: distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 7:1–7:16, Berkeley, CA, USA, 2007. USENIX Association.

[105] Tyler Durden. Bypassing PaX ASLR Protection. *Phrack Magazine*, 59(9), July 2002.

[106] Bagheri E. and Ghorbani A. The state of the art in critical infrastructure protection: a framework for convergence. In *International Journal of Critical Infrastructures*, volume 4 of *2*, pages 215–244, 2008.

[107] Bagheri E. and Ghorbani A. Uml-ci: A reference model for profiling critical infrastructure systems. In *Information Systems Frontiers*, volume 12 of *2*, pages 115–139, 2010.

[108] Nickolov E. Critical Information Infrastructure Protection: Analysis Evaluation and Expectations. In *Information & Security. An International Journal*, volume 17, pages 105–119, 2005. Available at http://www.comw.org/tct/fulltext/05nickolov.pdf.

[109] E. Mills. Facebook suspends app that permitted peephole. http://news.cnet.com/8301-10784\3-9977762-7.html, 2008.

[110] M. Egele, A. Moser, C. Kruegel, and E. Kirda. PoX: Protecting Users from Malicious Facebook Applications. 2011.

[111] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. PiOS: Detecting privacy leaks in iOS applications. In *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, February 2011.

[112] Manuel Egele, Christopher Kruegel, Engin Kirda, Heng Yin, and Dawn Song. Dynamic spyware analysis. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, pages 18:1–18:14, Berkeley, CA, USA, 2007. USENIX Association.

[113] Manuel Egele, Peter Wurzinger, Christopher Kruegel, and Engin Kirda. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In Ulrich Flegel and Danilo Bruschi, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5587 of *Lecture Notes in Computer Science*, pages 88–106. Springer Berlin / Heidelberg, 2009.

[114] Toby Ehrenkranz and Jun Li. On the state of ip spoofing defense. *ACM Trans. Internet Technol.*, 9(2):1–29, 2009.

[115] William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 235–245, New York, NY, USA, 2009. ACM.

[116] Robert Epstein. From Russia, with love: How I got fooled (and somewhat humiliated) by a computer. *Scientific American Mind*, pages 16–17, October 2007.

[117] Steven J. et all. Chip and PIN is Broken. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 433–446, 2010. ISBN 978-0-7695-4035-1.

[118] Hiroaki Etoh and Kunikazu Yoda. Protecting from stack-smashing attacks. http://www.trl.ibm.com/projects/security/ssp/main.html, June 2000.

[119] Garcia F., Rossum P., Ronny R., and Schreur W. Wirelessly pickpocketing a mifare classic card. In *IEEE Symposium on Security and Privacy*, 2009.

[120] F-Secure. Bluetooth-worm:symbos/cabir. http://www.f-secure.com/v-descs/cabir.shtml.

[121] F-Secure. More on redbrowser. http://www.f-secure.com/weblog/archives/00000823.html, February 2006.

[122] F-Secure. Windows mobile trojan infojack. http://www.f-secure.com/weblog/archives/00001391.html, February 2008.

[123] F-Secure. Sexy space symbian worm. http://www.f-secure.com/weblog/archives/00001721.html, July 2009.

[124] Nicolas Falliere, Liam O. Murchu, and Eric Chien. W32.Stuxnet dossier. http://goo.gl/N2nhQ, 2011.

[125] B. Feinstein, D. Peck, and I. SecureWorks. Caffeine monkey: Automated collection, detection and analysis of malicious javascript. *Black Hat USA*, 2007, 2007.

[126] A. Felt and D. Evans. Privacy protection for social networking platforms. In *Workshop on Web*, volume 2. Citeseer, 2008.

[127] P. Ferrie. Attacks on more virtual machine emulators. Technical report, Symantec, 2007.

[128] FHM Crew. ASLR bypassing method on 2.6.17/20 Linux Kernel. http://www.exploit-db.com/exploits/13030/, September 2008.

[129] A.D. Flaxman. Expansion and lack thereof in randomly perturbed graphs. *Internet Mathematics*, 4(2):131–147, 2007.

[130] Chris Fleizach, Michael Liljenstam, Per Johansson, Geoffrey M. Voelker, and Andras Mehes. Can you infect me now?: malware propagation in mobile phone networks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, WORM '07, pages 61–68, New York, NY, USA, 2007. ACM.

[131] Aurelien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. Cryptology ePrint Archive, Report 2010/332, June 2010.

[132] S. Furnell. Handheld hazards: The rise of malware on mobile devices. *Computer Fraud & Security*, 2005(5):4–8, 2005.

[133] Ateniese G., Chou1 D. H., B. de Medeiros, and Tsudik G. Sanitizable signatures. In *10th European Symposium on Research in Computer Security, Lecture Notes in Computer Science*, volume 3679, pages 159–177, 2005. DOI: 10.1007/11555827.

[134] Deborah Gage. Malware's new infection route: photo frames. http://articles.sfgate.com/2008-01-26/news/17149112_1_digital-photo-infected-frames, January 2008. SF Gate.

[135] Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly pickpocketing a mifare classic card. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, pages 3–15, Washington, DC, USA, 2009. IEEE Computer Society.

[136] D. Geels, G. Altekar, S. Shenker, and I. Stoica. Replay debugging for distributed applications. In *USENIX Annual Technical Conference*, volume 2006, pages 2–3, 2006.

[137] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA, 2009. ACM.

[138] Zenobia Godschalk. Cloud security alliance continues rapid growth. http://www.cloudsecurityalliance.org/pr20090825.html, 2009.

[139] Nico Golde and Collin Mulliner. SMS-o-Death: From Analyzing To Attacking Mobile Phones on a Large Scale. In *CanSecWest 2011*, Vancouver, British Columbia, Canada, March 2011.

[140] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proc. NDSS*, volume 3. Citeseer, 2003.

[141] Dan Goodin. Did playstation network hackers plan supercomputer botnet? http://www.theregister.co.uk/2011/04/29/sony_playstation_breach_analysis, April 2011. The Register.

[142] T. Goodspeed and A. Francillon. Half-blind attacks: mask ROM bootloaders are dangerous. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, pages 6–6. USENIX Association, 2009.

[143] Google. Google latitude. https://www.google.com/latitude.

[144] Google. Google Latest Cyberattacks. Google Points the Finger at the Chinese Government In Latest Cyberattack. 2011. Available at http://business2press.com/2011/03/20/google-blames-chinese-government-second-gmail-cyber-attack/.

[145] S. Govindavajhala and A.W. Appel. Using memory errors to attack a virtual machine. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 154–165. IEEE, 2003.

[146] Patrick Gray. Telstra distributes malware-infected usb drives at auscert. http://searchsecurity.techtarget.com.au/news/2240019408/Telstra-distributes-malware-infected-USB-drives-at-AusCERT, May 2008.

[147] greydns and Theo de Raadt. OT: PaX question. http://marc.info/?t=105053744900004&r=1&w=2, April 2003.

[148] V. Griffith and M. Jakobsson. Messin2019with Texas Deriving Mother2019s Maiden Names Using Public Records. In *Applied Cryptography and Network Security*, pages 91–103. Springer, 2005.

[149] C. Guo, H.J. Wang, and W. Zhu. Smart-phone attacks and defenses. In *Third Workshop on Hot Topics in Networks (HotNets-III)*. ACM SIGCOMM, November 2004.

[150] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

[151] Sheikh Mahbub Habib, Cyril Jacob, and Tomas Olovsson. A practical analysis of the robustness and stability of the network stack in smartphones. *IEEE International Conference on Computer and Information Technology, ICCIT 2008*, 2008.

[152] Tzipora Halevi and Nitesh Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: the case of acoustic eavesdropping. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 97–108, New York, NY, USA, 2010. ACM.

[153] O. Hallaraker and G. Vigna. Detecting malicious javascript code in mozilla. 2005.

[154] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 129–142, Washington, DC, USA, 2008. IEEE Computer Society.

[155] Derrick Harris. Experts get serious about cloud security. http://gigaom.com/2009/03/31/experts-get-serious-about-cloud-security/, 2009.

[156] R. Heatherly, M. Kantarcioglu, B. Thuraisingham, and J. Lindamood. Preventing Private Information Inference Attacks on Social Networks. 2009.

[157] Michael Howard. Address Space Layout Randomization in Windows Vista. http://blogs.msdn.com/b/michael_howard/archive/2006/05/26/608315.aspx, May 2006.

[158] Michael Howard. Alleged Bugs in Windows Vistas ASLR Implementation. http://blogs.msdn.com/b/michael_howard/archive/2006/10/04/alleged-bugs-in-windows-vista_1920_s-aslr-implementation.aspx, October 2006.

[159] $http://dos-attacks.com/2010/10/26/botnet-economy/$. Botnet Economy.

[160] http://www.bbc.co.uk/news/technology-11971259. Anonymous wikileaks supporters explain web attacks, Dec. 2010.

[161] $http://www.cert.org/incident_notes/IN-2001-09.html$. CERT CC. Code Red II.

[162] http://www.wired.com/epicenter/2009/08/twitter-apparently-down/. Denial-of-service attack knocks twitter offline, August 2009.

[163] http://www.wired.com/threatlevel/2010/12/web20-attack-anonymous/. Joining pro-wikileaks attacks is as easy as clicking a button, Dec. 2010.

[164] $http://www.wired.com/wired/archive/14.06/posts.html?pg=2$. Your Money or Your Site.

[165] Xin Hu and Z. Morley Mao. Accurate real-time identification of ip prefix hijacking. *Security and Privacy, IEEE Symposium on*, 0:3–17, 2007.

[166] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. Spv: secure path vector routing for securing bgp. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, pages 179–192, New York, NY, USA, 2004. ACM.

[167] Nathaniel Husted and Steven Myers. Mobile location tracking in metro areas: malnets and others. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 85–96, New York, NY, USA, 2010. ACM.

[168] M. Hypponen. State of cell phone malware in 2007. http://www.usenix.org/events/sec07/tech/xs, 2007. Invited talk at the 16th usenix security symposium, Boston.

[169] Dobson I., Carreras B. A., and Newman D. E. Probabilistic load-dependent cascading failure with limited component interactions. In *IEEE International Symposium on Circuits and Systems*, Vancouver, Canada, May 2004.

[170] Fovinoa I., Guidib L., Maseraa M., and Stefaninia A. Cyber security assessment of a power plant. In *Electric Power Systems Research*, volume 81, pages 518–526, 2011.

[171] ICANN. Factsheet for the February 6, 2007 Root Server Attack. http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf, March 2007.

[172] IETF. RCF-4033: DNS Security Introduction and Requirements. Technical report, March 2005. http://www.ietf.org/rfc/rfc4033.txt.

[173] Apple Inc. Mac os x security. www.samug.org/web/MacOSX_Leopard_Security_TB.pdf, October 2007.

[174] Internet Security News. Akamai DDoS Attack. http://www.landfield.com/isn/mail-archive/2004/Jun/0088.html, Jun 2004.

[175] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu. Reverse Social Engineering Attacks in Online Social Networks . *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2011.

[176] D. Irani, S. Webb, J. Giffin, and C. Pu. Evolutionary study of phishing. In *eCrime Researchers Summit, 2008*, pages 1–10. IEEE, 2008.

[177] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, 2007.

[178] Markus Jakobsson and Karl-Anders Johansson. Retroactive detection of malware with applications to mobile platforms. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec'10, pages 1–13, Berkeley, CA, USA, 2010. USENIX Association.

[179] Lukas Jeter, Meenakshi Mani, and Tia Reinschmidt. Smart Phone Malware: The danger and protective strategies. May 2010.

[180] Hong J.H.C.S., Ho Ju S., Lim Y.H., Lee B.S., and Hyun D.H. A security mechanism for automation control in PLC-based networks. In *Proceedings of the ISPLC 2007. IEEE International Symposium on Power Line Communications and Its Applications*, Pisa, Italy, Mar 2007.

[181] T. Jim, N. Swamy, and M. Hicks. Defeating script injection attacks with browser-enforced embedded policies. In *Proceedings of the 16th international conference on World Wide Web*, pages 601–610. ACM, 2007.

[182] Geers K. The challenge of cyberattack deterrence. In *Computer law & Security Review*, volume 26, pages 298–303, 2010.

[183] Koscher K., Czeskis A., Roesner F., Patel S., and Kohno T. Experimental Security Analysis of a Modern Automobile. In *Proceedings 2010 IEEE Symposium on Security and Privacy*, pages 447–462, 2010. ISBN 978-0-7695-4035-1.

[184] Stouffer K., Falco J., and KentGuide K. Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security. In *National Institute of Standards and Technology, USA, Special Publication 800-82*, 2006.

[185] Peter Kafka. The twitterhack is cloud computing's wake-up call: Time for security that works. http://mediamemo.allthingsd.com/20090715/the\discretionary{-}{}{}twitterhack-is-cloud-computings-wakeup\discretionary{-}{}{}call-time-for-security-that-works/, 2009.

[186] Christoph Karlberger, Guenter Bayler, Christopher Kruegel, and Engin Kirda. Exploiting Redundancy in Natural Language to Penetrate Bayesian Spam Filters. In *First USENIX Workshop on Offensive Technologies (WOOT '07), Boston, MA*, August 2007.

[187] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Autonomous security for autonomous systems. *Computer Networks*, 52(15):2908 – 2923, 2008. Complex Computer and Communication Networks.

[188] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (s-bgp). *Selected Areas in Communications, IEEE Journal on*, 18(4):582 –592, April 2000.

[189] Stephen T. Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure border gateway protocol (s-bgp) - real world performance and deployment issues. In *NDSS*, 2000.

[190] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting energy-greedy anomalies and mobile malware variants. In *MobiSys*, pages 239–252, 2008.

[191] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch. Subvirt: Implementing malware with virtual machines. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society.

[192] Kingpin Kingpin and Mudge Mudge. Security analysis of the palm operating system and its weaknesses against malicious code threats. In *Proceedings of the 10th conference on USENIX Security Symposium - Volume 10*, pages 11–11, Berkeley, CA, USA, 2001. USENIX Association.

[193] Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard A. Kemmerer. *Behavior-Based Spyware Detection*. 2006.

[194] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2010.

[195] Karl Koscher, Ari Juels, Vjekoslav Brajkovic, and Tadayoshi Kohno. Epc rfid tag security weaknesses and defenses: passport cards, enhanced drivers licenses, and beyond. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 33–42, New York, NY, USA, 2009. ACM.

[196] E. Kranakis, T. Wan, and PC Oorschot. On Inter-domain Routing Security and Pretty Secure BGP (psBGP). *Carleton University, School of Computer Science, Technical Report TR-05-08*, 2005.

[197] Brian Krebs. Amazon: Hey spammers, get off my cloud! http://blog.washingtonpost.com/securityfix/2008/07/amazon_hey_spammers_get_off_my.html, July 2008.

[198] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. Topology-based detection of anomalous bgp messages. In Giovanni Vigna, Christopher Kruegel, and Erland Jonsson, editors, *Recent Advances in Intrusion Detection*, volume 2820 of *Lecture Notes in Computer Science*, pages 17–35. Springer Berlin / Heidelberg, 2003.

[199] McAfee Labs. Mcafee threats report fourth quarter 2010, 2011. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2010.pdf.

[200] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. PHAS: A prefix hijack alert system. In *Proc. USENIX Security Symposium*, 2006.

[201] Kris Lamb. Virtualization and security. http://blogs.iss.net/archive/virtblog.html, September 2007.

[202] Lucas Laursen. Fake facebook pages spin web of deceit. *Nature*, 458(1089), 2009.

[203] M. Lesk. The new front line: Estonia under cyberassault. *Security Privacy, IEEE*, 5(4):76 –79, july-aug. 2007.

[204] Jun Li, J. Mirkovic, Mengqiu Wang, P. Reiher, and Lixia Zhang. Save: source address validity enforcement protocol. In *IEEE INFOCOM 2002.*, volume 3, pages 1557–1566, June 2002.

[205] Liquida. Liquida Blog. 2011. Available at http://www.liquida.com/article/19143241/iran-skype-mozilla/.

[206] David Litchfield. Defeating the Stack Based Buffer Overflow Prevention Mechanism of Microsoft Windows 2003 Server . http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-litchfield.pdf, September 2003.

[207] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter: Preventing your cellphone from spies. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 244–264, Berlin, Heidelberg, 2009. Springer-Verlag.

[208] Thomas Lopatic. Vulnerability in NCSA HTTPD 1.3. http://seclists.org/bugtraq/1995/Feb/109, February 1995.

[209] Georgios Loukas and Glay ke. Protection against denial of service attacks: A survey. *The Computer Journal*, 2009.

[210] M.T. Louw and VN Venkatakrishnan. Blueprint: Robust prevention of cross-site scripting attacks for existing browsers. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 331–346. IEEE, 2009.

[211] M.M. Lucas and N. Borisov. flybynight: Mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 1–8. ACM, 2008.

[212] W. Luo, Q. Xie, and U. Hengartner. Facecloak: An architecture for user privacy on social networking sites. In *2009 International Conference on Computational Science and Engineering*, pages 26–33. IEEE, 2009.

[213] Clarkson M., Chong S., and Myers A. C. Civitas: Toward a secure voting system. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 354–368, 2008.

[214] Dunn M. The socio-political dimensions of critical information infrastructure protection ciip. In *International Journal of Critical Infrastructures*, volume 1, pages 258–268, 2005.

[215] Lee M. Who is Gary McKinnon? In *ABC News*, May 2006.

[216] Majdalawieh M., Parisi-Presicce F., and Wijesekera D. Distributed network protocol security (DNPSec) security framework. In *Proceedings of the 21st Annual Computer Security Applications Conference*, Tucson, Arizona, USA, Dec 2005.

[217] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1245–1254, New York, NY, USA, 2009. ACM.

[218] W. Maes, T. Heyman, L. Desmet, and W. Joosen. Browser protection against cross-site request forgery. In *Proceedings of the first ACM workshop on Secure execution of untrusted code*, pages 3–10. ACM, 2009.

[219] Justin Manweiler, Ryan Scudellari, and Landon P. Cox. SMILE: encounter-based trust for mobile social services. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 246–255, New York, NY, USA, 2009. ACM.

[220] Z.M. Mao, R. Govindan, G. Varghese, and R.H. Katz. Route flap damping exacerbates Internet routing convergence. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 221–233. ACM, 2002.

[221] John Markoff. Researchers show how a cars electronics can be taken over remotely. http://www.nytimes.com/2011/03/10/business/10hack.html, March 2011. New York Times.

[222] John Markoff. Before the gunfire, cyberattacks, Aug. 2008 http://www.nytimes.com/2008/08/13/technology/13cyber.html?em.

[223] Denis Maslennikov. Mobile malware evolution: An overview, part 4. http://www.securelist.com/en/analysis/204792168/Mobile_Malware_Evolution_An_Overview_Part_4, 2011.

[224] Paul McFedries. The cloud is the computer. http://www.spectrum.ieee.org/computing/hardware/the-cloud-is-the-computer, August 2008.

[225] Stephen McLaughlin, Dmitry Podkuiko, Adam Delozier, Sergei Miadzvezhanka, and Patrick McDaniel. Embedded firmware diversity for smart electric meters. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[226] Leslie Meredith. Malware implicated in fatal Spanair plane crash. http://www.msnbc.msn.com/id/38790670/ns/technology_and_science-security/, August 2010. Security on msnbc.com.

[227] Microsoft. A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003. http://support.microsoft.com/kb/875352/, September 2006.

[228] Brad Miller. Cloud security alliance. http://www.cloudsecurityalliance.org/, 2009.

[229] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, 2004.

[230] G. Mishne, D. Carmel, and R. Lempel. Blocking blog spam with language model disagreement. In *Proceedings of the first international workshop on adversarial information retrieval on the Web (AIRWeb)*. Citeseer, 2005.

[231] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007.

[232] Kevin Mitnick, William L. Simon, and Steve Wozniak. *The Art of Deception: Controlling the Human Element of Security*. Wiley, 2002.

[233] P. Mockapetris. Domain names - concepts and facilities. Technical Report 882, Internet Engineering Task Force, November 1983. http://tools.ietf.org/html/rfc882.

[234] Ingo Molnar. "Exec Shield", new Linux security feature. http://lwn.net/Articles/31032/, May 2003.

[235] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *In Proceedings of the 10th Usenix Security Symposium*, pages 9–22, 2001.

[236] Shawn Moyer and Nathan Hamiel. Satan is on My Friends List: Attacking Social Networks. http://www.blackhat.com/html/bh-usa-08/bh-usa-08-archive.html, 2008.

[237] Tilo Müller. ASLR Smack & Laugh Reference. http://ivanlef0u.nibbles.fr/repo/expl0it/ASLRpaper.pdf, February 2008.

[238] C. Mulliner and J.P. Seifert. Rise of the iBots: 0wning a telco network. In *Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software (Malware)*, Nancy, France, October 2010.

[239] C. Mulliner and G. Vigna. Vulnerability Analysis of MMS User Agents. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Miami, FL, December 2006.

[240] Collin Mulliner and Charlie Miller. Fuzzing the phone in your phone. *Black Hat*, June 2009.

[241] Collin Mulliner and Charlie Miller. Injecting sms messages into smart phones for security analysis. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, WOOT'09, pages 5–5, Berkeley, CA, USA, 2009. USENIX Association.

[242] Basu N., Pryor R., and Quint T. Aspen: A microsimulation model of the economy. In *Computational Economics*, volume 12 of *3*, pages 223–241, 1998. DOI: 10.1023/A:1008691115079.

[243] Falliere N., Murchu L.O., and Chien E. W32.Stuxnet Dossier. In *Symantec Corporation, Version 1.4*, Feb 2011. Available at http://bit.ly/ns201010-stuxnet/.

[244] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173–187. IEEE, 2009.

[245] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[246] Arvind Narayanan, Narendran Thiagarajan, Michael Hamburg, Mugdha Lakhani, and Dan Boneh. Location Privacy via Private Proximity Testing. In *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, February 2011.

[247] Daniel C. Nash, Thomas L. Martin, Dong S. Ha, and Michael S. Hsiao. Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, PERCOMW '05, pages 141–145, Washington, DC, USA, 2005. IEEE Computer Society.

[248] Christine Negroni. Tracking your wi-fi trail. http://www.nytimes.com/2011/03/22/business/22airport.html?_r=1&ref=airports, March 2011. New York Times.

[249] NERC. 2011. Available at http://www.nerc.com.

[250] Nergal. The Advanced Return-Into-Lib(c) exploits (PaX Case study). *Phrack Magazine*, 58(4), December 2001.

[251] Tim Newsham. Format String Attacks. http://www.thenewsh.com/~newsham/format-string-attacks.pdf, September 2000.

[252] J. Ng. Extensions to BGP to support secure origin BGP (soBGP). *IETF Draft: draft-ng-sobgp-bgp-extensions-01. txt*, 2002.

[253] David M. Nicol, Sean W. Smith, and Meiyuan Zhao. Evaluation of efficient security for bgp route announcements using parallel simulation. *Simulation Modelling Practice and Theory*, 12(3-4):187 – 216, 2004. Modeling and Simulation of Distributed Systems and Networks.

[254] Dennis K. Nilsson. *How to Secure the Connected Car*. Phd thesis, Chalmers University of Technology, May 2009. Department of Computer Science and Engineering.

[255] O. Nordstrom and C. Dovrolis. Beware of BGP attacks. *ACM SIGCOMM Computer Communication Review*, 34(2):1–8, 2004.

[256] Jon Oberheide, Kaushik Veeraraghavan, Evan Cooke, Jason Flinn, and Farnam Jahanian. Virtualized in-cloud security services for mobile devices. In *Proceedings of the First Workshop on Virtualization in Mobile Computing*, MobiVirt '08, pages 31–35, New York, NY, USA, 2008. ACM.

[257] Brendan O'Connor. Vulnerabilities in Not-So Embedded Systems. *Black Hat, USA*, 2006.

[258] OGC-CI. Sep 2002. Available at http://www.opengeospatial.org.

[259] Open Web Application Security Project. Denial of service, Mar. 2010 $https://www.owasp.org/index.php/Denial_of_service$.

[260] T. Ormandy. An empirical study into the security exposure to host of hostile virtualized environments. In *Proceedings of CanSecWest Applied Security Conference*, 2007.

[261] Donzelli P. and Setola R. Identifying and evaluating risks related to enterprise dependencies: a practical goal-driven risk analysis framework. In *International Journal of Risk Assessment and Management*, volume 7 of *8*, pages 1120–1137, 2007.

[262] Goble P. Russia: analysis from washington: a real battle on the virtual front. In *Radio Free Europe/Radio Liberty*, volume 26, pages 298–303, October 1999.

[263] D. Senie P. Ferguson. Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. rfc 2827. may 2000.

[264] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31:38–47, July 2001.

[265] Captain Planet. A Eulogy for Format Strings. *Phrack Magazine*, 67(9), November 2010.

[266] Michalis Polychronakis, Panayiotis Mavrommatis, and Niels Provos. Ghost turns zombie: exploring the life cycle of web-based malware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 11:1–11:8, Berkeley, CA, USA, 2008. USENIX Association.

[267] Alin Rad Pop. Dep/aslr implementation progress in popular third-party windows applications.
secunia.com/gfx/pdf/DEP_ASLR_2010_paper.pdf, June 2010.

[268] Raluca Ada Popa, Hari Balakrishnan, and Andrew Blumberg. VPriv: Protecting Privacy in Location-Based Vehicular Services. In *18th USENIX Security Symposium*, Montreal, Canada, August 2009.

[269] Phillip A. Porras, Hassen Saïdi, and Vinod Yegneswaran. An analysis of the ikee.b iphone botnet. In *MobiSec*, pages 141–152, 2010.

[270] Georgios Portokalidis, Philip Homburg, Kostas Anagnostakis, and Herbert Bos. Paranoid android: Versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, Austin, Texas, December 2010.

[271] V. Prevelakis and D. Spinellis. The Athens Affair. *Spectrum, IEEE*, 44(7):26–33, 2007.

[272] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All your iframes point to us. In *Proceedings of the 17th conference on Security symposium*, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

[273] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 4–4, Berkeley, CA, USA, 2007. USENIX Association.

[274] Niels Provos, Moheeb Abu Rajab, and Panayiotis Mavrommatis. Cybercrime 2.0: when the cloud turns dark. *Commun. ACM*, 52:42–47, April 2009. Also publicly available at http://cacm.acm.org/magazines/2009/4/22956-cybercrime-20/fulltext.

[275] Akella R., Tang H., and McMillin B. Analysis of information flow security in cyber-physical systems. In *International Journal of Critical Infrastructure Protection*, volume 4, pages 157–173, 2010.

[276] Chandia R., Gonzalez J., Kilpatrick T., Papa M., and Shenoi S. Security strategies for scada networks. In *International Conference on Critical Infrastructure Protection. Proceeding of the First Annual IFIP Working Group 11.10*, Dartmouth College Hanover New Hampshire, USA, 2007.

[277] Radmilo Racic, Denys Ma, Hao Chen, and Xin Liu. Exploiting and defending opportunistic scheduling in cellular data networks. *IEEE Transactions on Mobile Computing*, 9:609–620, May 2010.

[278] Barath Raghavan, Saurabh Panjwani, and Anton Mityagin. Analysis of the spv secure routing protocol: weaknesses and lessons. *SIGCOMM Comput. Commun. Rev.*, 37:29–38, March 2007.

[279] Ali Rahbar. An analysis of Microsoft Windows Vista's ASLR.
http://www.sysdream.com/articles/Analysis-of-Microsoft-Windows-Vistas-ASLR.pdf, October 2006.

[280] Kasper Bonne Rasmussen, Claude Castelluccia, Thomas S. Heydt-Benjamin, and Srdjan Capkun. Proximity-based access control for implantable medical devices. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 410–419, New York, NY, USA, 2009. ACM.

[281] Paruj Ratanaworabhan, Benjamin Livshits, and Benjamin Zorn. Nozzle: a defense against heap-spraying code injection attacks. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 169–186, Berkeley, CA, USA, 2009. USENIX Association.

[282] Redhat. Selinux policy. https://access.redhat.com/kb/docs/DOC-18042, August 2009.

[283] P. Reynolds, C. Killian, J.L. Wiener, J.C. Mogul, M.A. Shah, and A. Vahdat. Pip: Detecting the unexpected in distributed systems. In *Symposium on Networked Systems Design and Implementation*, pages 115–128, 2006.

[284] Gerardo Richarte. Four different tricks to bypass StackShield and StackGuard protection. http://www.coresecurity.com/files/attachments/StackGuard.pdf, June 2002.

[285] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum. Is your cat infected with a computer virus? In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pages 169–179, Washington, DC, USA, 2006. IEEE Computer Society.

[286] Gera & Riq. Advances in format string exploitation. *Phrack Magazine*, 59(7), July 2002.

[287] Thomas Ristenpart, Gabriel Maganis, Arvind Krishnamurthy, and Tadayoshi Kohno. Privacy-preserving location tracking of lost or stolen devices: cryptographic techniques and replacing trusted third parties with dhts. In *Proceedings of the 17th conference on Security symposium*, pages 275–290, Berkeley, CA, USA, 2008. USENIX Association.

[288] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud! Exploring information leakage in third-party compute clouds. In Somesh Jha and Angelos Keromytis, editors, *Proceedings of CCS 2009*, pages 199–212. ACM Press, November 2009.

[289] Thomas Ristenpart and Scott Yilek. When Good Randomness Goes Bad: Virtual Machine Reset Vulnerabilities and Hedging Deployed Cryptography. In *Proceedings of the Network and Distributed System Security Symposium - NDSS 2010*. Internet Society, March 2010.

[290] M. Roesch et al. Snort-lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, pages 229–238. Seattle, Washington, 1999.

[291] Neil Roiter. Mcafee, verizon business team up on cloud security. http://www.searchsecurityasia.com/content/mcafee-verizon-business-team-cloud-security, 2009.

[292] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[293] Joanna Rutkowska. Subverting vista kernel for fun and profit. Black Hat USA, 2006.

[294] Lawson S. Beyond Cyber-Doom: Cyberattack Scenarios and the Evidence of History. In *WORKING PAPER No. 11-01 Mercatus Center at George Mason University*, Jan 2011. Available at http://mercatus.org/sites/default/files/publication/beyond-cyber-doom-cyber-attack-scenarios-evidence-history_1.pdf.

[295] Panzieri S., Setola R., and Ulivi G. An agent based simulator for critical interdependent infrastructures. In *Proceedings of 2nd International Conference on Critical Infrastructures*, 2004.

[296] Rinaldi S. Modeling and simulating critical infrastructures and their interdependencies In: System Sciences. In *Proceedings of the 37th Annual Hawaii International Conference on on System Sciences (HICSS'04)*, 2004.

[297] T. Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, and Tadayoshi Kohno. Devices that tell on you: privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 5:1–5:16, Berkeley, CA, USA, 2007. USENIX Association.

[298] Marcia Savage. Security challenges with cloud computing services. http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1368905,00.html, 2009.

[299] Dries Schellekens, PaX Team, and Anonymous. recent OpenBSD changes vs PaX. http://undeadly.org/cgi?action=article&sid=20030417082752&pid=20&mode=expanded, April 2003.

[300] Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kiraz, Kamer A. Yüksel, Seyit A. Camtepe, and Sahin Albayrak. Static analysis of executables for collaborative malware detection on android. In *Proceedings of the 2009 IEEE international conference on Communications*, ICC'09, pages 631–635, Piscataway, NJ, USA, 2009. IEEE Press.

[301] Charles Schmidt and Tom Darby. The What, Why, and How of the 1988 Internet Worm. http://www.snowplow.org/tom/worm/worm.html, July 2001.

[302] Scut / Team Teso. Exploiting Format String Vulnerabilities. http://julianor.tripod.com/bc/formatstring-1.2.pdf, September 2001.

[303] K. Seo, C. Lynn, and S. Kent. Public-key infrastructure for the secure border gateway protocol (s-bgp). In *DARPA Information Survivability Conference Exposition II, 2001. DISCEX '01. Proceedings*, volume 1, pages 239 –253 vol.1, 2001.

[304] Nicolas Seriot. iPhone Privacy. http://seriot.ch/resources/talks_papers/iPhonePrivacy.pdf, 2010. Black Hat, DC, Arlington, Virginia, US.

[305] Hovav Shacham, Matthew Page, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, and Dan Boneh. On the Effectiveness of Address-Space Randomization. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 298–307. ACM, October 2004.

[306] M. Shehab, A. Squicciarini, and G.J. Ahn. Beyond user-to-user access control for online social networks. *Information and Communications Security*, pages 174–189, 2008.

[307] K. Singh, S. Bhola, and W. Lee. xBook: Redesigning privacy control in social networking platforms. In *Proceedings of the 18th conference on USENIX security symposium*, pages 249–266. USENIX Association, 2009.

[308] Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating bluetooth as a medium for botnet command and control. In *Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment*, DIMVA'10, pages 61–80, Berlin, Heidelberg, 2010. Springer-Verlag.

[309] Stephen M. Specht. Distributed denial of service: taxonomies of attacks, tools and countermeasures. In *Proceedings of the International Workshop on Security in Parallel and Distributed Systems*, pages 543–550, 2004.

[310] Brad Spengler. PaX: The Guaranteed End of Arbitrary Code Execution. http://grsecurity.net/PaX-presentation.ppt, October 2003.

[311] Brad Spengler. Enlightenment. http://grsecurity.net/~spender/enlightenment.tgz, September 2009.

[312] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 133–145. ACM, 2002.

[313] StackShield. Stack Shield: A "stack smashing" technique protection tool for Linux. http://www.angelfire.com/sk/stackshield/info.html, December 1999.

[314] E Steel and G.A Fowler. Facebook in online privacy breach; applications transmitting identifying information. http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html, 2010.

[315] Allen Stern. Update from amazon regarding friday's s3 downtime. http://www.centernetworks.com/amazon-s3-downtime-update, July 2008.

[316] Brett Stone-Gross, Marco Cova, Christopher Kruegel, and Giovanni Vigna. Peering through the iframe. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, IEEE INFOCOM 2011, 2011.

[317] G. Stringhini, C. Kruegel, and G. Vigna. Detecting Spammers on Social Networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[318] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy H. Katz. Listen and whisper: security mechanisms for bgp. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[319] Andy Sudduth. The what, why, and how of the 1988 internet worm, Nov. 1988 http://www.snowplow.org/tom/worm/worm.html.

[320] David Dittrich Sven Dietrich, Neil Long. An analysis of the "shaft" distributed denial of service tool, Mar. 2000 $http://home.adelphi.edu/\ spock/shaft_analysis.txt$.

[321] Neil Long Sven Dietrich, George Weaver and David Dittrich. The "mstream" distributed denial of service attack tool, May. 2000 http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[322] Mander T., Nabhani F., L. Wang, and Cheung R. Data object based security for DNP3 over TCP/IP for increased utility commercial aspects security. In *Proceedings of the Power Engineering Society General Meeting*, Tucson, Arizona, Dec 2007.

[323] Pax Team. Design & Implementation of PAGEEXEC. http://pax.grsecurity.net/docs/pageexec.old.txt, November 2000.

[324] PaX Team, Theo de Raadt, and Dries Schellekens. [OT] PaX. http://marc.info/?t=105058908400003&r=1&w=2, April 2003.

[325] The Amazon S3 Team. Amazon s3 availability event: July 20, 2008. http://status.aws.amazon.com/s3-20080720.html, July 2008.

[326] Julien Tinnes and Tavis Ormandy. Bypassing Linux' NULL pointer dereference exploit prevention (mmap_min_addr). http://blog.cr0.org/2009/06/bypassing-linux-null-pointer.html, June 2009.

[327] Julien Tinnes and Tavis Ormandy. Linux NULL pointer dereference due to incorrect proto_ops initializations (CVE-2009-2692). http://blog.cr0.org/2009/08/linux-null-pointer-dereference-due-to.html, August 2009.

[328] Linus Torvalds. Linux 2.6.25 ChangeLog-2.6.25. http://kernel.org/pub/linux/kernel/v2.6/ChangeLog-2.6.25, April 2008.

[329] T. Toth and C. Krugel. Accurate buffer overflow detection via abstract payload execution. In *RAID*, pages 274–291, 2002.

[330] Patrick Traynor, William Enck, Patrick McDaniel, and Thomas F. La Porta. Mitigating attacks on open functionality in sms-capable cellular networks. In *MOBICOM*, pages 182–193, 2006.

[331] Patrick Traynor, Michael Lin, Machigar Ongtang, Vikhyath Rao, Trent Jaeger, Patrick McDaniel, and Thomas La Porta. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 223–234, New York, NY, USA, 2009. ACM.

[332] Patrick Traynor, Patrick McDaniel, and Thomas La Porta. On attack causality in internet-connected cellular networks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 21:1–21:16, Berkeley, CA, USA, 2007. USENIX Association.

[333] Tymm Twillman. Exploit for proftpd 1.2.0pre6. http://seclists.org/bugtraq/1999/Sep/328, September 1999.

[334] Arjan van de Ven. New Security Enhancements in Red Hat Enterprise Linux v.3, update 3. http://www.redhat.com/f/pdf/rhel/WHP0006US_Execshield.pdf, August 2004.

[335] Arjan van de Ven. Patch 0/6 virtual address space randomisation. http://lkml.org/lkml/2005/1/27/56, January 2005.

[336] Arjan van de Ven. Summary of changes from v2.6.11 to v2.6.12-rc1. http://www.kernel.org/pub/linux/kernel/v2.6/testing/v2.6.12/ChangeLog-2.6.12-rc1, March 2005.

[337] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[338] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, and G. Vigna. Cross site scripting prevention with dynamic data tainting and static analysis. In *Proceeding of the Network and Distributed System Security Symposium (NDSS)*, volume 42. Citeseer, 2007.

[339] Martin Vuagnoux and Sylvain Pasini. Compromising Electromagnetic Emanations of Wired and Wireless Keyboards. In *Proceedings of the 18th USENIX Security Symposium*, pages 1–16, Montreal, Canada, 2009. USENIX Association. This paper received the Outstanding Student Paper Award.

[340] Perry Wagle and Crispin Cowan. StackGuard: Simple Stack Smash Protection for GCC. In *Proceedings of the GCC Developers Summit*, pages 243–256, May 2003.

[341] Dan Walsh. Confining the unconfined. http://danwalsh.livejournal.com/30084.html, July 2009.

[342] Yi-Min Wang, Doug Beck, Xuxian Jiang, and Roussi Roussev. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *IN NDSS*, 2006.

[343] S. Webb, J. Caverlee, and C. Pu. Social Honeypots: Making Friends with a Spammer Near You. In *Conference on Email and Anti-Spam (CEAS)*, 2008.

[344] Ollie Whitehouse. An analysis of address space layout randomization on windows vista. http://www.symantec.com/avcenter/reference/Address_Space_Layout_Randomization.pdf, March 2007.

[345] WikiLeaks. Pro WikiLeaks Attacks. 2010. Available at http://www.csmonitor.com/Business/new-economy/2010/1208/WikiLeaks-cyberattacks-now-involve-Visa-Facebook-Twitter-MasterCard.

[346] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. In *IEEE Security and Privacy*, Mar./Apr. 2007.

[347] Scott Wilson. Appengine outage. http://www.cio-weblog.com/50226711/appengine_outage.php, June 2008.

[348] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy*, pages 223–238. IEEE, 2010.

[349] B. Wu and B.D. Davison. Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 820–829. ACM, 2005.

[350] Sanjay Rao Xin Sun, Ruben Torres. Ddos attacks by subverting membership management in p2p systems. In *3rd IEEE Workshop on Secure Network Protocols*, pages 1–6, 2007.

[351] Haimes Y. Y., Horowitz B. M., Lambert J. H., Santos J. R., Lian C., and Crowther K. G. Inoperability input-output model for interdependent infrastructure sectors. i: Theory and methodology. In *Journal of Infrastructure Systems*, volume 11 of *2*, pages 67–79, 2005.

[352] Haimes Y. Y., Horowitz B. M., Lambert J. H., J. R. Santos, C. Lian, and Crowther K. G. Noperability input-output model for interdependent infrastructure sectors. i: Theory and methodology. In *Journal of Infrastructure Systems*, volume 11 of *2*, pages 80–92, 2005.

[353] Liu Y., Ning P., and Reiter M. False Data Injection Attacks against State Estimation in Electric Power Grids. In *Proceedings of 16th ACM conference on Computer and communications security*, pages 21–32, 2010. DOI 10.1145/1653662.1653666.

[354] Guanhua Yan and Stephan Eidenbenz. Modeling propagation dynamics of bluetooth worms (extended version). *IEEE Transactions on Mobile Computing*, 8:353–368, 2009.

[355] Guanhua Yan, Stephan Eidenbenz, and Emanuele Galli. Sms-watchdog: Profiling social behaviors of sms users for anomaly detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 202–223, Berlin, Heidelberg, 2009. Springer-Verlag.

[356] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 116–127, New York, NY, USA, 2007. ACM.

[357] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 267–278. ACM, 2006.

[358] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *IEEE Symposium on Security and Privacy*, 2008.

[359] Saira Zahid, Muhammad Shahzad, Syed Ali Khayam, and Muddassar Farooq. Keystroke-based user identification on smart phones. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 224–243, Berlin, Heidelberg, 2009. Springer-Verlag.

[360] Vyacheslav Zakorzhevsky. Monthly malware statistics, march 2011. http://www.securelist.com/en/analysis/204792170/Monthly_Malware_Statistics_March_2011, March 2011.

[361] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated path authentication for efficient bgp security. In *Proceedings of the 12th ACM conference on Computer and communications security*, CCS '05, pages 128–138, New York, NY, USA, 2005. ACM.

[362] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. An analysis of bgp multiple origin as (moas) conflicts. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 31–35, New York, NY, USA, 2001. ACM.

[363] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. Detection of invalid routing announcement in the internet. *Dependable Systems and Networks, International Conference on*, 0, 2002.

[364] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM, 2009.