SECURE2013

ANDROTOTAL
A SCALABLE FRAMEWORK FOR ANDROID ANTIMALWARE TESTING

Federico Maggi, Andrea Valdi, Stefano Zanero
Politecnico di Milano, DEIB
fede@maggi.cc
ROADMAP

1. Android threats and protections
2. Limitations
3. Testing antimalware
4. AndroTotal
5. Status
1. ANDROID THREATS AND PROTECTIONS
   2. LIMITATIONS
   3. TESTING ANTIMALWARE
   4. ANDROTOTAL
   5. STATUS
ANDROID FACTS

- Android is the most popular mobile platform (79%)
- Rich marketplaces stocked with apps
- Very attractive target for attackers
ATTACKERS GOALS

- Steal sensitive data (intercept texts or calls)
- Turn devices into bots (perform malicious actions)
- Financial gain (call or text premium numbers)
GROWTH OF MALICIOUS APPS (2011–2012)

NUMBER OF MOBILE 'THREATS' (Q1 2013)

- Symantec: ~3,900
- McAfee: ~60,000
- TrendMicro: ~509,000

Google @ VB2013: Situation is vastly exaggerated
GOOGLE’S LAYERED SECURITY APPROACH

- Google Play vetting
- Install and permission confirmation
- SMS/call blacklisting and quota
- Runtime checks (?)
- App sandboxing
"Sensitive" operations require static permissions
1. Threats and Protections

2. Limitations

3. Testing AntiMalware

4. AndroTotal

5. Status
ANTIMALWARE LIMITATIONS

• No primitives for auditing running processes
• Workarounds:
  ■ Signature-based matching
  ■ Custom kernel (e.g., intercept syscalls)
  ■ Root the device and increase the antimalware's privileges
MALWARE LIMITATIONS

- Less freedom: a malware is an isolated app itself
- Workarounds:
  - Social engineering
  - Signature evasion
SIGNATURE EVASION
MORE VARIANTS THAN DISTINCT FAMILIES

SIGNATURE EVASION

OBFUSCATION, ENCRYPTION, REPACKAGING

ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-Virus Systems, DIVMA2013

DroidChameleon: Evaluating Android Anti-malware against Transformation Attacks, AsiaCCS2013

Based on this research we implemented 11 mutation scripts.
1. THREATS AND PROTECTIONS
2. LIMITATIONS
3. TESTING ANTIMALWARE
4. ANDROTOTAL
5. STATUS
ANTIMALWARE PRODUCTS

- About 100 (free) antimalware apps
- Extra features on rooted devices
HOW TO TEST THEM?

1. Obtain $M$ samples of known malware
2. Apply $T$ transformations to each sample
3. Analyze $M \times T$ variants with $P$ antimalware apps
4. Repeat for each of the $A$ Android versions
NUMBERS

- M = 1,000 (very conservative)
- T = 11
- P = 100
- A = 3 (2.3, 4.1, 4.2)

1,000 \times 11 \times 100 \times 3 = 3,300,000 \text{ TESTS}
LACK OF AUTOMATION TOOLS
VIRUSTOTAL.COM?

- Command-line, desktop-based AVs with signatures for Android
- Unclear whether the same signatures will work on the respective mobile products
- No versioning support
STATE OF THE ART

  - Human oracle is needed
  - Focus on transformation
  - Focus on transformation
TECHNICAL REQUIREMENTS

- Scalable architecture
- Android antimalware products are UI driven
1. THREATS AND PROTECTIONS
2. LIMITATIONS
3. TESTING ANTIMALWARE
4. ANDROTOTAL
5. STATUS
- SDK for writing UI tests/scrapers
- Pluggable adapters for each antimalware
- Parametric tests (e.g., version, platform)
- Task queues with distributed workers
CHARACTERISTICS

- Web frontend for humans
- JSON/REST API for machines
- Pluggable code-transformation modules
- Works on both emulators and physical devices
Scan application (advanced)

Sample File

Is this sample a malware?
- Yes
- No
- I do not know

Force sample reanalysis

Are you human?

<table>
<thead>
<tr>
<th>Antivirus name</th>
<th>Antivirus version</th>
<th>Android platform</th>
<th>Detection method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trend Micro, Mobile Security &amp; Antivirus</td>
<td>2.6.2</td>
<td>Android 4.1.2</td>
<td>On install</td>
</tr>
<tr>
<td>AVAST Software, avast! Mobile Security</td>
<td>2.0.3380</td>
<td>Android 4.1.2</td>
<td>On install</td>
</tr>
<tr>
<td>AVAST Software, avast! Mobile Security</td>
<td>2.0.3380</td>
<td>Android 4.1.2</td>
<td>On demand</td>
</tr>
<tr>
<td>AVAST Software, avast! Mobile Security</td>
<td>2.0.3917</td>
<td>Android 4.1.2</td>
<td>On install</td>
</tr>
<tr>
<td>Sample MD5</td>
<td>cbdf63b2e5666799c4b74a8cd15565dd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample SHA-1</td>
<td>d9c2bc199769f8e1c817cced23f1860f5125bdaf6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample SHA-256</td>
<td>d1de9bb4d7451f7e7e4b5bd6bab529e7411e3dbe90d468243ef87a5ed98941e</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File size</td>
<td>959488 Bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First seen on</td>
<td>08 May 2013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package name</td>
<td>com.issghai.thattere</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File names</td>
<td>com.issghai.thattere.apk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External analysis</td>
<td>[VirusTotal] [SandDroid]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Last 10 scans performed on this sample**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Antivirus Name</th>
<th>Detected name</th>
<th>Date</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 4.1.2</td>
<td>Doctor Web, Ltd, Dr.Web Anti-virus Light (free) 7.00.3</td>
<td>not a virus Adware.Startapp.origin.5</td>
<td>08/05/13</td>
<td>Full report</td>
</tr>
<tr>
<td>Android 4.1.2</td>
<td>Trend Micro, Mobile Security &amp; Antivirus 2.6.2</td>
<td>AndroidOS_FakeInst.VTD</td>
<td>08/05/13</td>
<td>Full report</td>
</tr>
<tr>
<td>Android 4.1.2</td>
<td>AVAST Software, avast! Mobile Security 2.0.3917</td>
<td>(Android:FakeInst-EO [PUP]).</td>
<td>08/05/13</td>
<td>Full report</td>
</tr>
<tr>
<td>Android 4.1.2</td>
<td>Kaspersky Lab, Kaspersky Mobile Security Lite 9.36.28</td>
<td>No threat detected</td>
<td>08/05/13</td>
<td>Full report</td>
</tr>
<tr>
<td>Android 4.1.2</td>
<td>NortonMobile, Norton Security &amp; Antivirus 3.3.4.970</td>
<td>No threat detected</td>
<td>08/05/13</td>
<td>Full report</td>
</tr>
<tr>
<td>Task id</td>
<td>131bd4fe-3bcd-4a72-a207-683ed8eb79f1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor name</td>
<td>Trend Micro</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antivirus name</td>
<td>Mobile Security &amp; Antivirus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engine version</td>
<td>2.6.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis started on</td>
<td>08/05/2013 at 17:05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis completed on</td>
<td>08/05/2013 at 17:07 (took 91 seconds)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detection method</td>
<td>On install</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis result</td>
<td>AndroidOS_FakeInst.VTD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sample md5</td>
<td>cbdf63b2e5666799c4b74a8cd15565dd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Logcat dump

<table>
<thead>
<tr>
<th>Line</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>I/tmms-vsapi-jni(674): VSReadVirusPattern OK. Action successful.</td>
</tr>
<tr>
<td>100</td>
<td>I/tmms-vsapi-jni(674): OK. VSSetProcessAllFileInArcFlag. oldValue = ret = 0.</td>
</tr>
<tr>
<td>101</td>
<td>I/tmms-vsapi-jni(674): OK. VSSetExpandLiteFlag. oldValue = ret = 1.</td>
</tr>
<tr>
<td>102</td>
<td>I/tmms-vsapi-jni(674): OK. VSSetProcessAllFileFlag. oldValue = ret = 0.</td>
</tr>
<tr>
<td>103</td>
<td>I/tmms-vsapi-jni(674): OK. VSSetCleanZipFlag. oldValue = ret = 0.</td>
</tr>
<tr>
<td>104</td>
<td>I/tmms-vsapi-jni(674): OK. VSSetCleanBackupFlag. oldValue = ret = 0.</td>
</tr>
<tr>
<td>105</td>
<td>I/tmms-vsapi-jni(674): VSGetDetectableVirusNumber virus in patter num = 3283</td>
</tr>
<tr>
<td>106</td>
<td>I/tmms-vsapi-jni(674): filename = /data/data/com.trendmicro.tmmspersonal/Library/pattern/msvpnaos.457</td>
</tr>
<tr>
<td>107</td>
<td>I/tmms-vsapi-jni(674): InternalVer = 145700, PtnVer = 457.</td>
</tr>
<tr>
<td>108</td>
<td>D/PrepareVSAPI4RTScan(674): before tmmsAntiMalwareJni4RTScan.init()</td>
</tr>
<tr>
<td>109</td>
<td>I/tmms-vsapi-jni(674): VSIInit OK!</td>
</tr>
<tr>
<td>110</td>
<td>D/PrepareVSAPI4RTScan(674): after tmmsAntiMalwareJni4RTScan.init()</td>
</tr>
<tr>
<td>111</td>
<td>I/tmms-vsapi-jni(674): in vsSetPatternPath, vc = 711579352</td>
</tr>
<tr>
<td>112</td>
<td>I/tmms-vsapi-jni(674): Current pattern path is: /etc/iscan</td>
</tr>
<tr>
<td>113</td>
<td>I/tmms-vsapi-jni(674): Pattern path is set to: /data/data/com.trendmicro.tmmspersonal/Library/pattern</td>
</tr>
<tr>
<td>114</td>
<td>I/tmms-vsapi-jni(674): Pattern file(s) successfully deleted.</td>
</tr>
<tr>
<td>115</td>
<td>I/tmms-vsapi-jni(674): in vsLoadPattern, vc = 711579352, sharedVC = 708085592, scanType =</td>
</tr>
<tr>
<td>116</td>
<td>I/tmms-vsapi-jni(674): vsLoadPattern patternPath = /data/data/com.trendmicro.tmmspersonal/Library/pattern.</td>
</tr>
</tbody>
</table>
### Antivirus Name

<table>
<thead>
<tr>
<th>Antivirus Name</th>
<th>Antivirus Version</th>
<th>Android Platform</th>
<th>Detection Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAST Software, avast! Mobile Security</td>
<td>2.0.3917</td>
<td>Android 4.1.2</td>
<td>On install</td>
</tr>
</tbody>
</table>

### Indirections

- Alignment
- ArithmeticBranch
- Debug
- Defunct
- Goto
- Indirections
- Nop
- Rebuild
- Reflection
- Renaming
- Reordering
- Repacking
- Resigned
- StringEncrypt

**Start scan!**

By clicking “Start scan!”, you agree to our Terms of Service and our Privacy Policy.
Writing tests is was tedious

We have abstracted away the low level details, so that we can focus on the important things: extracting the results.
class TestSuite(base.BaseTestSuite):
    def on_install_detection(self, sample_path):
        self.pilot.install_package(sample_path)

        if self.pilot.wait_for_activity("com.zoner.android.antivirus_common.ActScanResults", 10):
            result = self.pilot.get_view_by_id("scaninfected_row_virus")
        else:
            result = False
```python
#...
def on_demand_detection(self, sample_path):
    self.pilot.install_package(sample_path)
    self.pilot.start_activity("com.zoner.android.antivirus", ".ActMain")
    self.pilot.wait_for_activity("com.zoner.android.antivirus.ActMain")

    self.pilot.tap_on_coordinates(120, 130)
    self.pilot.wait_for_activity("com.zoner.android.antivirus.ActMalware")

# start scan
self.pilot.tap_on_coordinates(120, 80)
self.pilot.wait_for_activity(
    "com.zoner.android.antivirus_common.ActScanResults")

self.pilot.refresdsh()
#...
```
Tap on 'Antivirus' to scan device or files.

Tap on 'Scan device' to scan all installed applications.

Event waiting for scanning.

Screen scraping alert for 'Trojan.AndroidOS.KungFu.A' with options to Ignore or Remove.
WORKFLOW

1. Retrieve a suspicious APK
2. Choose parameters
   - Android version(s)
   - List of antimalware product and versions
   - Apply chain of mutations
3. Pull clean image(s) from repository
4. Instantiate one test per combination of
   - Android version
   - Product version
5. Enqueue test instances
ARCHITECTURE

- Web frontend
- Repository of clean Android images
- Asynchronous task dispatcher
- Distributed workers
REST/JSON API and Client

- Push (public) and pull (invite only) samples
- Python client: [https://bitbucket.org/andrototal/tools](https://bitbucket.org/andrototal/tools)

```
$ python andrototal_cli.py -l DEBUG scan -at-key <...> -ms-key <...> path/to

Running command: scan
Uploading file sample.apk
Scan response: {
  "resource": "10a6f3efc8bc40c1922facde7d055208"
}
Uploading file sample2.apk
Scan response: {
  "resource": "e870c6748ca3409f84c9c9e1a91daf3f"
}
Uploading file 40156a176bb4554853f767bb6647fd0ac1925eac.apk
Scan response: {
  "resource": "21d6c7234a184db6b8e52f2bab523787"
}
Uploading file samples-3.apk
Scan response: {
  "resource": "ec5b3c94ed624d6993b52a50d63153fa"
}
```
SCALABILITY

The graph illustrates the throughput (tests per minute) as a function of the number of active workers. Two scenarios are compared:

- **AndroTotal scalability (real)**: This line represents the actual scalability of the system in real-world conditions.
- **Linear scalability (ideal)**: This line shows the ideal linear scalability, which is not achieved in reality.

The graph shows:

- **1 machine (dual-core)**: The throughput increases with the number of active workers, but not linearly.
- **2 machines (dual-core + quad-core)**: Here, the throughput also increases with the number of active workers, but the system is more scalable due to the additional cores.

The graph highlights the gap between theoretical (linear) and practical scalability, especially as the number of active workers increases.
1. THREATS AND PROTECTIONS
2. LIMITATIONS
3. TESTING ANTIMALWARE
4. ANDROTOTAL
5. STATUS
NUMBERS

- 1,275 users subscribed
- 13 antimalware vendors supported (not all public)
- 16 products overall (not all public)
- 23,215 distinct APKs submitted and analyzed
SUPPORTED APPS (PUBLIC)

- ZONER, Inc. - Zoner AntiVirus Free 1.8.0
- ZONER, Inc. - Zoner AntiVirus Free 1.7.6
- AVAST Software - avast! Mobile Security 2.0.3917
- Doctor Web, Ltd - Dr.Web Anti-virus Light (free) 7.00.3
- Kaspersky Lab - Kaspersky Mobile Security Lite 9.36.28
- Kaspersky Lab - Kaspersky Mobile Security 10.4.41
- Trend Micro - Mobile Security & Antivirus 2.6.2
- Trend Micro - Mobile Security & Antivirus 3.1
- NortonMobile - Norton Security & Antivirus 3.2.0.769
- NortonMobile - Norton Security & Antivirus 3.3.4.970
<table>
<thead>
<tr>
<th>Label</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDS:DangerousObject.Multi.Generic</td>
<td>3963</td>
</tr>
<tr>
<td>HEUR:Trojan-SMS.AndroidOS.Opfake.bo</td>
<td>1252</td>
</tr>
<tr>
<td>not a virus Adware.Airpush.origin.7</td>
<td>701</td>
</tr>
<tr>
<td>AndroidOS Opfake.CTD</td>
<td>700</td>
</tr>
<tr>
<td>HEUR:Trojan-SMS.AndroidOS.Opfake.a</td>
<td>628</td>
</tr>
<tr>
<td>Android.SmsSend.origin.281</td>
<td>620</td>
</tr>
<tr>
<td>HEUR:Trojan-SMS.AndroidOS.FakeInst.a</td>
<td>512</td>
</tr>
<tr>
<td>Android.SmsSend.origin.315</td>
<td>485</td>
</tr>
<tr>
<td>HEUR:Backdoor.AndroidOS.KungFu.a</td>
<td>466</td>
</tr>
<tr>
<td>Android.SmsSend.origin.585</td>
<td>462</td>
</tr>
<tr>
<td>Android.SmsSend.origin.629</td>
<td>461</td>
</tr>
<tr>
<td>Adware.AndroidOS.Airpush-Gen</td>
<td>432</td>
</tr>
<tr>
<td>HEUR:Backdoor.AndroidOS.BaseBrid.a</td>
<td>390</td>
</tr>
<tr>
<td>AndroidOS Opfake.CTC</td>
<td>386</td>
</tr>
</tbody>
</table>
FUTURE WORK

• Add more cores and scale
• Compare labels and detection results with VirusTotal.com
• Deploy on ARM boards and monitor power consumption
• Open malware repository and API: anyone interested?
GRAB A STICKER!

QUESTIONS?

http://andrototal.org
@andrototal_org
fede@maggi.cc