Systems Security Research at the Distributed Computing Systems Lab

Sotiris Ioannidis
FORTH
sotiris@ics.forth.gr

Who we are: People

- Created in 2004, O(25) people
  - Head
    - Evangelos Markatos, Ph.D. U. of Rochester, USA
  - Researchers/Associated Researchers
    - Elias Athanasopoulos, Ph.D. U. of Crete, Greece
    - Vivi Fragopoulou, Ph.D. Queen's U., Canada
    - George Kopidakis, Ph.D. U. of Iowa, USA
    - Sotiris Ioannidis, Ph.D. U. of Pennsylvania, USA
    - Michalis Polychronakis, Ph.D. U. of Crete, Greece
  - Students, PhD: 3, MSc + Ugrads: O(10)
  - Engineers: 4, Other staff: 2

What we do: Research

- Study planet-wide distributed systems
  - to understand the forces that drive their day-to-day operation
  - to master the dimensions that sustain their long-term evolution
- Example Questions:
  - Why do they work at all?
  - How do they break?
  - What kind of traffic flows through the “veins” of such systems?
  - What holds these systems together?
  - How do they respond to various types of attacks?
  - Under what circumstances would they collapse?
  - How can we make them more robust?
  - How can we trust them?
  - How can we be safe in them?

(Some funded) Research Projects

- SysSec
- iCode
- ForToo
- TRACER
- EUINCOOP
- PASS
- SAFELINE

Some of our recent Research Work

- Provenance
- High-speed IDS
- Privacy

Data Provenance

- Describe how an object came to be in its present state
  - e.g., search terms for a resulting webpage
  - or, queries(*create*, *insert*, *alter*, etc.) for database results
- Towards a Universal Data Provenance Framework using Dynamic Instrumentation
  - With: Eleni Gessiou, Vasilis Pappas, Elias Athanasopoulos, Angelos D. Keromytis

Motivation

- Many interesting data provenance scenarios
- **But** challenging to prototype in large systems

Application	SLOC (Million)	# files
Firefox 4	5	40,000
MySQL 5.5	1.2	3,000

- Even worse in proprietary systems!

Our approach

- Design a framework that:
  1. assist the user in discovering paths in the system that interesting data pass through
  2. the user can dynamically instrument these points to record provenance about this transit data

- Preferably, without requiring source code

Implementation

- Built on Dynamic Binary Instrumentation (DTrace)
  - available in modern OSes (e.g., Mac, Solaris)
  - no changes in the instrumented system
  - easily enabled or disabled, even at runtime
  - no requirement of having the source code
- Features:
  - Configurable Logging
  - Assisted Discovery

Configurable Logging

- System call logging
  - arguments, return value, user id, process name, process id and timestamp
- Enabled by default for all processes
  - Can be configured based on process id, program’s name and user
- Library function call logging
  - Specified by library and function name

Assisted Discovery

- Instrument all the functions and check for a specific value
  - given as input to the monitored system
- Log all the occurrences thought the system
  - The output forms an execution path
- Choose the best points in the path which meet the developer's needs

Case Study #1: File-System

- Points of interest - System call logging:
  - *creat*, *write*, *chmod*, *chown* and *unlink* system calls
- Output example:
  - 2012 Jan 7 23:44:52 TextEdit (23624) uid 501 open: .../paper.txt -> 15
  - 2012 Jan 7 23:44:52 TextEdit (23624) uid 501 write: 15, My paper -> 9
  - 2012 Jan 7 23:44:52 TextEdit (23624) uid 501 close: 15 -> 0
- No changes in the OS

Case Study #2: Database (SQLite)

- Goal: find appropriate point/function to log all queries
- Employ assisted discovery:
  - Perform specific queries of all types (create, insert, etc.)
  - Find the intersection of all paths
- Modified SQLite vs. binary instrumentation
  - Only **0.8%** overhead

Case Study #3: Web browser (Safari)

- Similar approach with file-system
- Although sufficient as proof of concept, revealed some limitations:
  - System call level instrumentation cannot be used when data are encoded, encrypted
  - Assisted discovery does not work with complex types:
    - E.g., class that represents URLs in Safari

Conclusion & Future Directions

- Designed a data provenance framework based on dynamic binary instrumentation
- Our framework covers classic data provenance applications

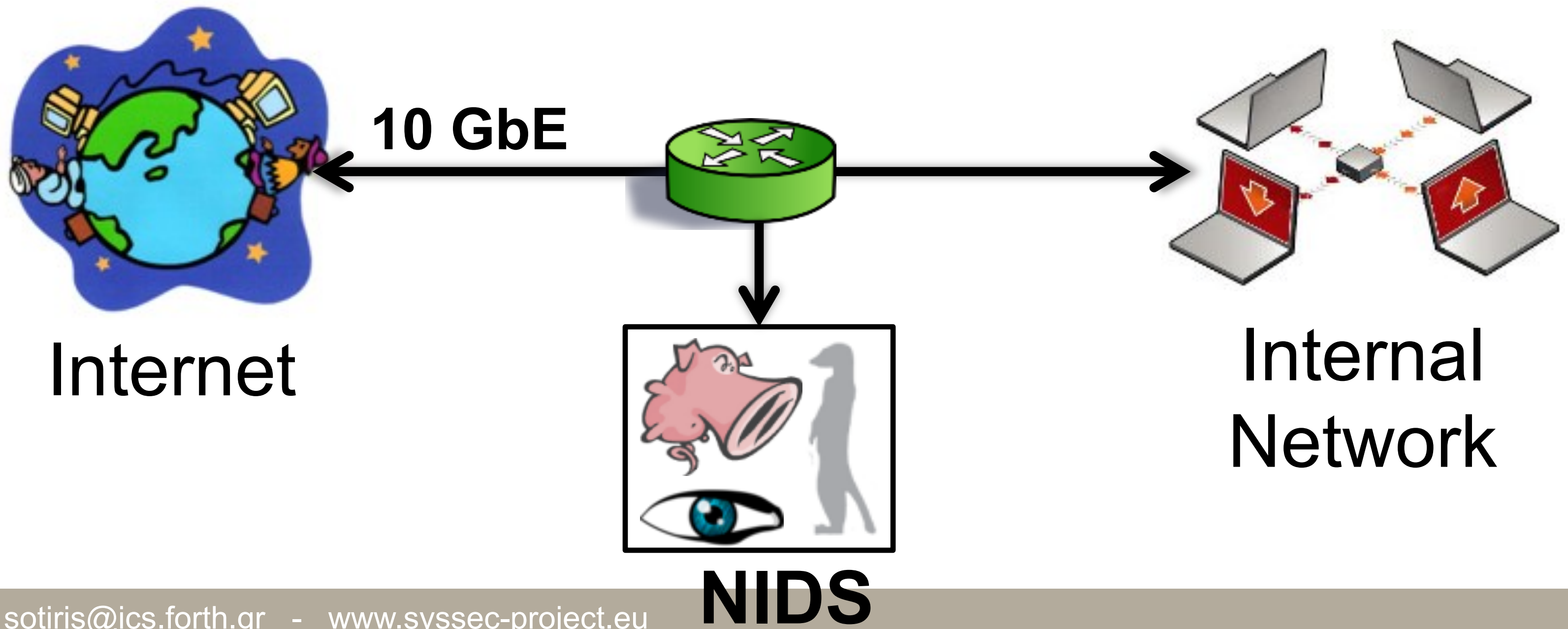
- Extend our framework (e.g., complex types)

High-speed IDS

- MIDeA a Multi-parallel Intrusion Detection Architecture
- With Giorgos Vasiliadis and Michalis Polychronakis

Network Intrusion Detection Systems

- Typically deployed at ingress/egress points
  - Inspect *all* network traffic
  - Look for suspicious activities
  - Alert on malicious actions



Challenges

- ***Traffic rates*** are increasing
  - 10 Gbit/s Ethernet speeds are common in metro/ enterprise networks
  - Up to 40 Gbit/s at the core

- Keep needing to perform ***more complex analysis*** at ***higher speeds***
  - Deep packet inspection
  - Stateful analysis
  - 1000s of attack signatures

Designing NIDS

- Fast
  - Need to handle many Gbit/s
  - Scalable
    - Moore's law is challenged

- Commodity hardware
  - Cheap
  - Easily programmable

Today: fast *or* commodity

- Fast "hardware" NIDS
  - FPGA/TCAM/ASIC based
  - Throughput: High

- Commodity "software" NIDS
  - Processing by general-purpose processors
  - Throughput: Low

MIDeA

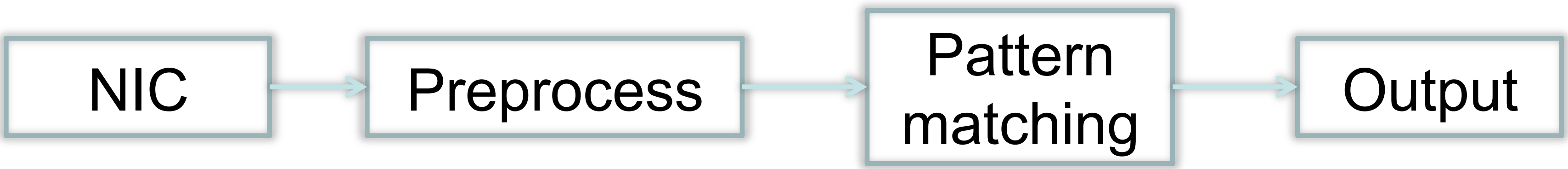
- A NIDS out of *commodity* components
  - Single-box implementation
  - Easy programmability
  - Low price

Can we build a 10 Gbit/s NIDS with commodity hardware?

Outline

- Architecture
- Implementation
- Performance Evaluation
- Conclusions

Single-threaded performance

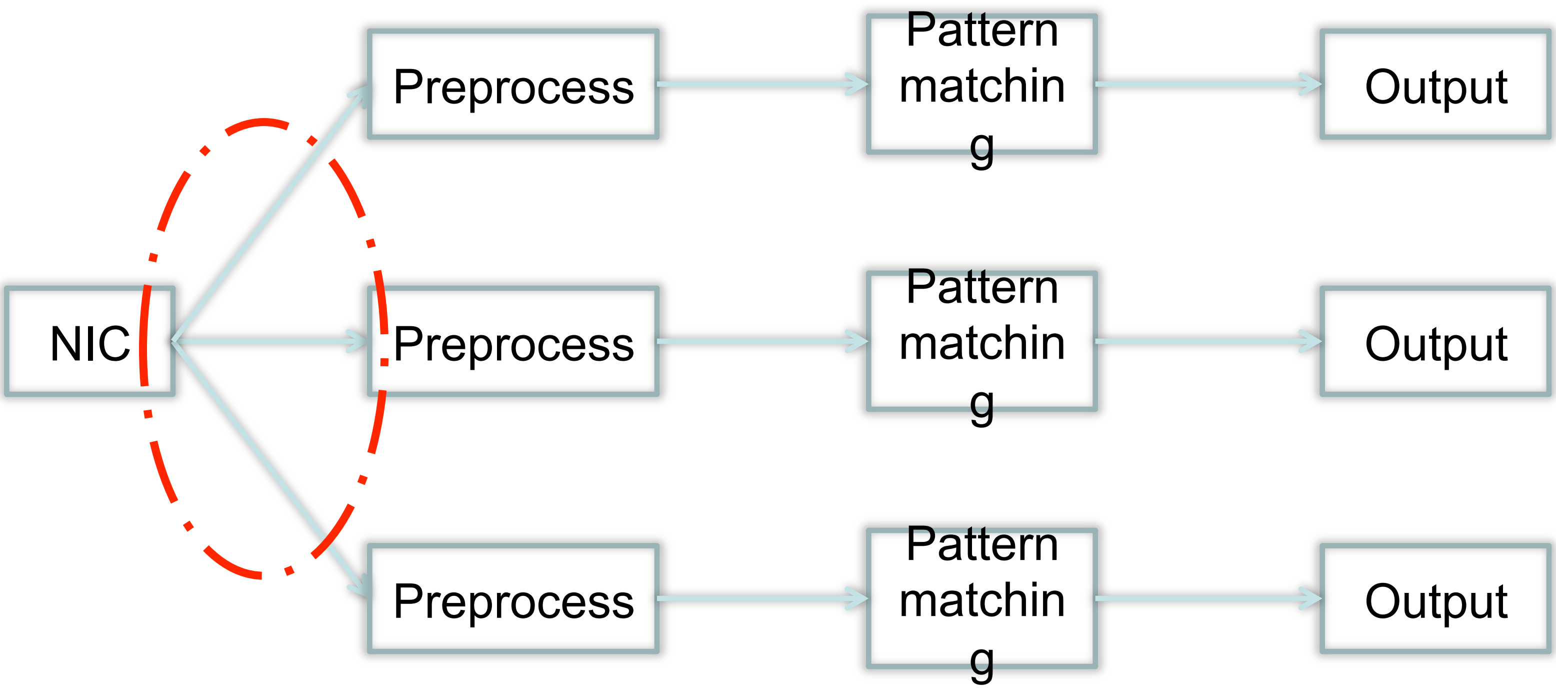


- **Vanilla Snort: 0.2 Gbit/s**

Problem #1: Scalability

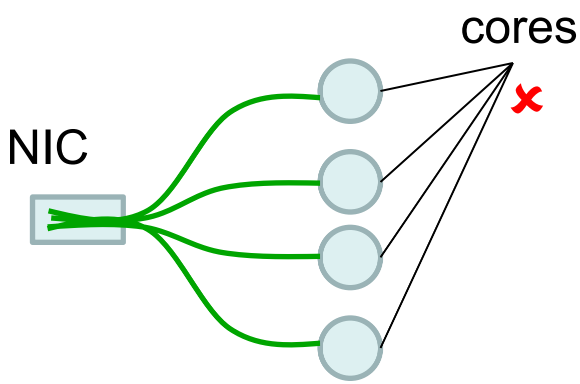
- Single-threaded NIDS have limited performance
  - Do not scale with the number of CPU cores

Multi-threaded performance

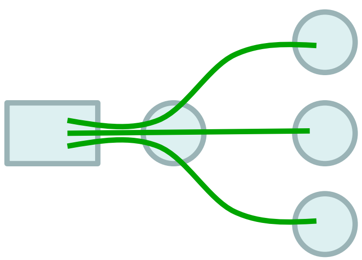


- Vanilla Snort: 0.2 Gbit/s
- **With multiple CPU-cores: 0.9 Gbit/s**

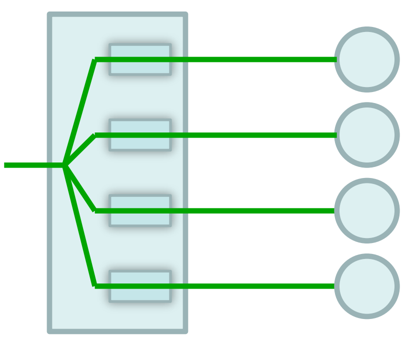
Problem #2: How to split traffic



- ✗ Synchronization overheads



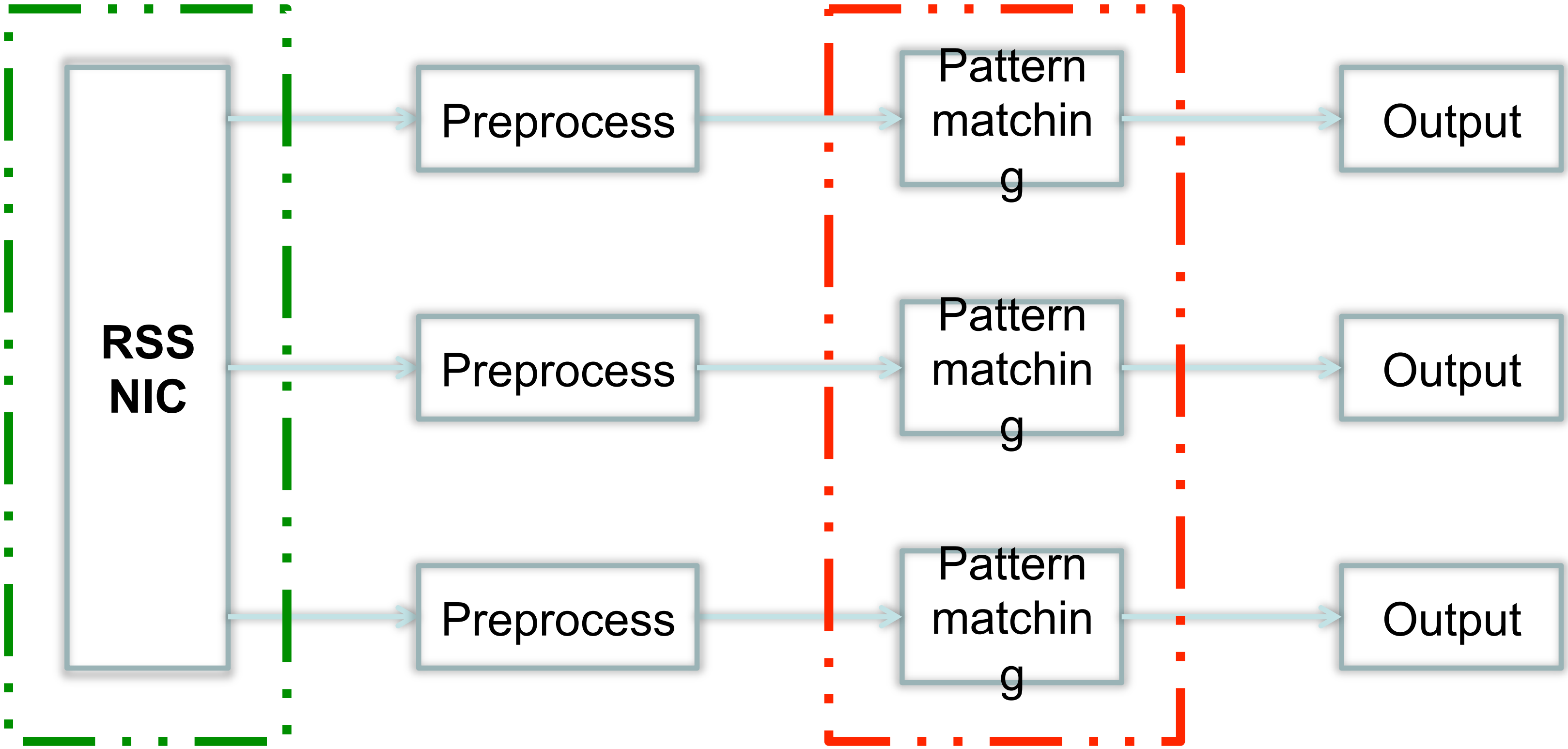
- ✗ Cache misses



- ✓ Receive-Side Scaling (RSS)

syssec

Multi-queue performance



- Vanilla Snort: 0.2 Gbit/s
- With multiple CPU-cores: 0.9 Gbit/s
- **With multiple Rx-queues: 1.1 Gbit/s**

Problem #3: Pattern matching is the bottleneck

> 75%

NIC → Preprocess → Pattern matching → Output

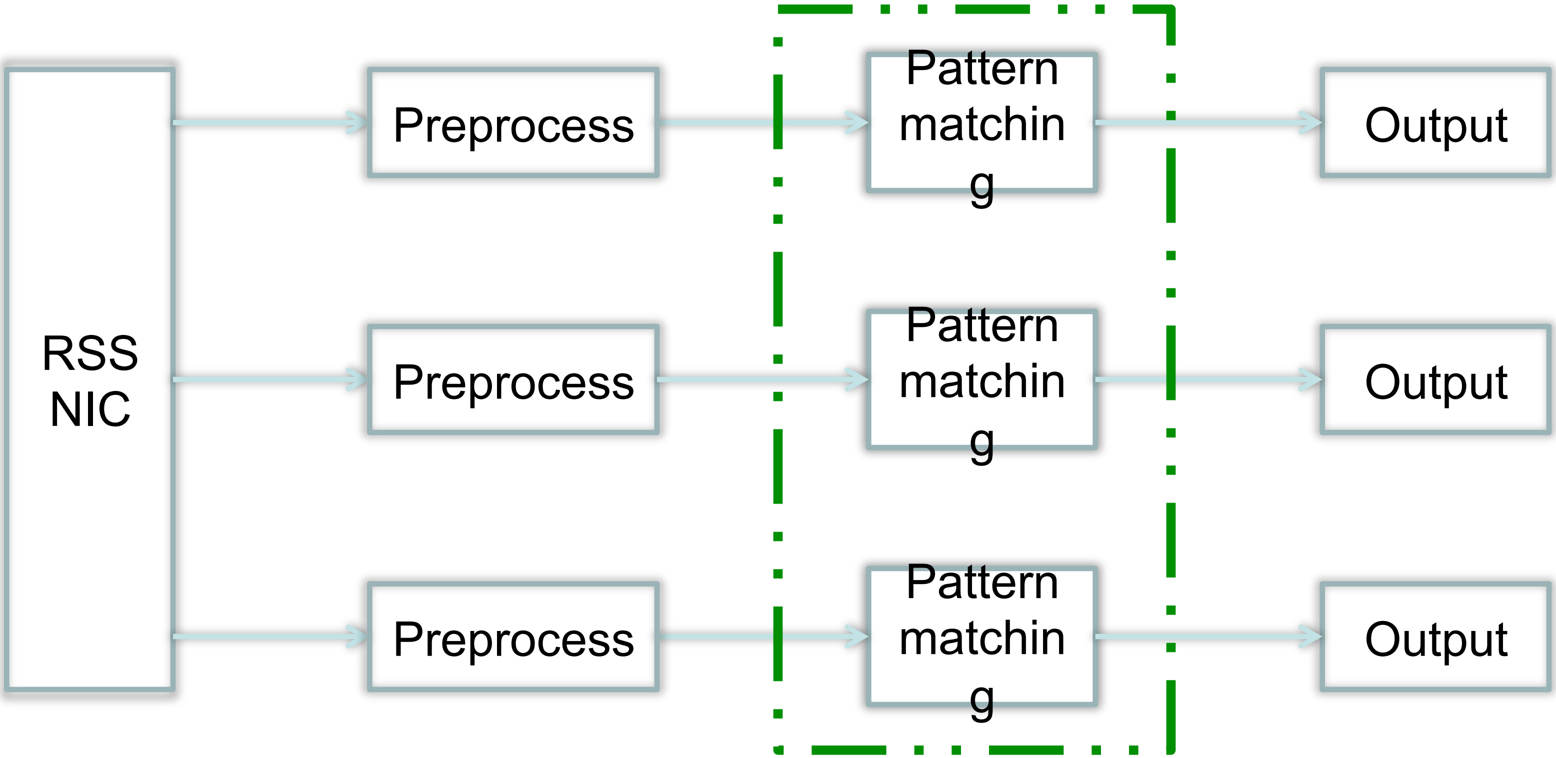
✓ Offload pattern matching on the GPU

NIC → Preprocess → Pattern matching → Output

Why GPU?

- General-purpose computing
  - Flexible and programmable
- Powerful and ubiquitous
  - Constant innovation
- Data-parallel model
  - More transistors for data processing rather than data caching and flow control

Offloading pattern matching to the GPU

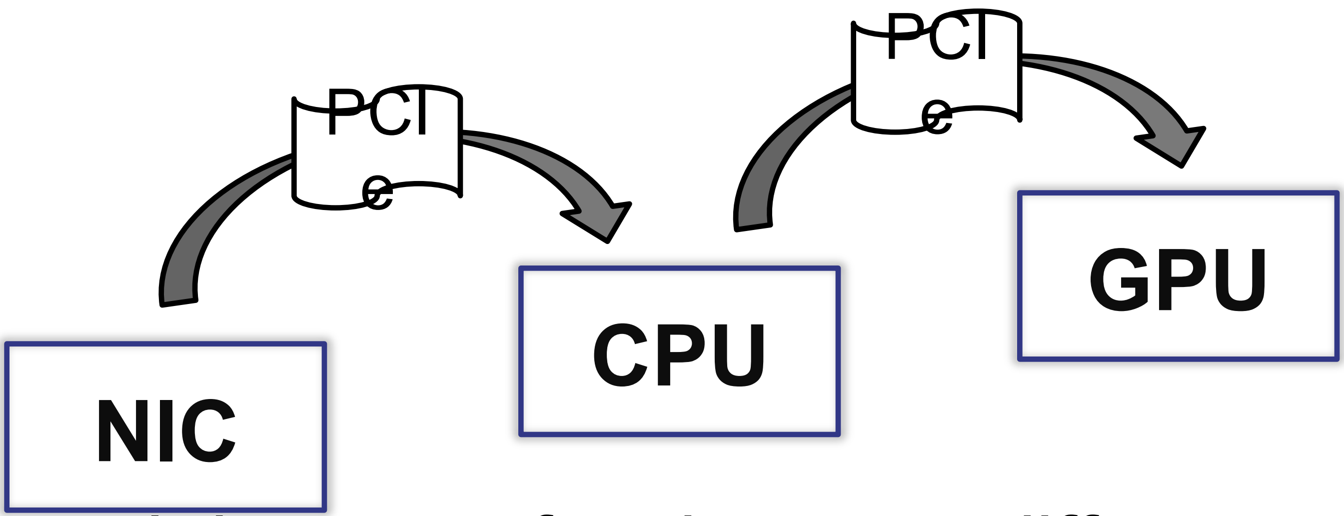


- Vanilla Snort: 0.2 Gbit/s
- With multiple CPU-cores: 0.9 Gbit/s
- With multiple Rx-queues: 1.1 Gbit/s
- **With GPU: 5.2 Gbit/s**

Outline

- Architecture
- **Implementation**
- Performance Evaluation
- Conclusions

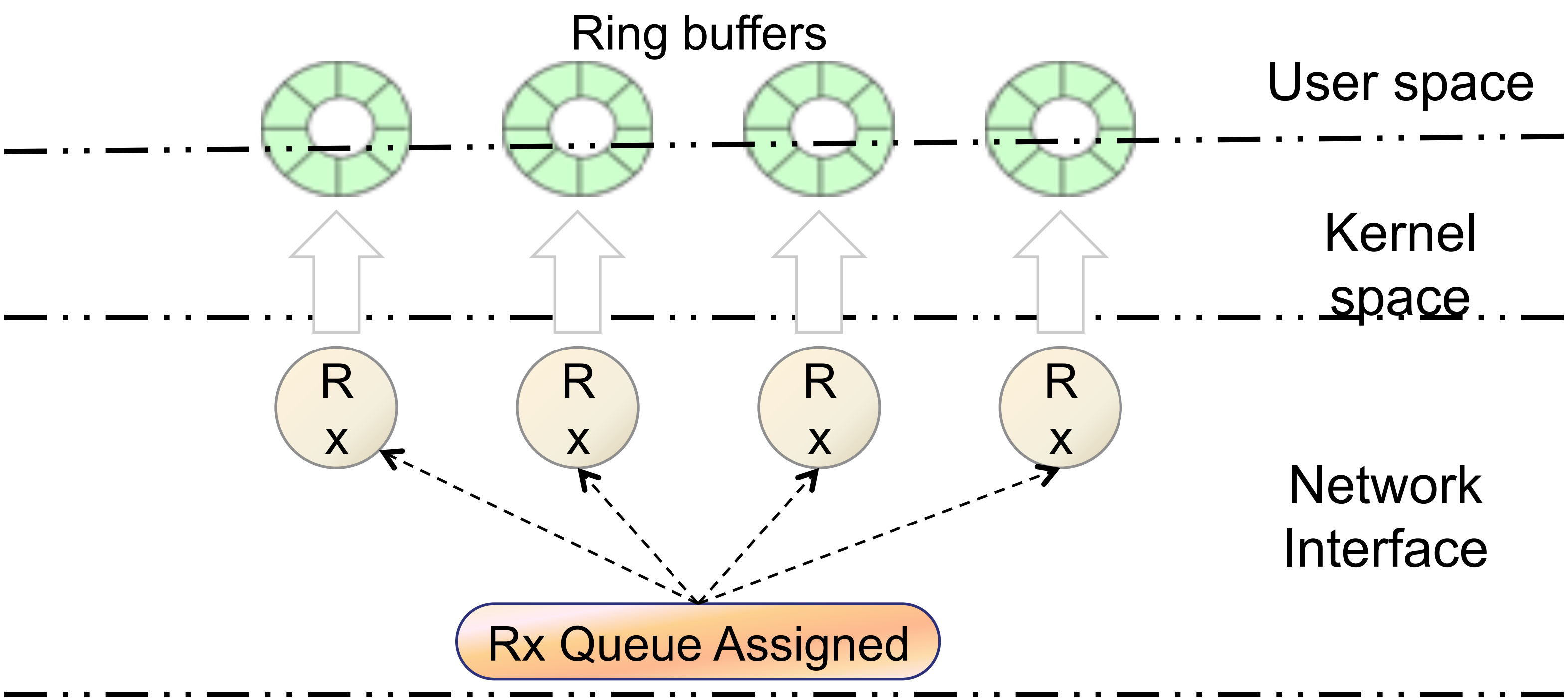
Multiple data transfers



- Several data transfers between different devices

Are the data transfers worth the computational gains offered?

Capturing packets from NIC



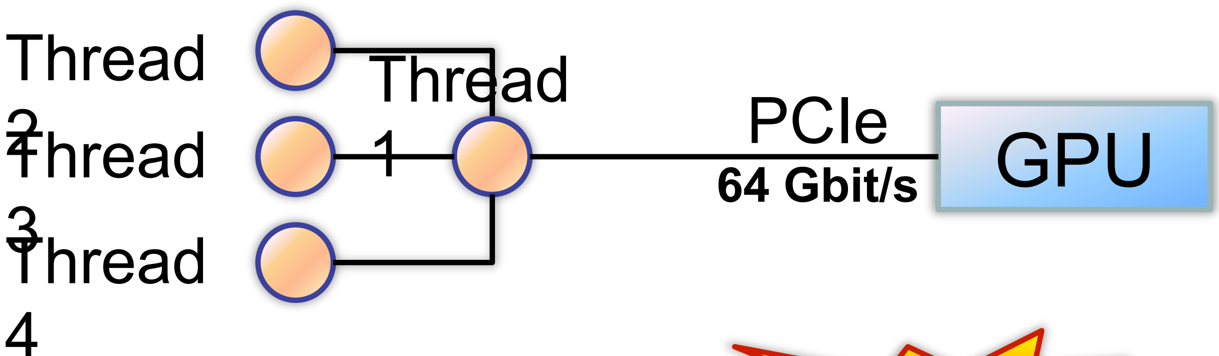
- Packets are hashed in the NIC and distributed to different Rx-queues
- Memory-mapped ring buffers for each Rx-queue

CPU Processing

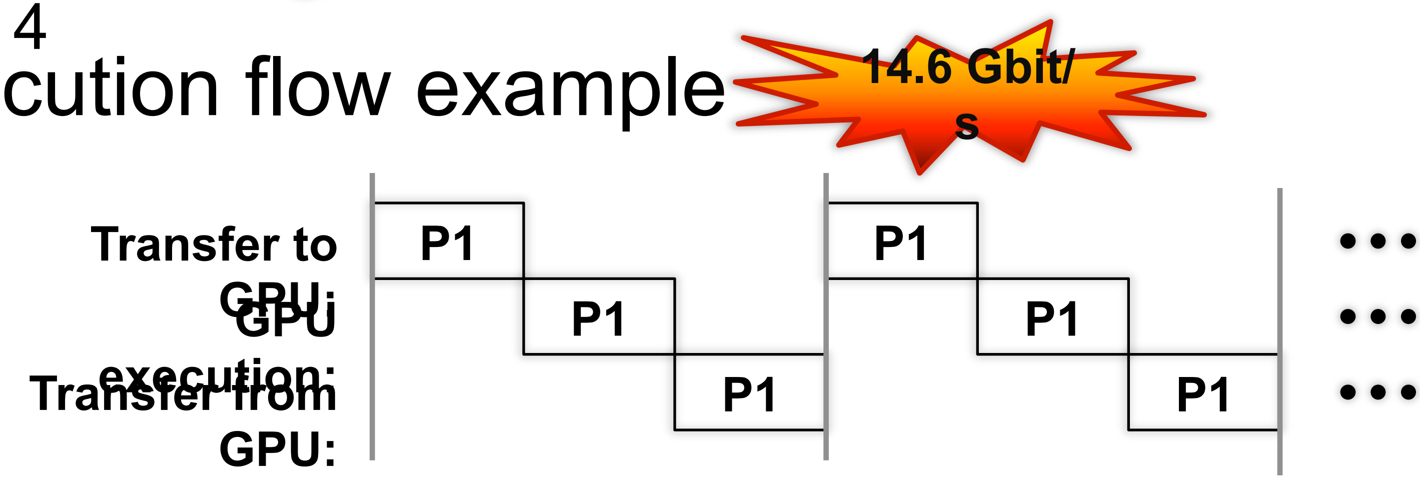
- Packet capturing is performed by different CPU-cores ***in parallel***
  - Process affinity
- Each core ***normalizes*** and ***reassembles*** captured packets to streams
  - Remove ambiguities
  - Detect attacks that span multiple packets
- Packets of the same connection *always* end up to the same core
  - No synchronization
  - Cache locality
- Reassembled packet streams are then ***transferred to the GPU*** for pattern matching
  - *How to access the GPU?*

Accessing the GPU

- Solution #1: Master/Slave model

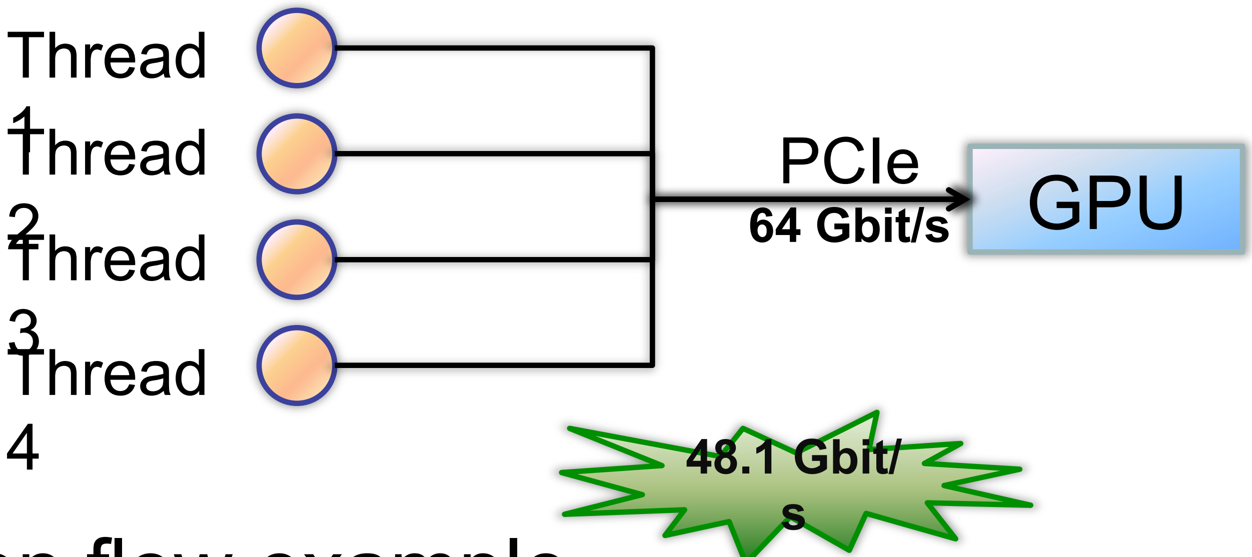


- Execution flow example

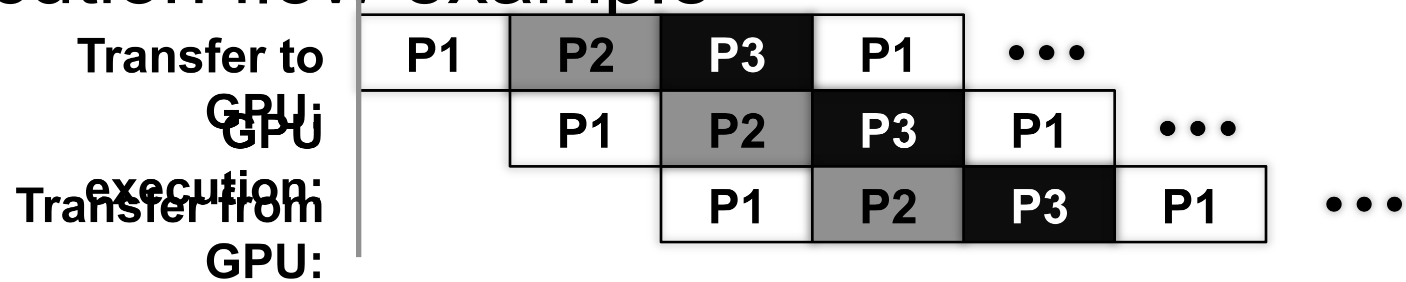


Accessing the GPU

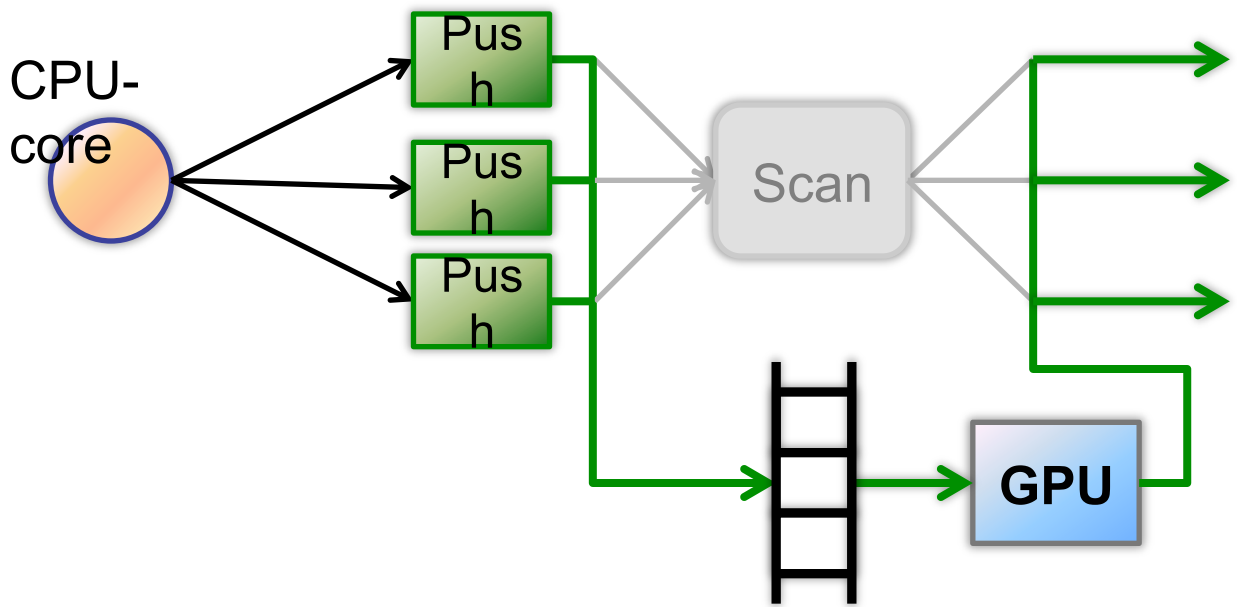
- Solution #2: Shared execution by multiple threads



- Execution flow example

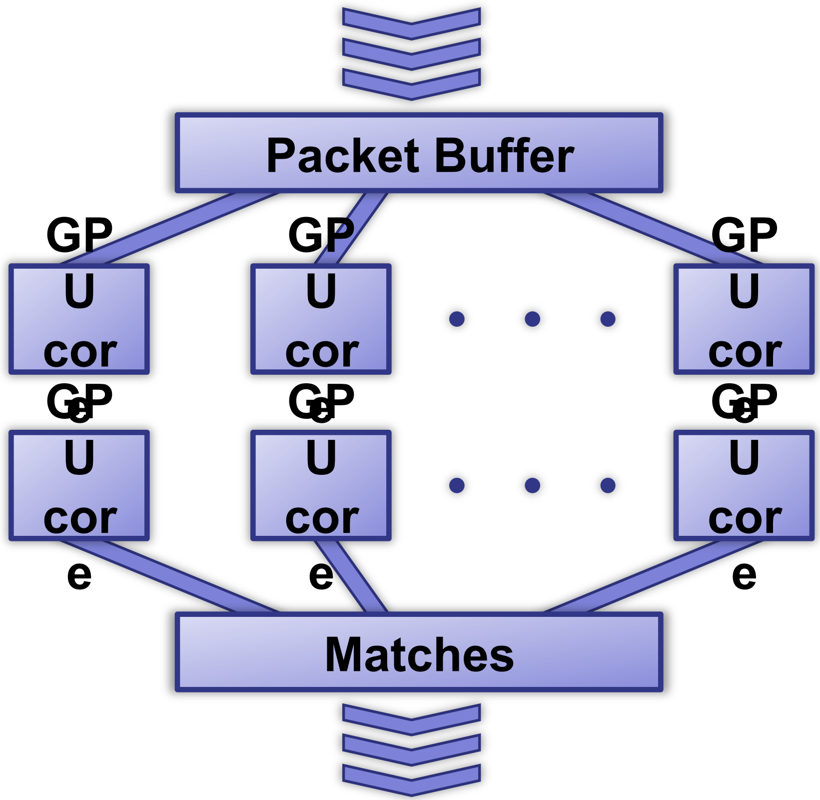


Transferring to GPU



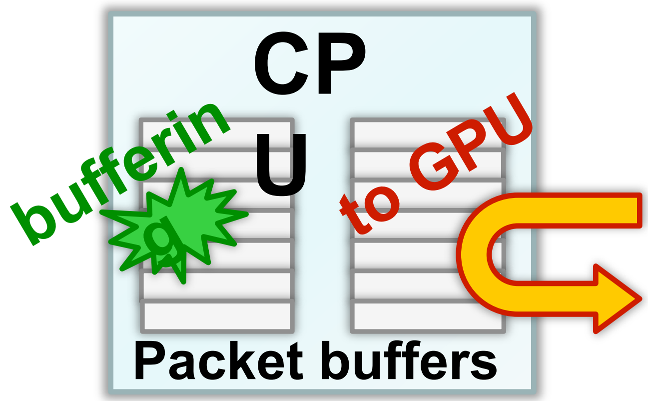
- Small transfer results to PCIe throughput degradation
  - ➔ Each core batches many reassembled packets into a single buffer

Pattern Matching on GPU



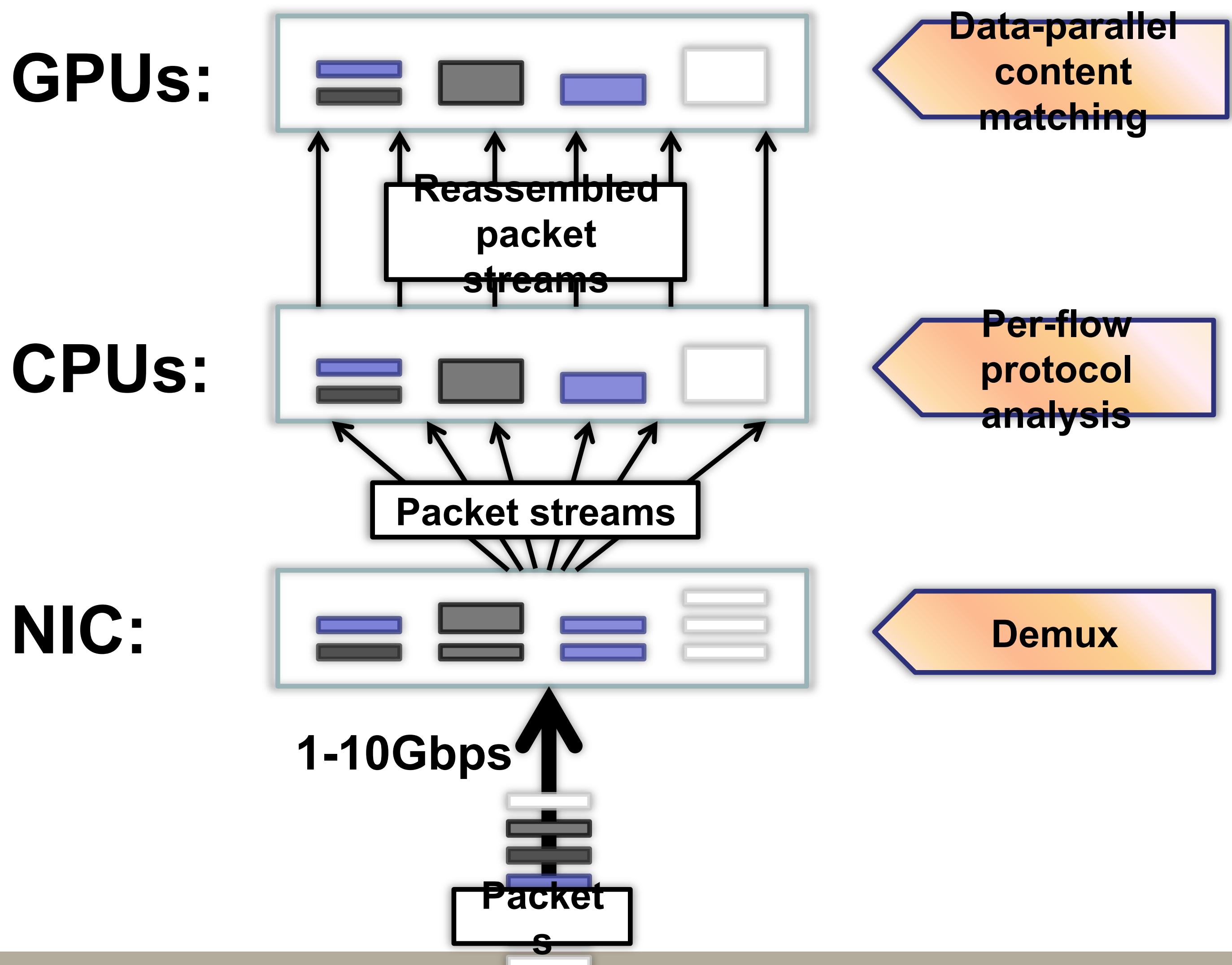
- Uniformly, *one GPU core for each* reassembled packet stream

Pipelining CPU and GPU



- Double-buffering
  - Each CPU core collects new reassembled packets, while the GPUs process the previous batch
  - Effectively hides GPU communication costs

Recap

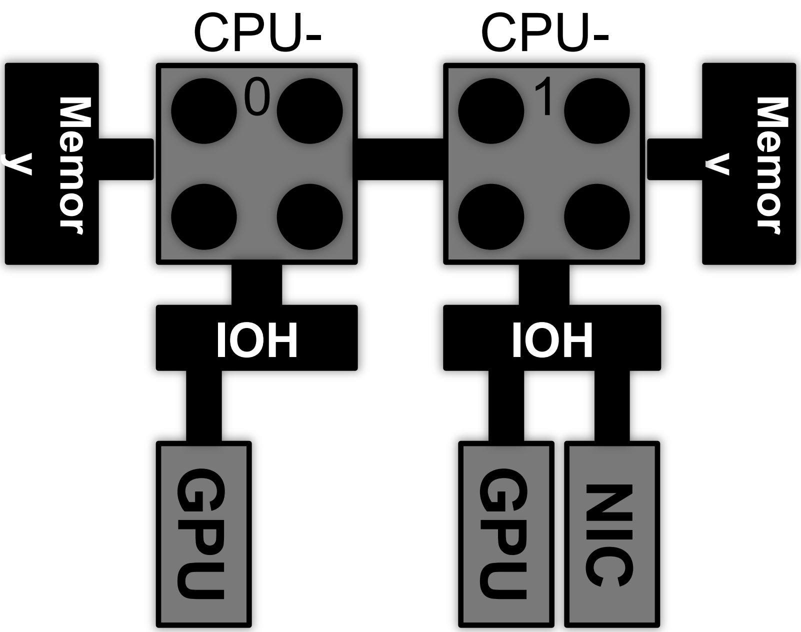


GPUs:
CPUs:
NIC:
Data-parallel content matching
Reassembled packet streams
Per-flow protocol analysis
Packet streams
Demux
1-10Gbps
Packets

Outline

- Architecture
- Implementation
- **Performance Evaluation**
- Conclusions

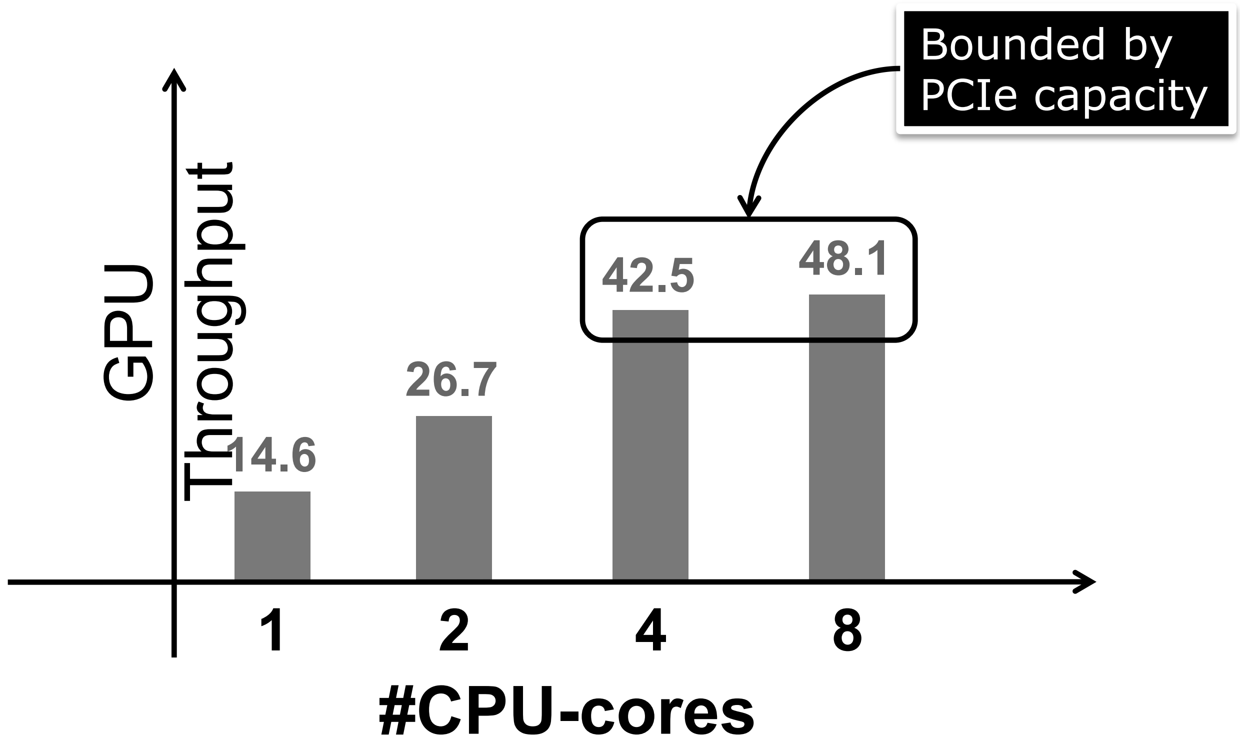
Setup: Hardware



- NUMA architecture, QuickPath Interconnect

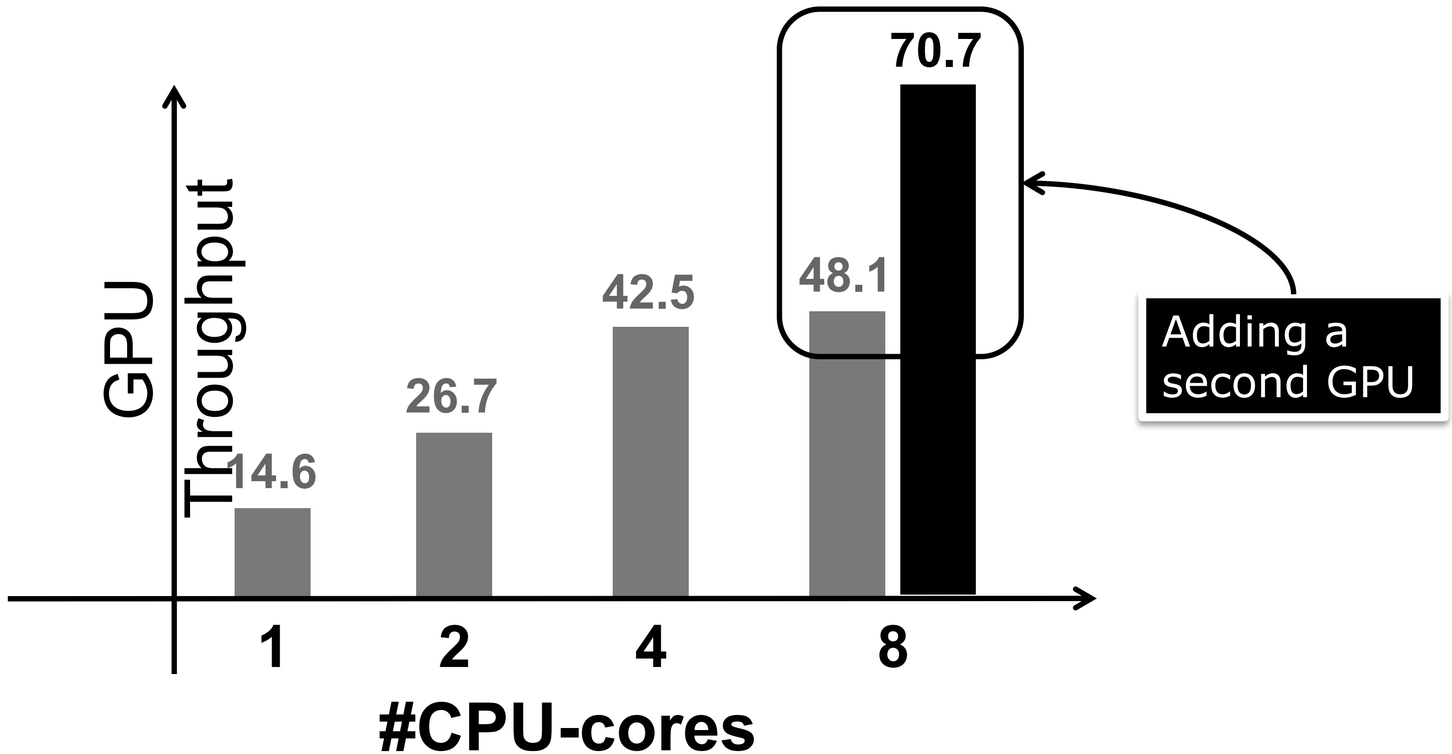
	Model	Specs
2 x CPU	Intel E5520	2.27 GHz x 4 cores
2 x GPU	NVIDIA GTX480	1.4 GHz x 480 cores

Pattern Matching Performance



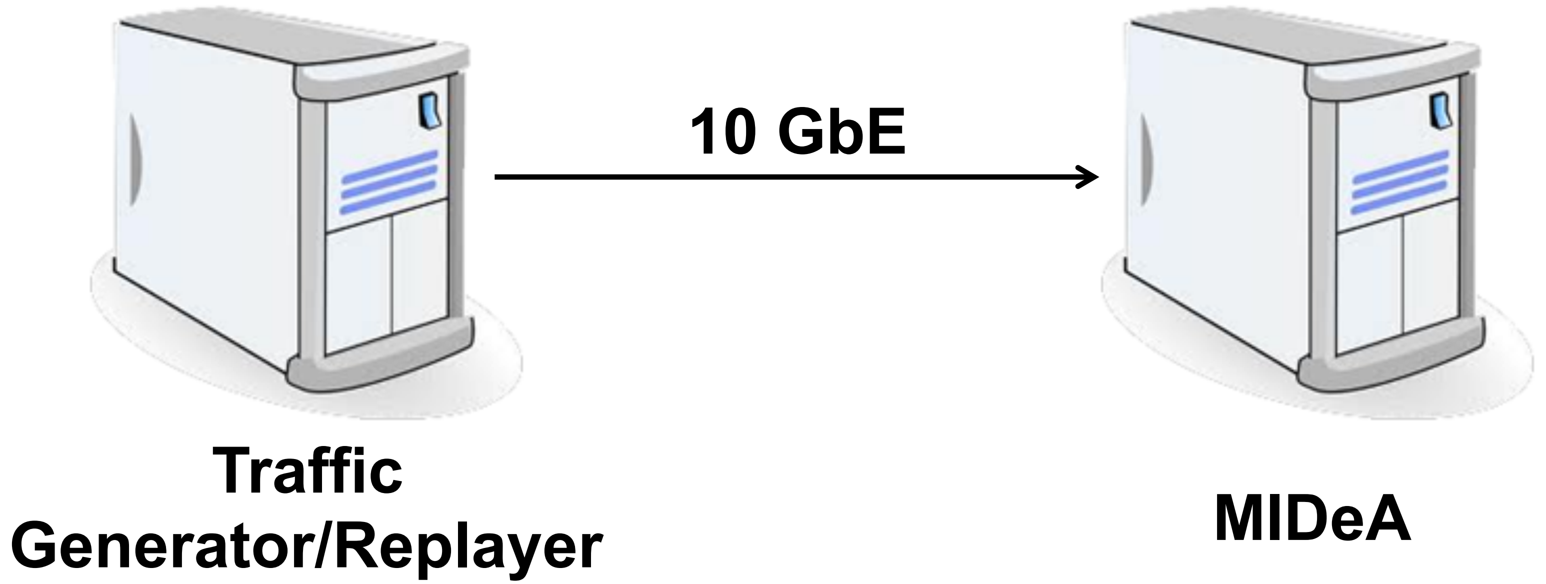
- The performance of a single GPU increases, as the number of CPU-cores increases

Pattern Matching Performance

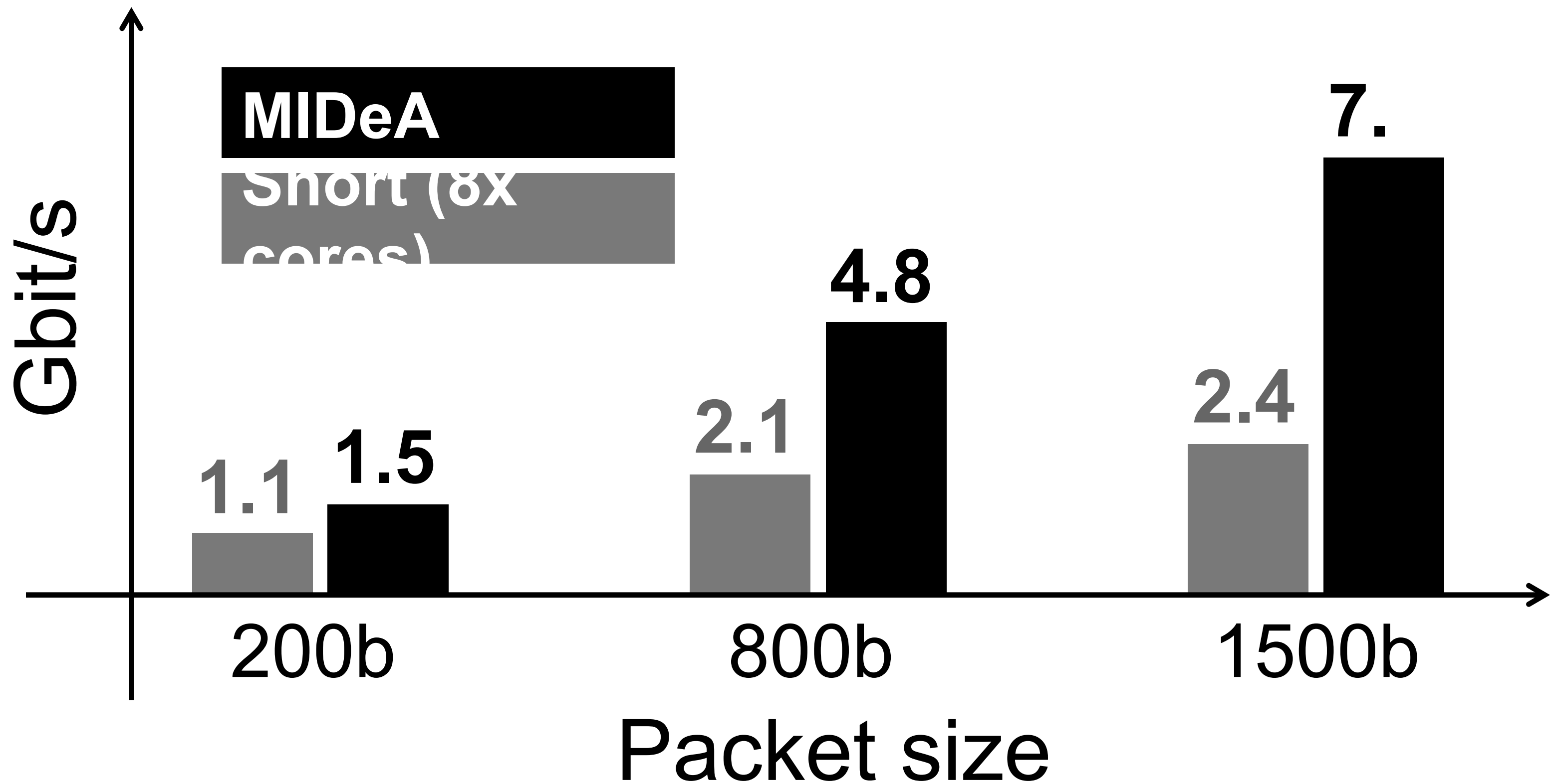


- The performance of a single GPU increases, as the number of CPU-cores increases

Setup: Network

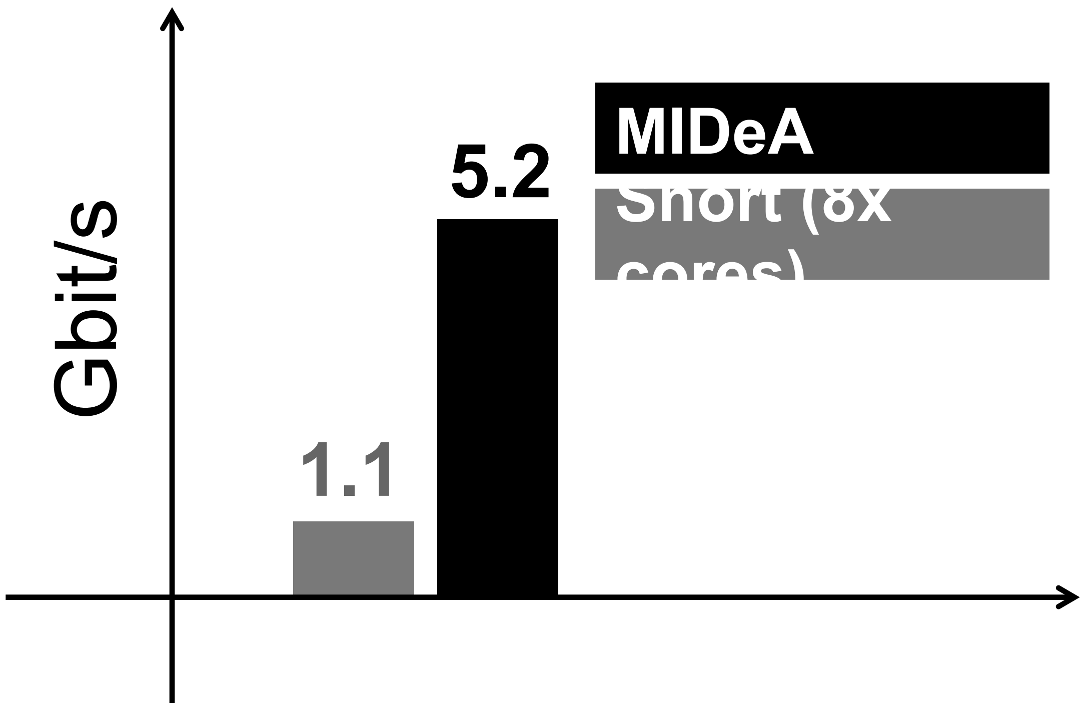


Synthetic traffic



- Randomly generated traffic

Real traffic

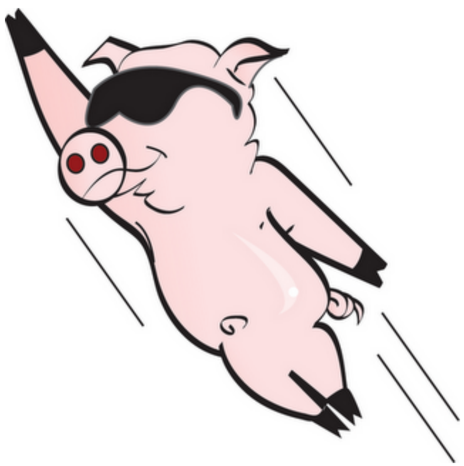


- **5.2 Gbit/s** with **zero** packet-loss
  - Replayed trace captured at the gateway of a university campus

syssec

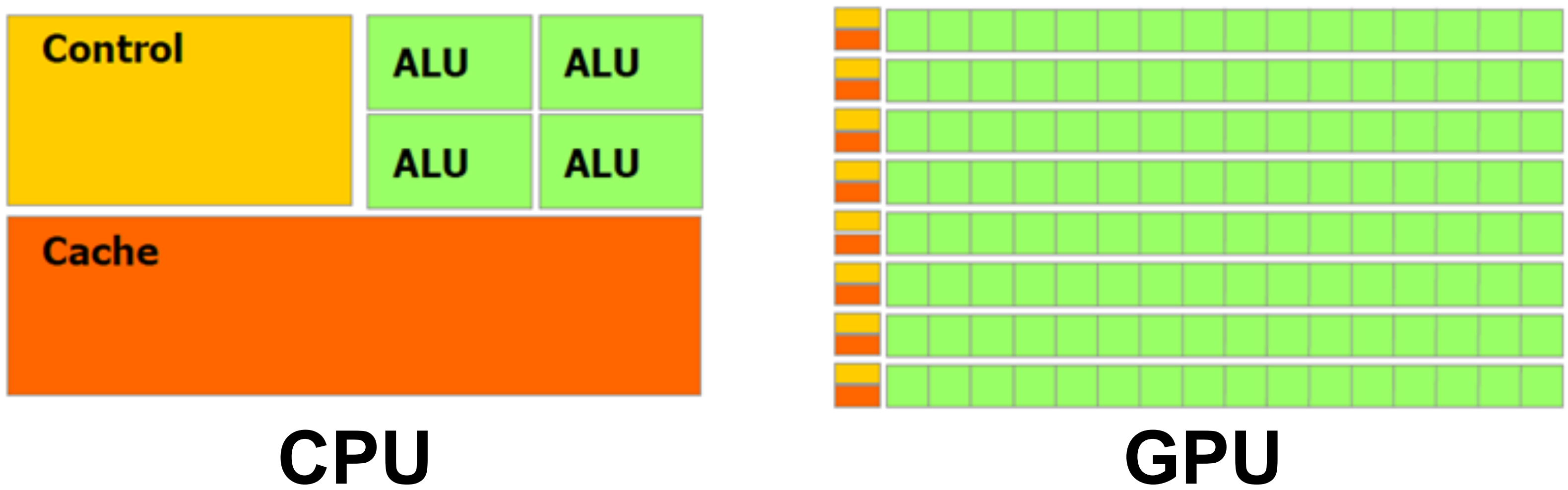
Summary

- MIDeA: A multi-parallel network intrusion detection architecture
  - **Single-box** implementation
  - Based on **commodity** hardware
  - Less than **$1500**



- Operate on **5.2 Gbit/s** with **zero packet loss**
  - **70 Gbit/s** pattern matching throughput

CPU vs GPU



- The GPU is specialized for compute-intensive, highly parallel computation
  - More transistors are devoted to data processing rather than data caching and flow control

Receive-Side Scaling (RSS)

- Modern NICs provide a small set of classification algorithms on the receive side
  - Address-based
  - Flow-based
  - Hash-based
- NIC controller classifies the packets and places them in one out of many queues
- Hardware queues are accessed independently
  - No need for synchronization

Harvesting Greek SSN

- **Study on the availability of Personal Identifiable Information in Greek Web sites**
- How easy it is to obtain someone's Greek SSN?
- Attack scenarios and possible solutions

Personal Identifiable Information

- Minimum:
  - First Name, Last Name
  - Father's First Name
  - Mother's First Name
- Additional:
  - Date of Birth
  - National ID#
  - Taxpayer ID#

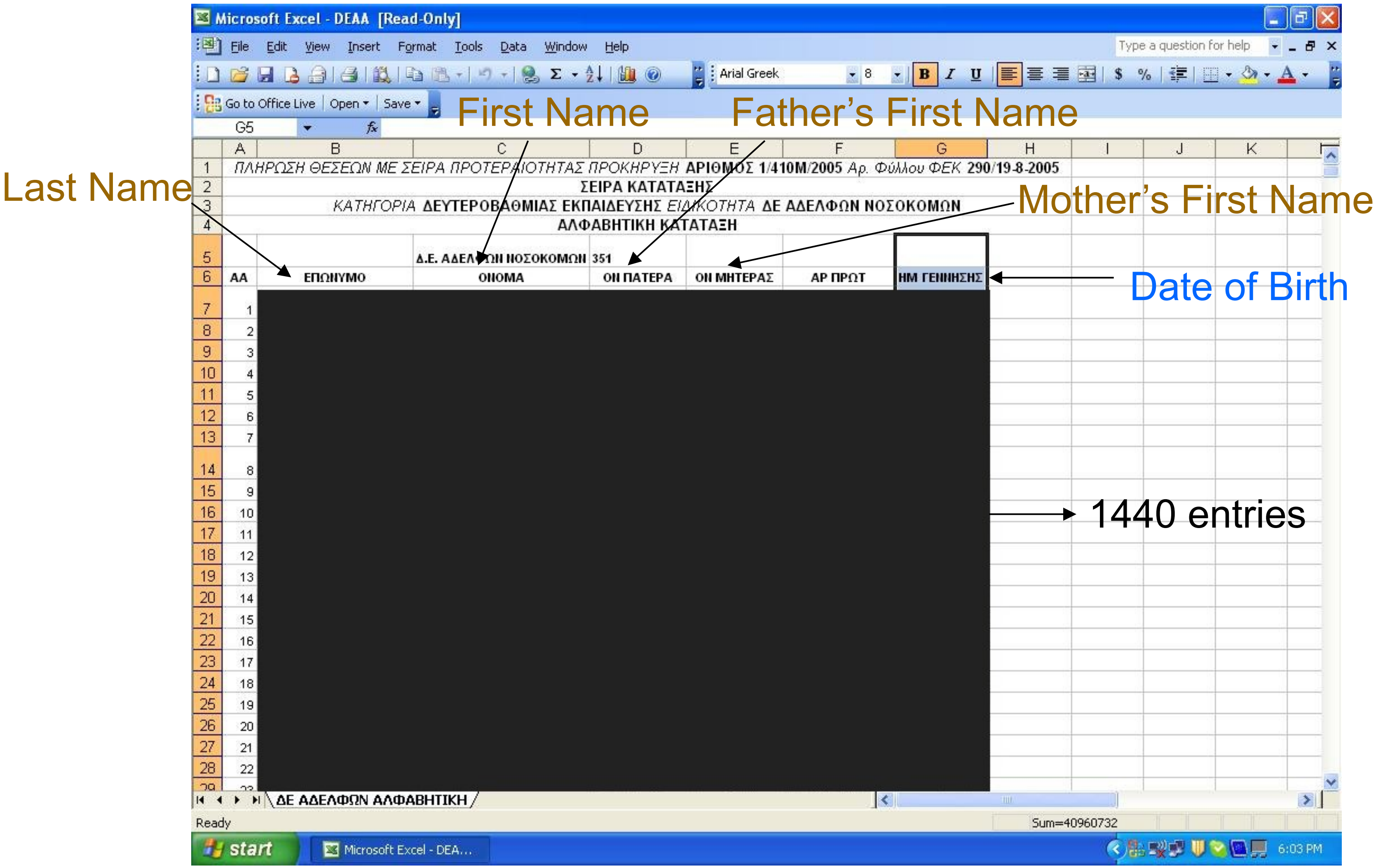
Searching for PII @ Greek Web

- We queried Google with all possible permutations of PII attribute names

  e.g. “filetype:xls site:.gr first_name last_name father’s_first_name date_of_birth”

- Focus on SpreadSheet format:
  - Primary source of information leaks
  - 3 .xls files

Example of a result
Spreadsheet document



Availability of PII

Full Name	Father's First Name	Mother's First Name	Date of Birth	ID#	Tax ID#	Total
✓	✓	✓	✓	✓	–	50
✓	✓	–	✓	✓	–	1,724
✓	✓	✓	✓	–	–	1,983
✓	✓	–	✓	–	–	3,843
✓	✓	✓	–	–	✓	4,244
✓	✓	✓	–	✓	–	4,895
✓	✓	–	–	–	✓	15,806
✓	✓	–	–	✓	–	22,099
✓	✓	✓	–	–	–	63,211

The introduction of SSN in Greece

- All Greek insurance agencies had their own registration numbers:
  - 23 different insurance agencies
- Greek SSN - **AMKA** to unify them
  - Proposed about 6 years ago
  - Mandatory as of October 2009

AMKA's format

Date of Birth

Odd for men and even for women

He was born on 30th of September in 1942

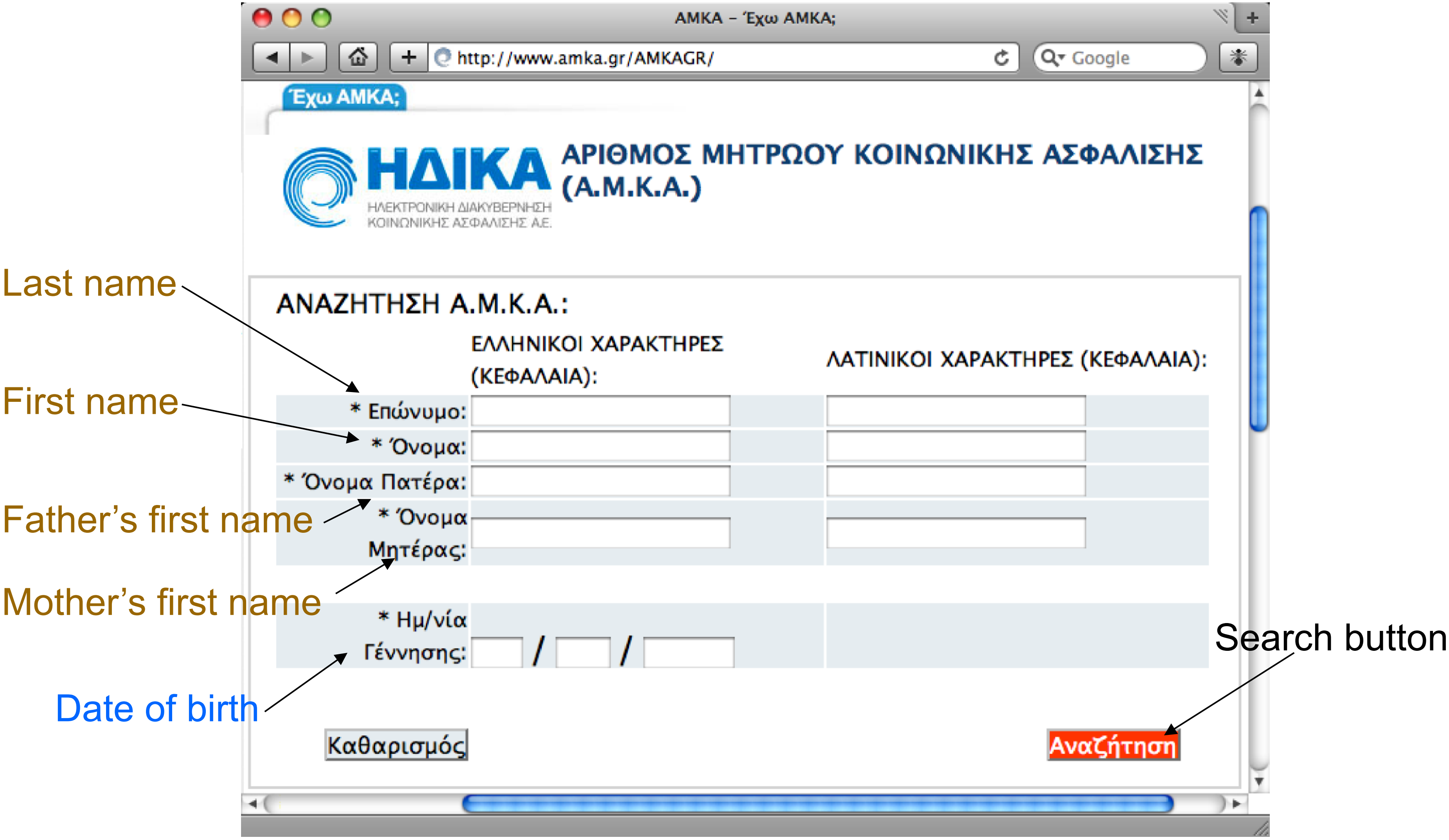
Control digit

30094201239

Sequence number

Male!

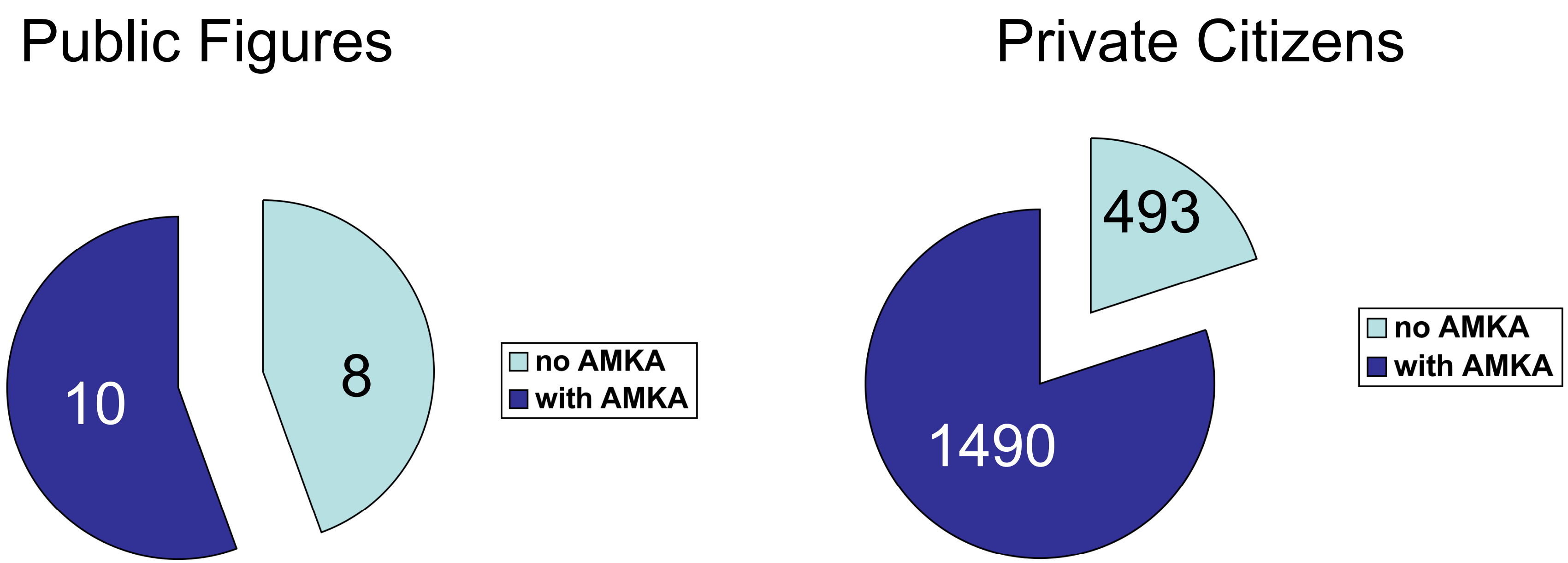
AMKA's Web site form



Finding AMKAs

- For public figures:
  - gather information from Wikipedia, personal sites, etc.
- For private citizens:
  - using the PII found on the Web. (previous table!..)
- We brute-forced the information we did not know

Sample Results
(data on the Web)



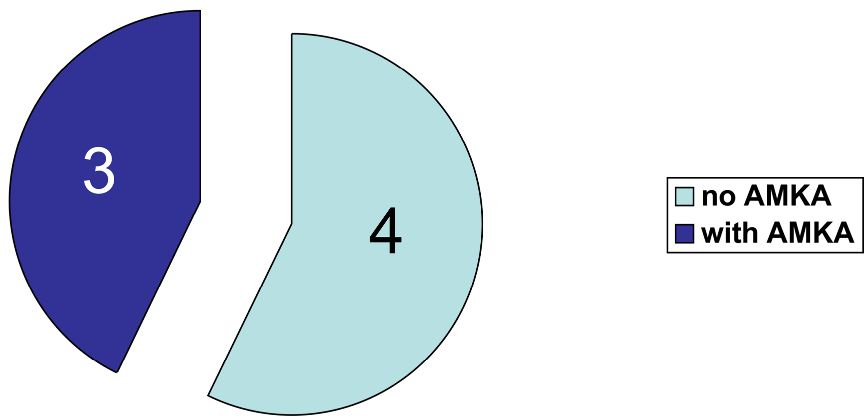
Sample Results
(data on the Web)

	Total	With AMKA	%
Public	**18**	**10**	**55.5**
Private	**1983**	**1490**	**75.1**

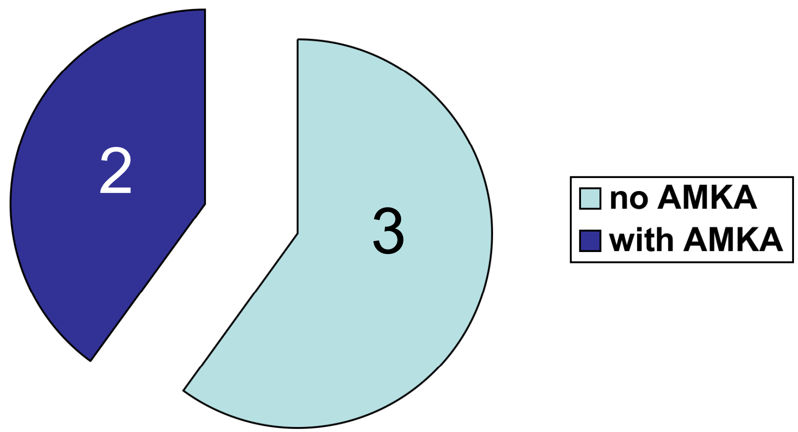
Sample Results
(data on the Web + brute-force search)

Public Figures

No Mother’s First Name

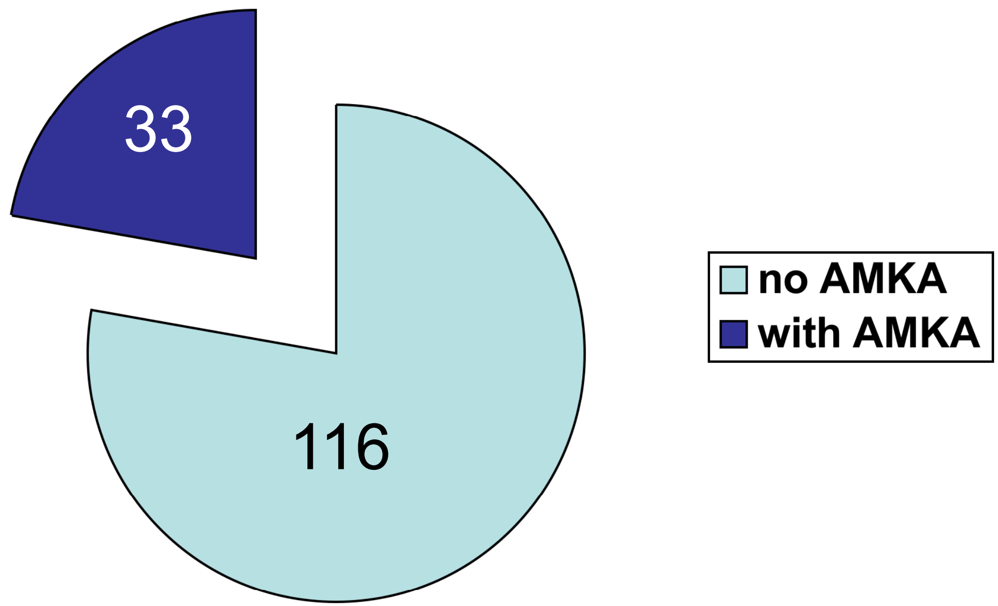


No Date of Birth



Sample Results
(data on the Web + brute-force search)

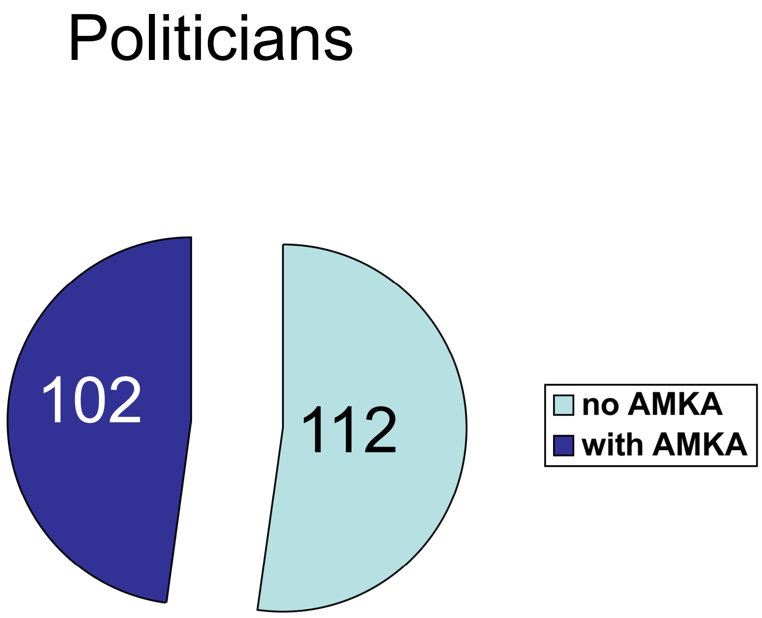
Private Citizens – No Mother’s First Name

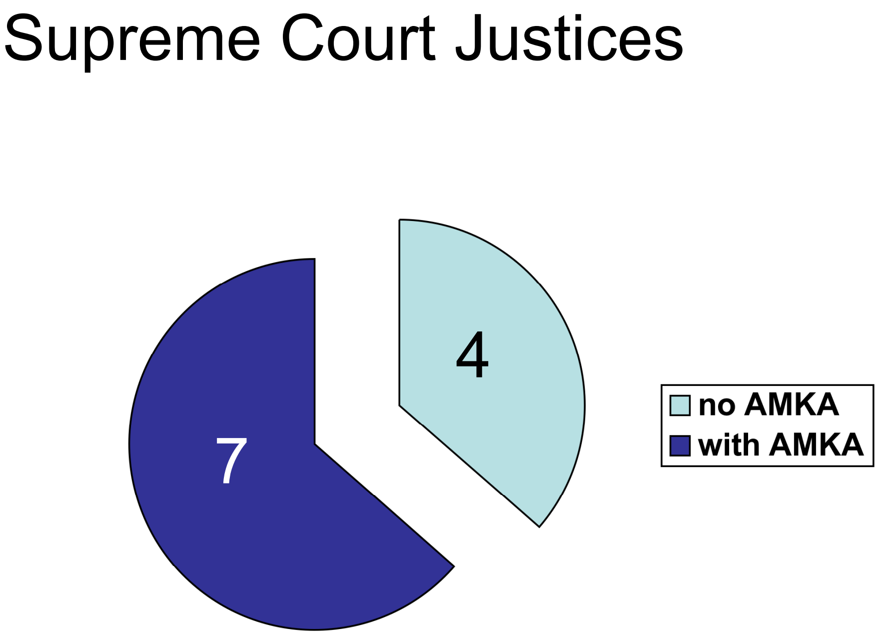


Sample Results
(data on the Web + brute-force search)

	Total	With AMKA	%
Public-No Mother's First Name	**7**	**3**	**42.8**
Public-No Date of Birth	**5**	**2**	**40**
Private-No Mother's First Name	**149**	**33**	**22.1**

More recent results





More recent results

	Total	With AMKA	%
Politicians	214	102	47.66
Supreme Court justices	11	7	63.64

Scenarios

- Private Data Confirmation
- Identity Confirmation
- False Medical Payments
- Identity Spoofing

Solutions

- Web form should require a Taxpayer ID or a National ID for authentication purposes
  - This exists for some politicians
- Greek citizens should have the choice to be taken off this online look-up service

SAFELINE: Safer Internet

- Safeline is a member of INHOPE
- INHOPE is the International Association of Internet Hotlines fighting Internet illegal content. Founded in 1999 under the EC Safer Internet Action Plan.

inhope.
Internet Hotline Providers

- Reports received by Safeline regarding Internet illegal content are rapidly increasing

Exploitation/Collaboration Opportunities

- New EU projects
- New International Projects
- Implement national projects
  - E.g. ΕΣΠΑ
- Provide expertise to
  - Organizations
  - Companies
- Exchange students
  - SysSec