On device Anomaly Detection for resource-limited systems

Maroua Ben Attia

École de technologie supérieure (ÉTS)
Dép. Génie logiciel et des TI

OUTLINE

1. Introduction and Motivation
2. Framework's Architecture:
   - ✓ Anomaly Detection
   - ✓ Storage
   - ✓ Profiling
3. Project Summary
4. Feedback?

Introduction:
Embedded System Evolution

Year	Mobile device	RAM (MB)	Storage (MB)	CPU(Mhz)	Connectivity
2004	Nokia 6630	**10**	**64**	**220**	Bluetooth, EDGE, GPRS
2005	HTC Universal	**64**	**128**	**520**	Bluetooth, WLAN, GPRS
2006	HTC TyTN	**64**	**128**	**400**	Bluetooth, WLAN, GPRS, EDGE
2007	Nokia N95	**128**	**8192**	**332 x2**	Bluetooth, WLAN, GPRS, EDGE
2008	Nokia N96	**128**	**16384**	**264 x2**	Bluetooth, WLAN, GPRS, EDGE
2009	Apple iPhone 3GS	**256**	**32764**	**600**	Bluetooth, WLAN, GPRS, EDGE
2010	Samsung Galaxy S	**512**	**16384**	**1000**	Bluetooth, WLAN, GPRS, EDGE
2011	Samsung Galaxy S2	**1024**	**16384**	**1200**	Bluetooth, WLAN, GPRS, EDGE
2012	iPhone 5	**1024**	**65563**	**1300 x2**	Bluetooth, WLAN, GPRS, EDGE
2013	Samsung Galaxy S4	**2048**	**65563**	**1600 x4**	Bluetooth, WLAN, GPRS, EDGE
2014	Samsung Galaxy S5	**2048**	**65563**	**2500 x4**	Bluetooth, WLAN, GPRS, EDGE

In 10 Years RAM: 204 x Storage: 1024 x CPU: 45 x

Introduction:
Malware Evolution (1)

Mobile devices

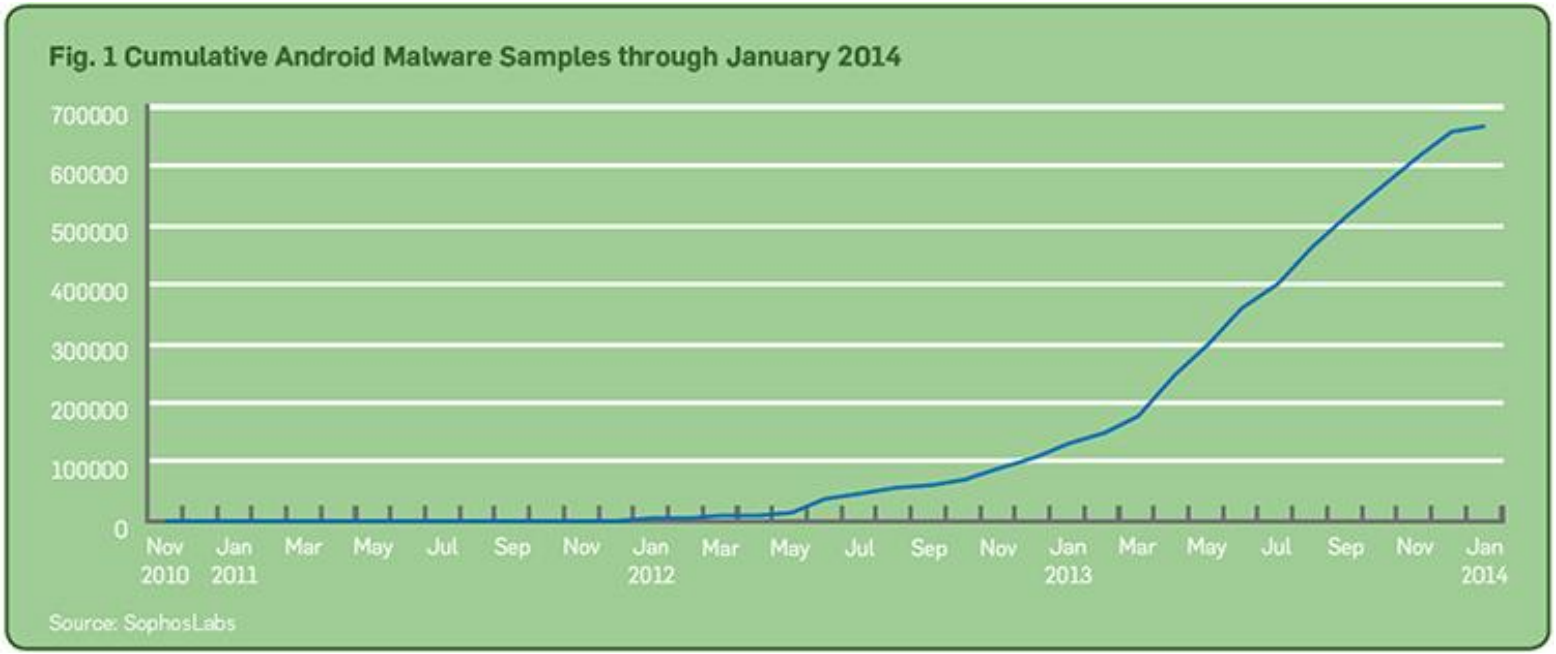
2 years of mobile malware evolution <=> 20 years of Computer malware evolution

SOURCE: Sophos, "Mobile Security Threat Report", 2014, http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf

Introduction:
Malware Evolution (2)

Mobile devices

"F-Secure 2014: Android devices are the more popular target for attacks with 294 new threat families or variants "



SOURCE: Sophos, " *Sophos Mobile Security Threat Report*",
http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf

Introduction:
Malware Evolution (3)

General-purpose small devices

Year	Malware /attack	Target	Threats
2009	psyb0t	Linux-based routers, DSL modems	DDoS
2010	Chuck Norris Botnet	Linux-based routers, DLS modems	DDoS +DNS Spoofing
	Stuxnet	Industrial control systems (ICS)	Alter PLCs for supported facilities
2012	DNSChanger	Computers and routers	DNS spoofing/poisoning
2013	JUL: GPS attack	GPS based systems	Total control of system
	Sept: Linux/Flasher	Wireless routers	Login credentials captured and transferred to remote web servers.
	Nov 26 : Linux.Darlloz	Linux-based computers, industrial control servers, routers, cameras, set-top boxes.	Generates IP @ randomly, accesses a specific path on the machine with well-known ID and passwords, and sends HTTP POST requests
2014	Feb: The Moon	Linksys routers	Bypasses authentication on the router and scans for other vulnerable routers to infect

Problematic (1)

Security Issue

Just 2014 !

March 20th, 2014, 12:55 GMT · By Eduard Kovacs

Linux Worm Darlloz Infects over 31,000 Devices in Four Months

Linux

http://news.softpedia.com/news/Linux-Worm-Darlloz-Infects-over-31-000-Devices-in-Four-Months-433242.shtml

Android

Monday, 17 February 2014

Android SMS malware hosted on Google Play infects 1.2 Million users

http://www.hackleaks.in/2014/02/android-sms-malware-hosted-on-google.html

"The Moon scans for vulnerable devices as it looks to continue spreading, over 1,000 Linksys routers are already believed to be infected by the malware." (2014)

Linux

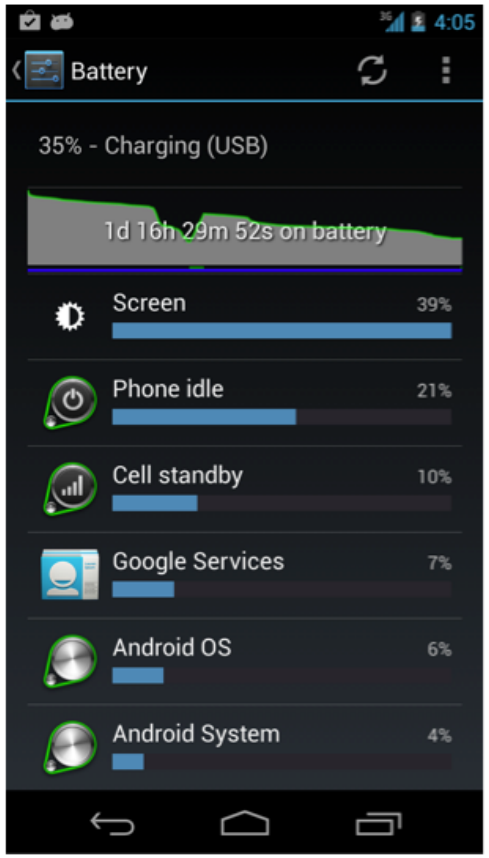
http://www.ubergizmo.com/2014/02/linksys-routers-malware-the-moon-spreading/

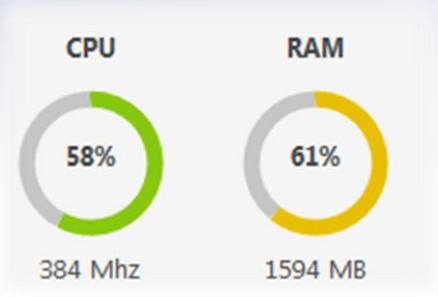
A Criminal campaign named Windigo Operation has controlled about 25 thousand Unix servers that send millions of fake mails and put 500 thousand computers at risk every day. ***(2014)***

http://www.rcoutada.net/2014/03/new-linux-servers-cpanel-backdoor-ebury-a/

Problematic (2)
Resource Limitations

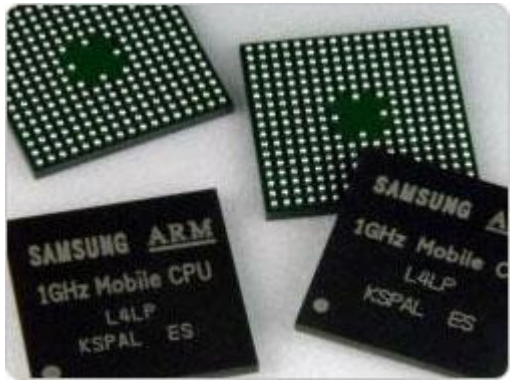
Battery life





Low power CPUs

- Lightweight processing
- limited multitasking



Limited Memory

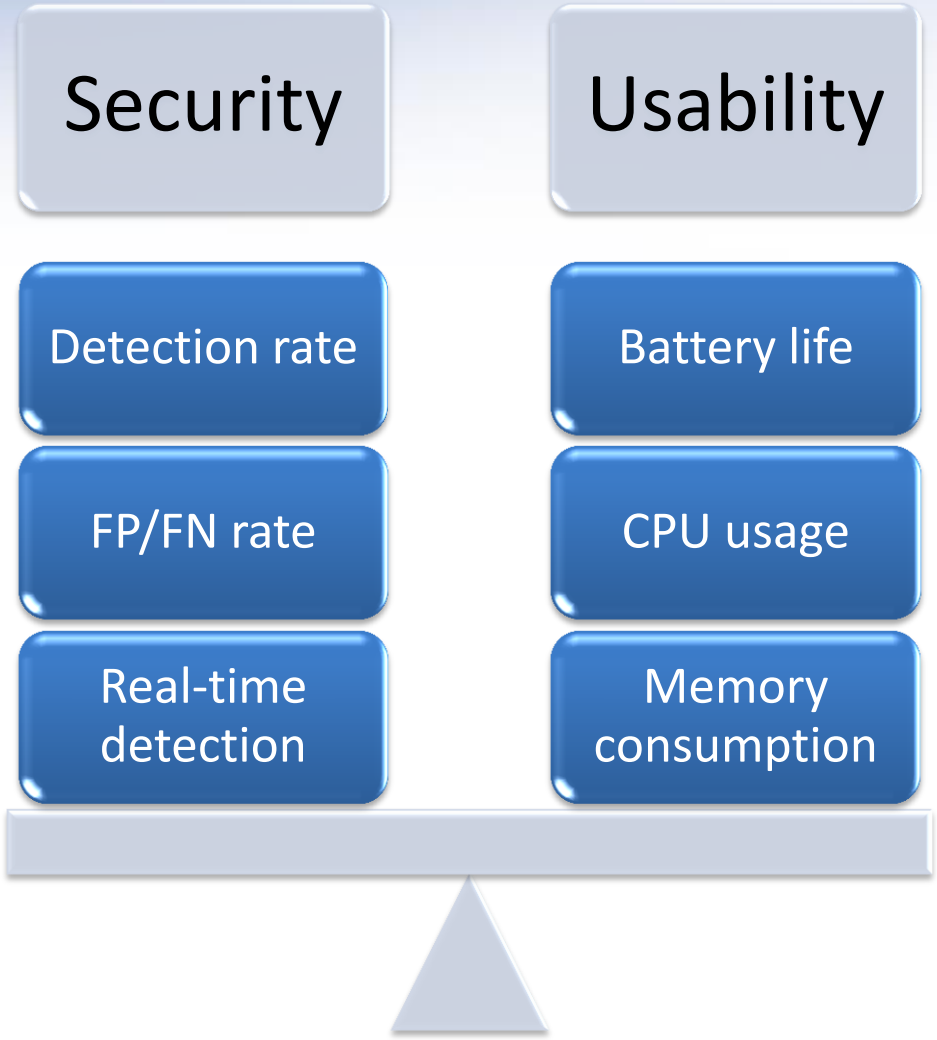
Problematic (3)

Resource Limitations

Other small-scale Embedded systems

	BeagleBone	Overo® FE COM (Gumstix)	Gumstix (DuoVero)	Radxa Roc	KTAM3874/ pITX	APC 8750
CPU	**720MHz**	**600 MHz**	**Dual-Core 1GHz**	**Quad Core, 1.6 GHz**	**800 MHz**	**800MHz**
RAM	**256 MB**	**512 MB**	**1GB**	**2 GB**	**2GB**	**512MB**
Storage	**4GB microSD**	**microSD slot**	**microSD slot**	**8GB**	**16 GB**	**2GB**
OS	Android, Linux	Linux , Android	Linux, Android	Android, Linux	Android, Linux	Android
Size	76.2 ×76.2 ×16mm	58mm x 17mm x 4.2mm	58mm x 17mm x 4.2mm	100 x 80 mm	100 x 72mm	170 x 85mm

Problematic (4)



Framework's Architecture

- Data Collection
- Data Processing
- Scan/Model Management

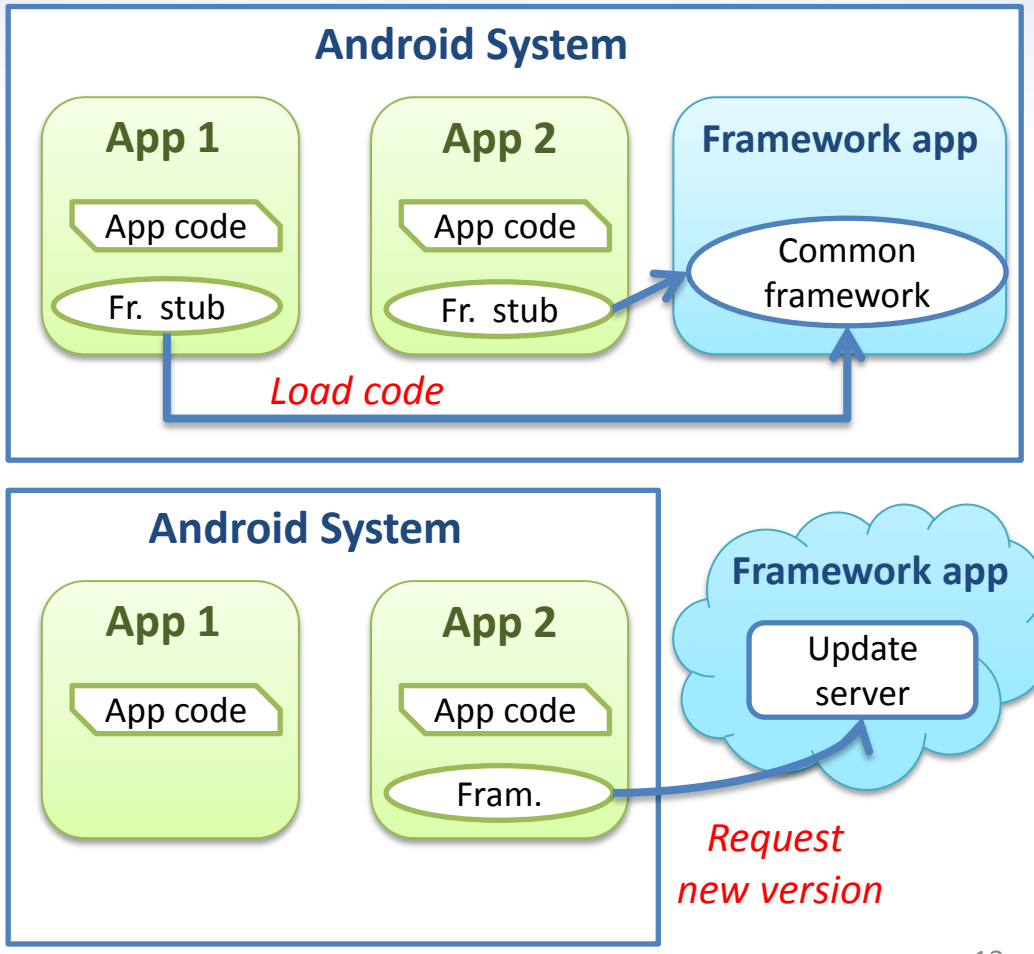
Development Platform: Android

- Open source
- Kernel Linux (modified)
- Programming languages: Java, C++, C#
- Mainly supports ARM
- Applications: Smartphones, tablet PCs, PDAs and mobile devices

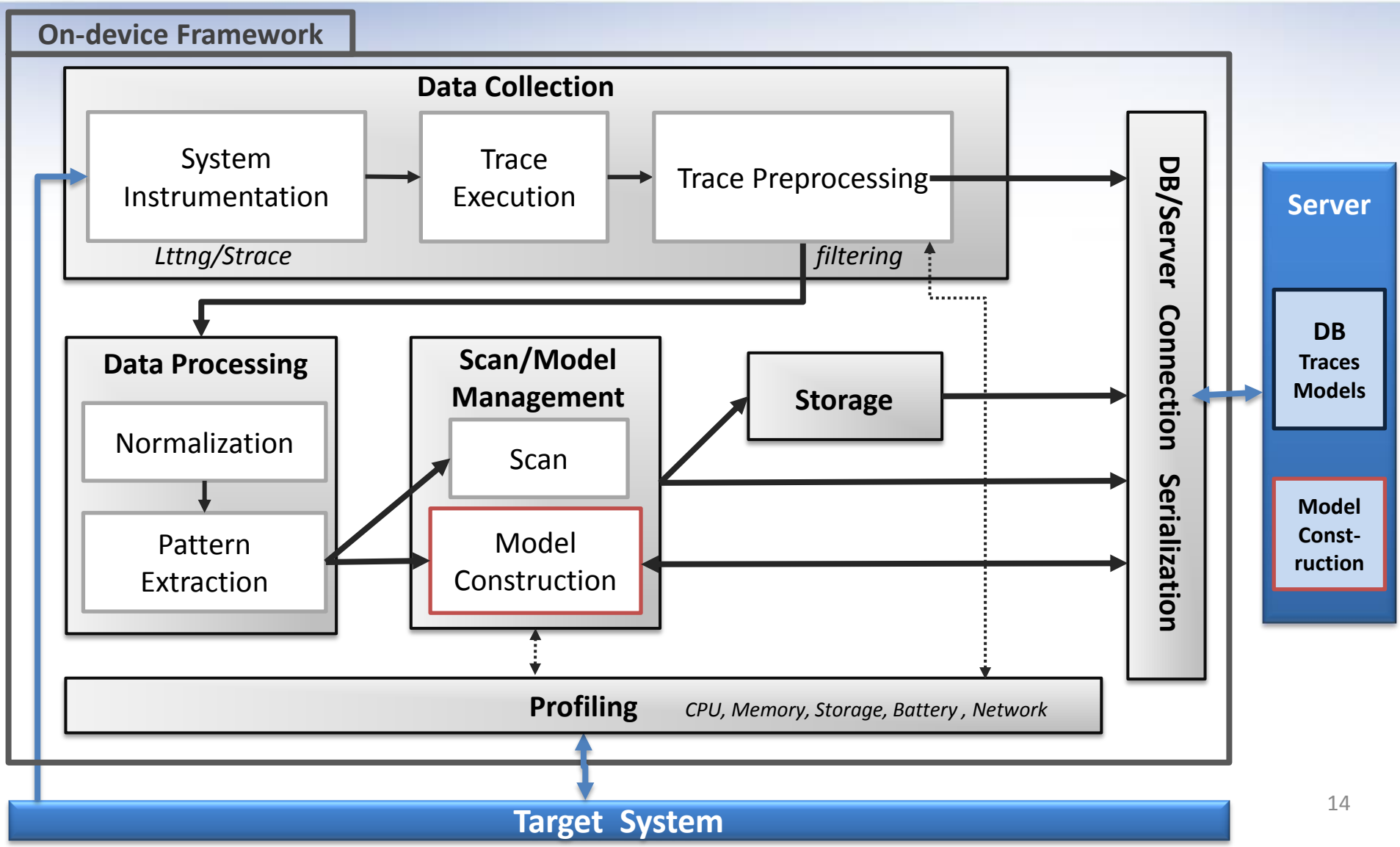
Attack scenario - Android

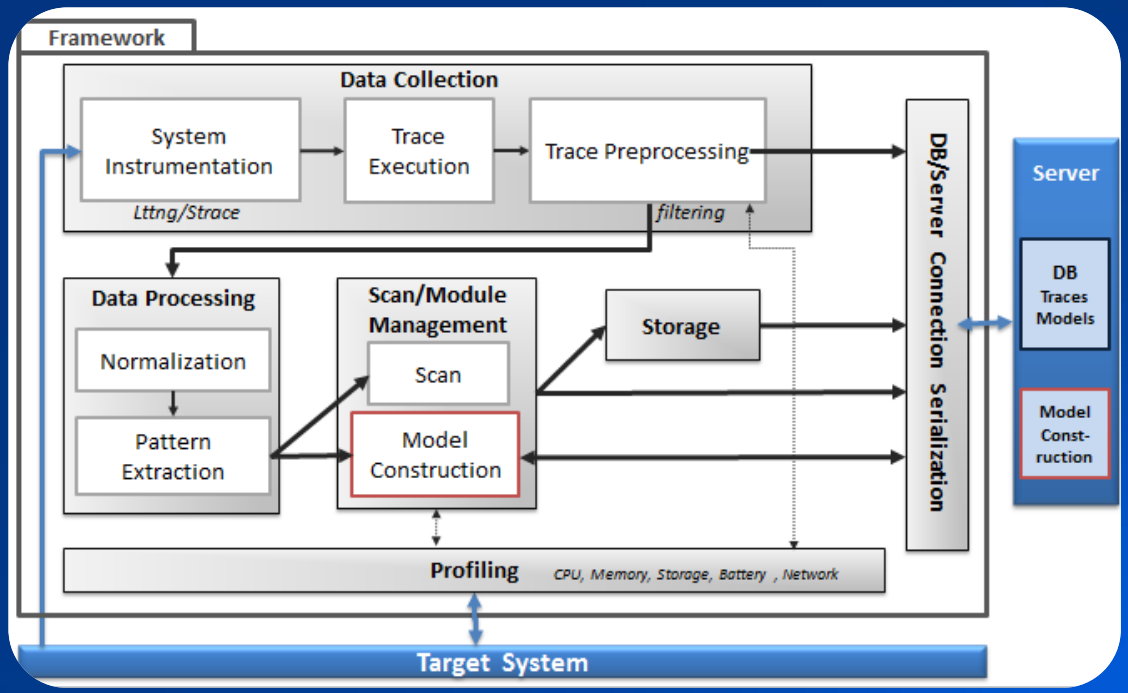
- Benign applications loads, for benign reasons, additional code that can be replaced with malicious ones by the attacker
- Malicious application that does not contain initially any clearly malicious code, but downloads additional faked code after being installed on a device .
- Attackers can trick users to download and install fake updates for already installed legitimate applications by injecting malicious code (without breaking their signatures).

→ Skype, Angry-birds, Opera browser…



Framework's Architecture



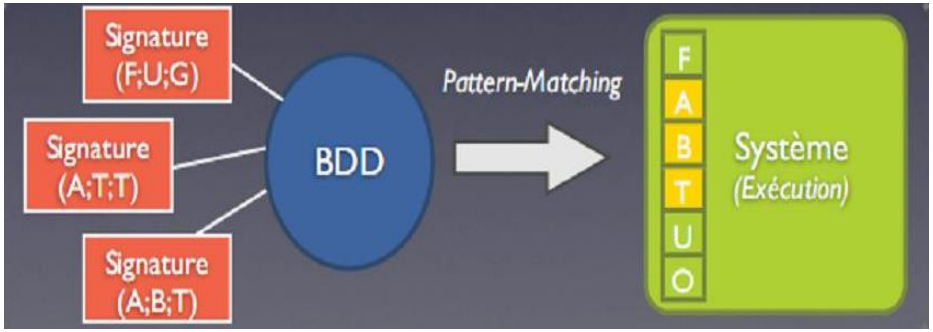


- Data Collection
- Data Processing
- Scan/Model Management

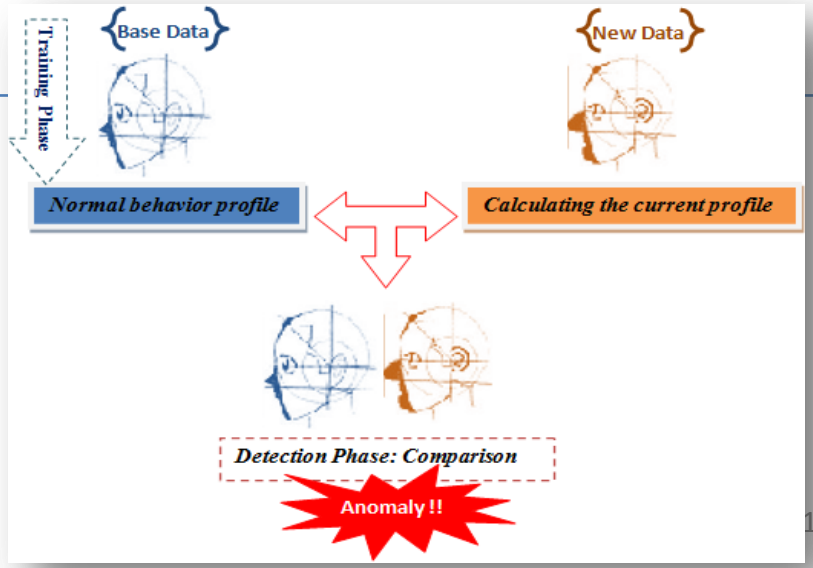
DETECTION PROCESS

Intrusion Detection Techniques

	Signature-Based	Anomaly-Based
?	Looking for "known patterns" of specific malware activity (list of stored signature for each malware)	Learning phase: establishes a base of knowledge about "normal" behavior. Detection phase: once a behavior is too different from training data, it is considered abnormal.
+	Low false positive rate Very accurate and Fast	Can detect both known and unknown malwares Accuracy increases as increasing training data
-	Can only detect known intrusions Required memory budget : varying numbers of signatures. DB must be constantly updated	High false positive rate Slow

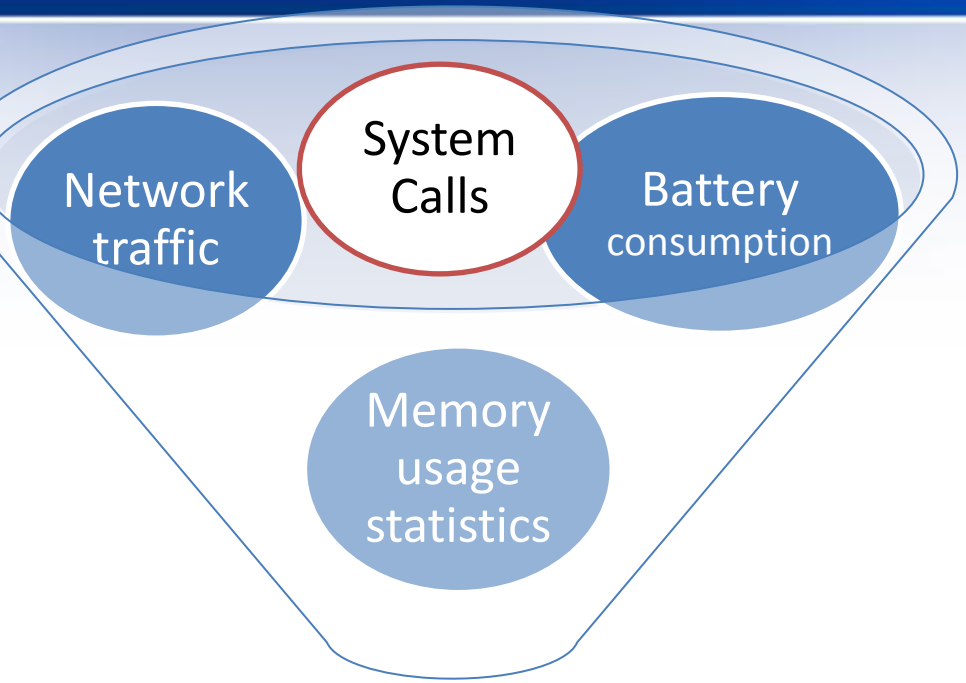


Fast evolution of signatures database
→ never feet memory of small-scale systems



Data Collection

System Instrumentation: strace



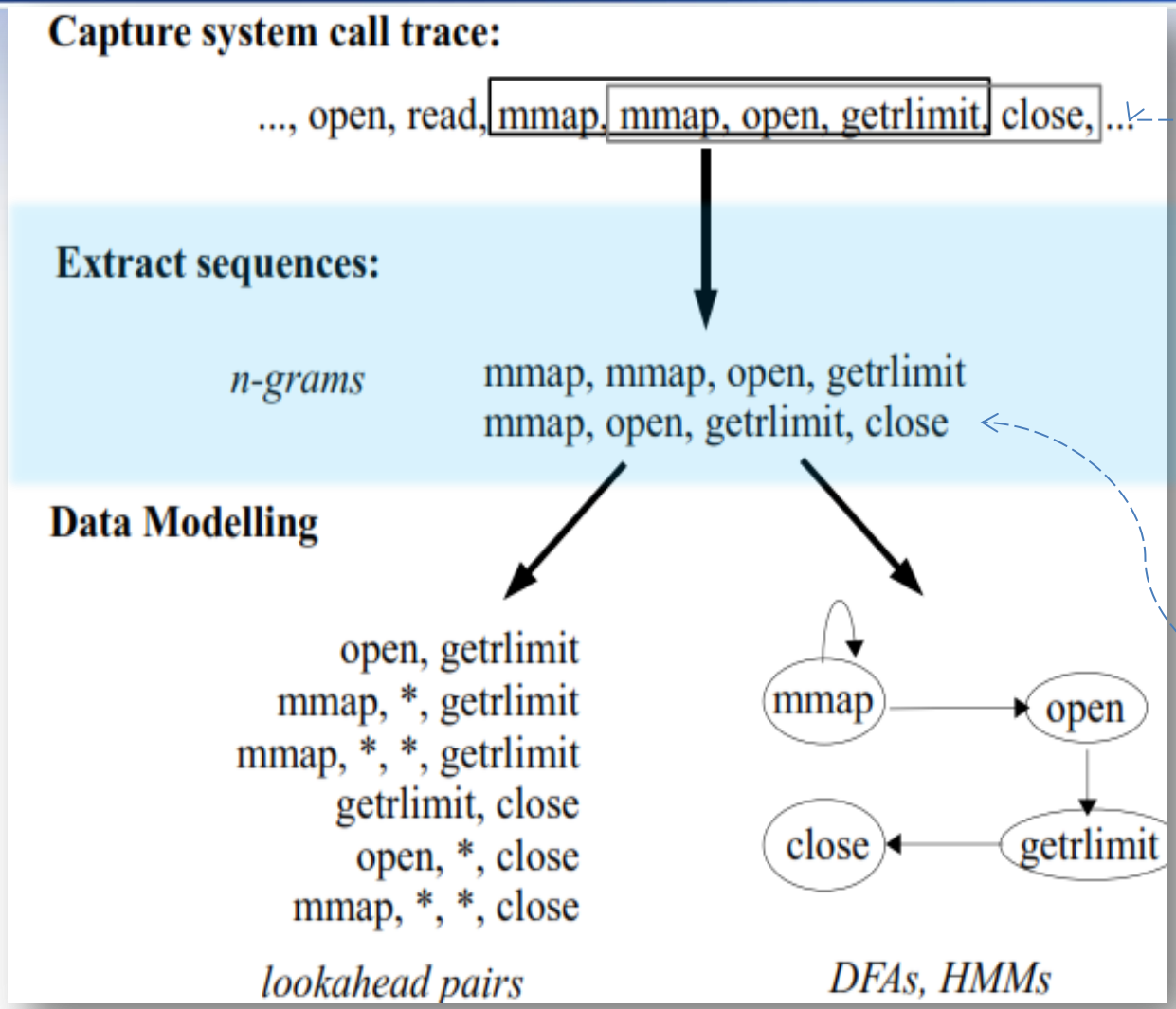
Types of Data

- ✓ Provided by the kernel and used by programs running in user space.
- ✓ Any compromised program must generate some system calls to cause the damage.

- **Output Data**

```
recvmsg(6, 0x7fff2a9896c0, 0)           = -1 EAGAIN (Resource temp
poll([{fd=3, events=POLLIN}, {fd=6, events=POLLIN}, {fd=8, events=
read(3, "\7\0\0\0\0\0\0\0", 16)         = 8
recvmsg(6, 0x7fff2a9896e0, 0)           = -1 EAGAIN (Resource temp
recvmsg(6, 0x7fff2a9896c0, 0)           = -1 EAGAIN (Resource temp
poll([{fd=3, events=POLLIN}, {fd=6, events=POLLIN}, {fd=8, events=
read(3, "\4\0\0\0\0\0\0\0", 16)         = 8
recvmsg(6, 0x7fff2a9896e0, 0)           = -1 EAGAIN (Resource temp
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
write(3, "\1\0\0\0\0\0\0\0", 8)         = 8
recvmsg(6, 0x7fff2a9896c0, 0)           = -1 EAGAIN (Resource temp
poll([{fd=3, events=POLLIN}, {fd=6, events=POLLIN}, {fd=8, events=
read(3, "\10\0\0\0\0\0\0\0", 16)        = 8
recvmsg(6, 0x7fff2a9896e0, 0)           = -1 EAGAIN (Resource temp
recvmsg(6, 0x7fff2a9896c0, 0)           = -1 EAGAIN (Resource temp
poll([{fd=3, events=POLLIN}, {fd=6, events=POLLIN}, {fd=8, events=
read(3, 0x7fff2a989850, 16)             = -1 EAGAIN (Resource temp
recvmsg(6, 0x7fff2a9896e0, 0)           = -1 EAGAIN (Resource temp
recvmsg(6, 0x7fff2a9896c0, 0)           = -1 EAGAIN (Resource temp
```

Data Processing
Pattern Extraction



n-gram extraction

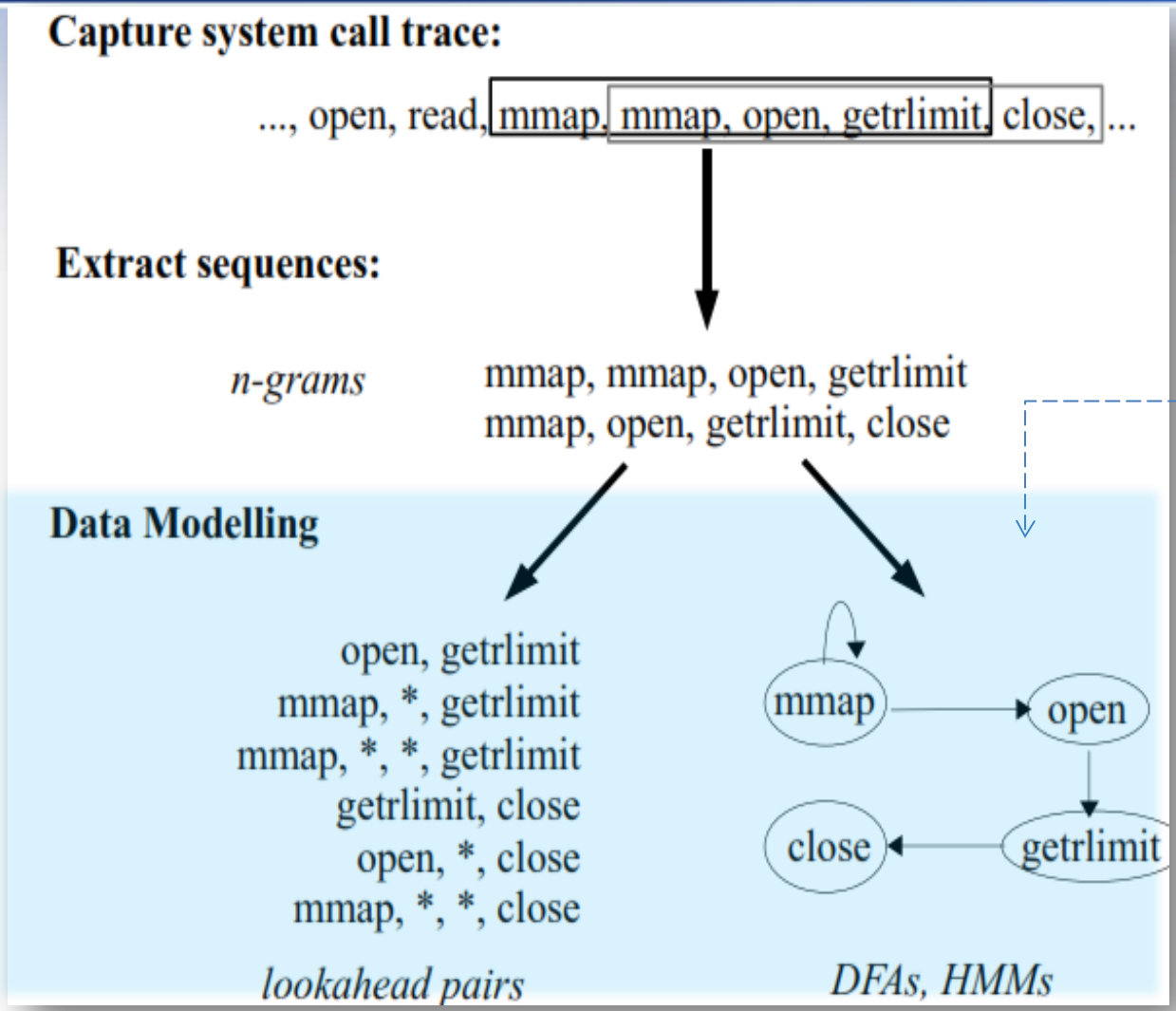
Sliding a window (Expl. Size = 4)

Size: static /dynamic

mmap → 1-gram
mmap,open → 2-grams
mmap,open, getrlimit → 3-grams
…

SOURCE: *Forrest, S.; Hofmeyr, S.; Somayaji, A.; The Evolution of System-Call Monitoring"*,2008, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4721577&isnumber=4721526

Model Construction



Modeling abnormal behavior

→ What about new malwares?

Modeling normal behavior

→ Large amount of trace data from normal operation is available

→ Detecting new malwares

This model is used to determine whether a new trace is normal or malicious.

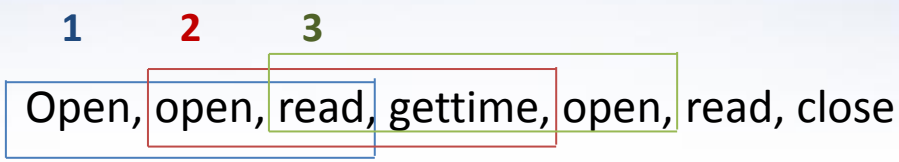
SOURCE: *Forrest, S.; Hofmeyr, S.; Somayaji, A.; The Evolution of System-Call Monitoring"*,2008, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4721577&isnumber=4721526

Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines

Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines

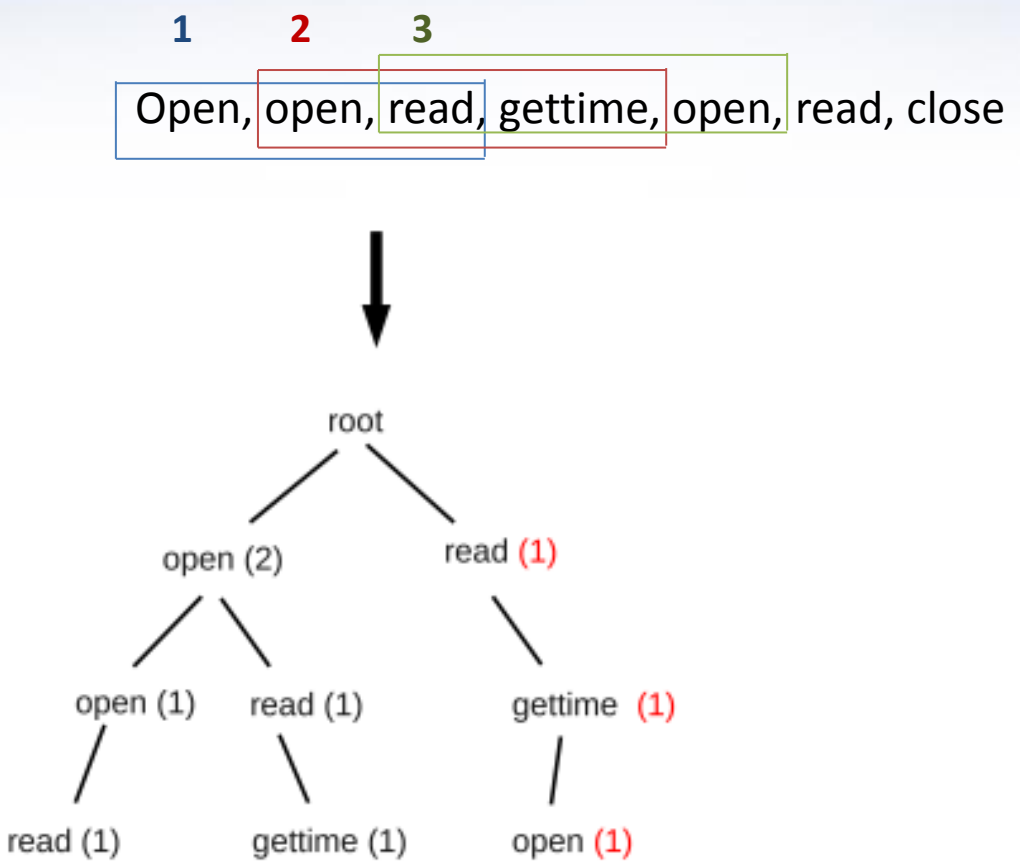


W=3	syscall	1 after	2 after
w1	open	open	read
w2	open	read	gettime
w3	read	gettime	open
w4	gettime	open	read
w5	open	read	close

W=3	syscall	1 after	2 after
Call 1	open	Open, read	Read, gettime, close
Call 2	read	gettime, close	open
Call 3	gettime	open	read

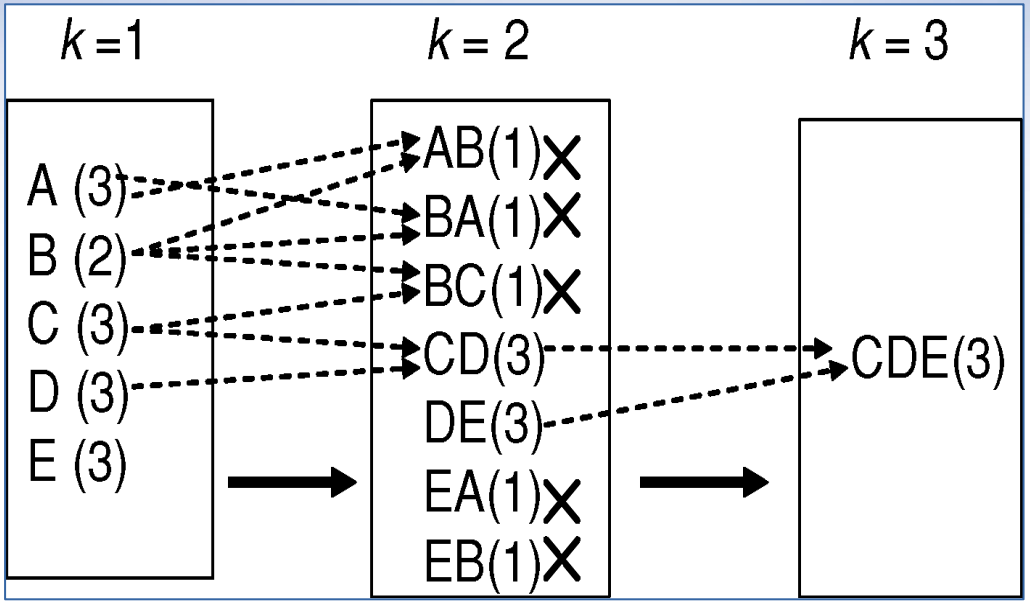
Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines



Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines



3 traces: «*ABCDE», «CDEA», «CDEBA»*

And α=0,6

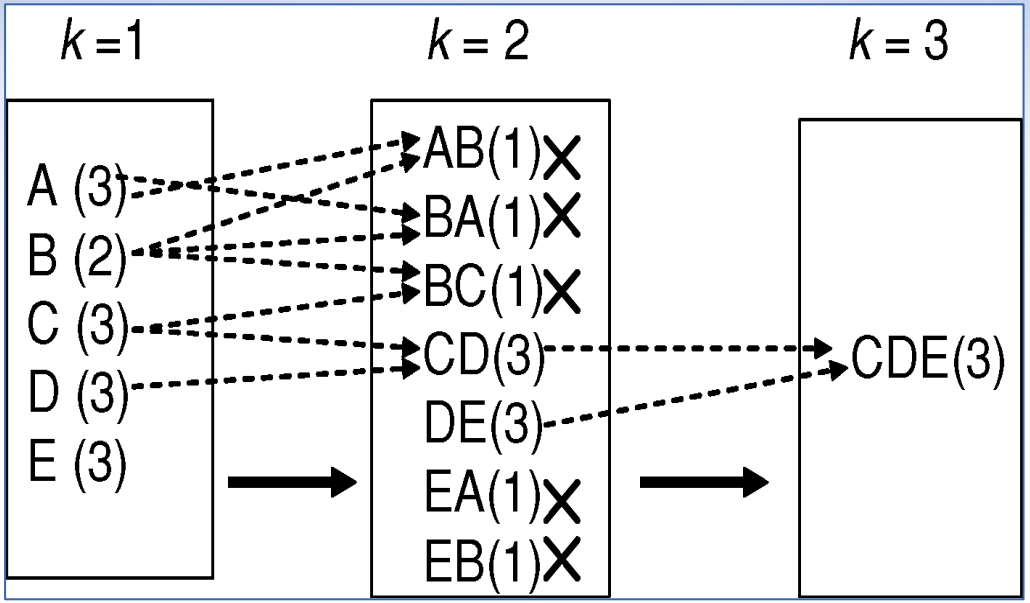
- $Last_{k-1}(CD) = D = First_{k-1}(DE)$
- f(CDE)=3> 0.6 min (f(CD),f(DE))=1.8
- C_3 . $add(S)$
  - C_3 ={CDE(3)}

Two valid n-grams of length k (r_k , q_k) are combined to create an n-gram of length k+1 (p_{k+1}) if:

$$f(p_{k+1}) > \alpha \min(f(r_k), f(q_k))$$

Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines



→ ***Reduce the size of dataset while keeping a good detection efficiency***

Output ={A, B, C, D, E, CD, DE, CDE}

→ 8 n-grams

VS

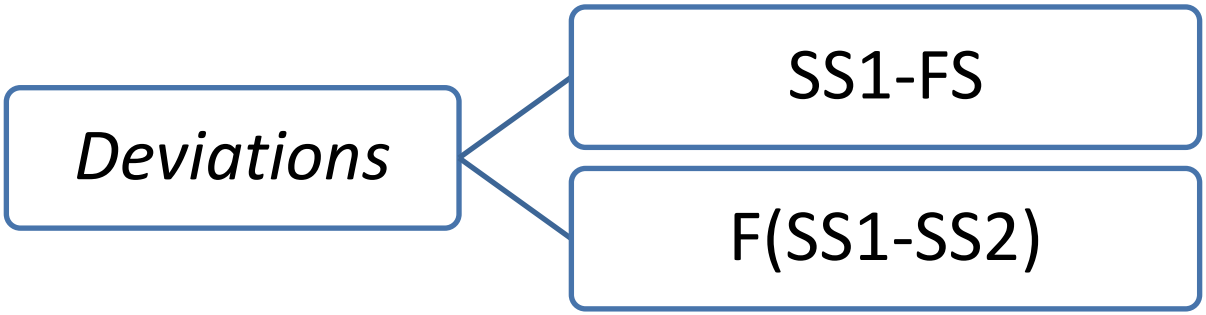
Wind(2)={AB, BC, CD, DE, CD, DE, EA, CD, DE, EB, BA}

→ 11 n-grams

Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines

✓ Syscall → numbers
✓ 2-grams (freq)
✓ P(Anomaly) for each Deviation



$$P(Anomaly) = \frac{(1 + SD_N(\boldsymbol{SS1})) * (\alpha * Z_N(\boldsymbol{SS1}) + 3)}{\left(Br(\boldsymbol{SS1})\right) * (Z_{Ab}(\boldsymbol{SS2}\ or\ \boldsymbol{FS}) + 3) * (Z_N(\boldsymbol{SS2}) + 3)}$$

Model Construction

- Lookahead pairs
- N-gram Tree
- Varied-length N-grams
- Finite State Machines

✓ Syscall → numbers
✓ 2-grams (freq)
✓ P(Anomaly) for each Deviation

Deviations
- SS1-FS
- F(SS1-SS2)

- *if* $(Z_N(\boldsymbol{SS1}) < -2)$ → *P=0*
- *if* $(Z_{Ab}(\boldsymbol{SS2}\ or\ \boldsymbol{FS}) < -2)$ → *P=1*
- *if* $(Z_N(\boldsymbol{SS2}) < -2)$ → *P=1*

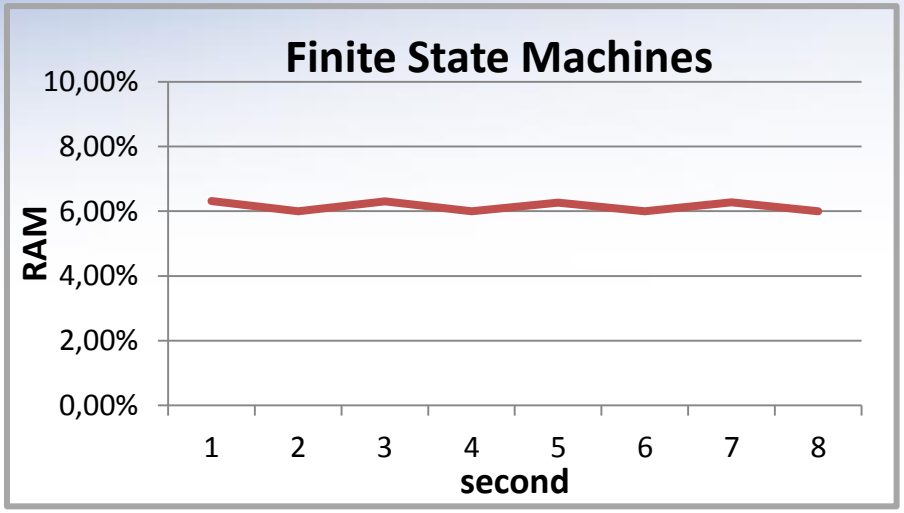
$$P(Anomaly) = \frac{(1 + SD_N(\boldsymbol{SS1})) * (\alpha * Z_N(\boldsymbol{SS1}) + 3)}{\left(Br(\boldsymbol{SS1})\right) * (Z_{Ab}(\boldsymbol{SS2}\ or\ \boldsymbol{FS}) + 3) * (Z_N(\boldsymbol{SS2}) + 3)}$$

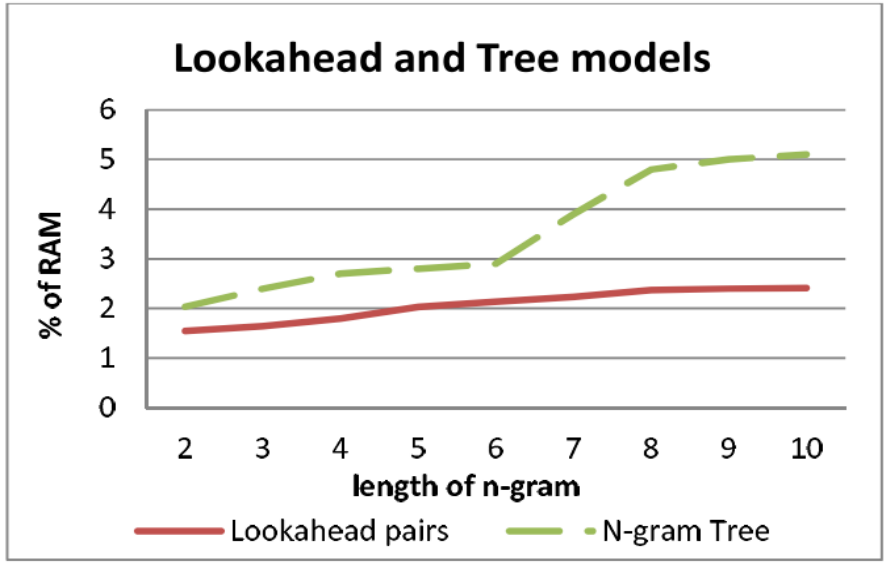
Experimental results
-Dataset-

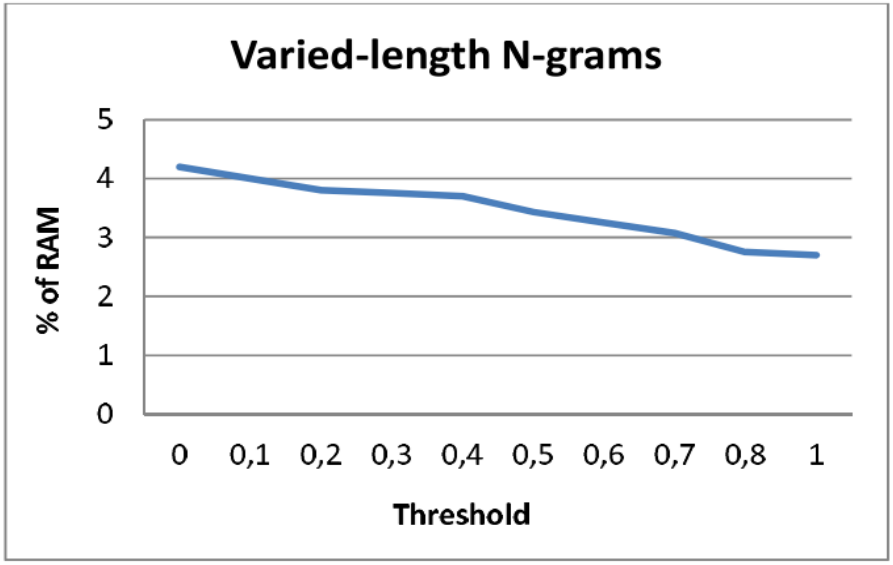
- Angry birds space
  - Normal version: 1.1.0
  - Malicious version: 1.1.2
- Candy Star
  - Normal version: 1.0.3
  - Malicious version: 1.0.2
- Ninja Chicken
  - Normal version: 1.4.8
  - Malicious version: 1.4.5

Experimental results
-Creating Normal profile-

- **RAM Overhead**

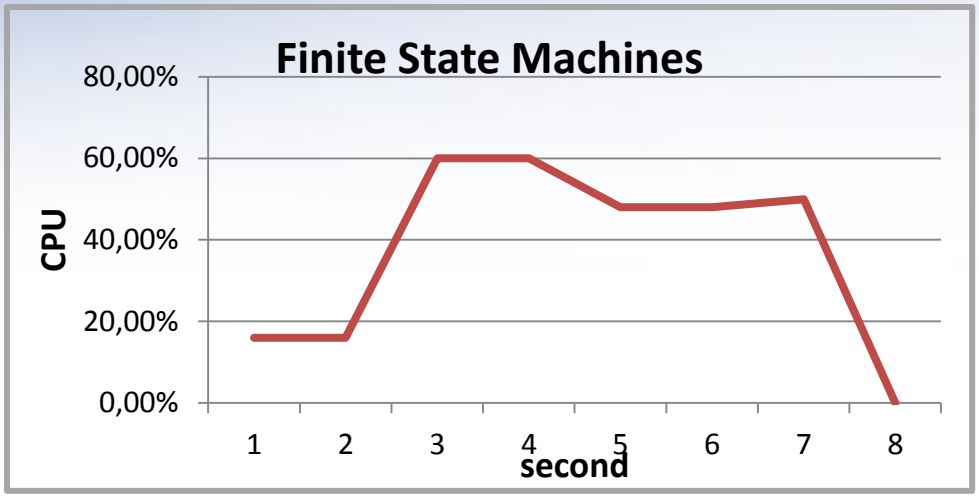


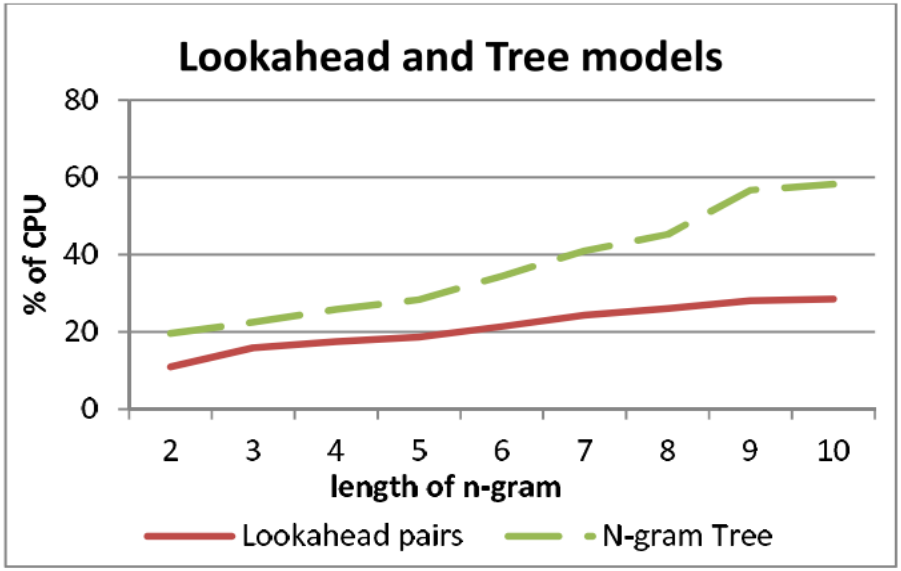


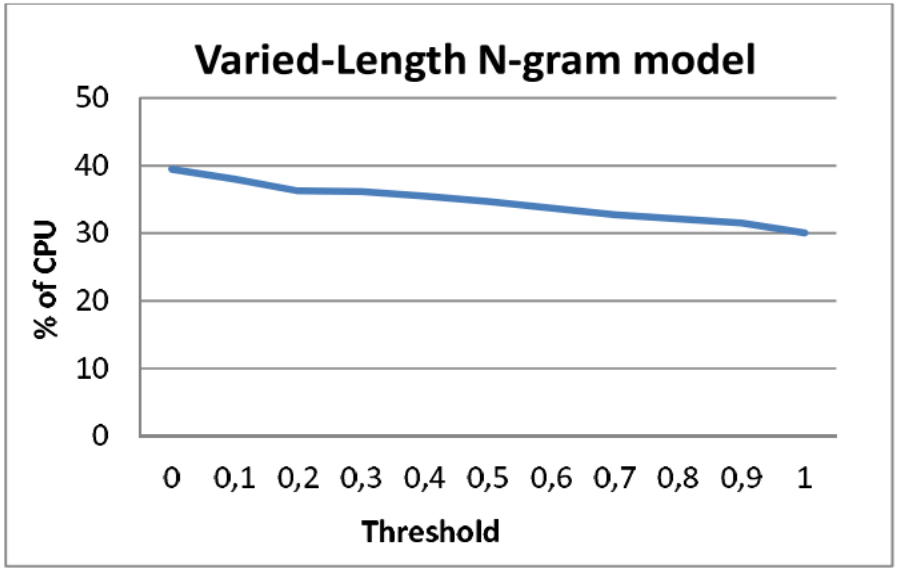


Experimental results
-Creating Normal profile-

- **CPU Overhead**

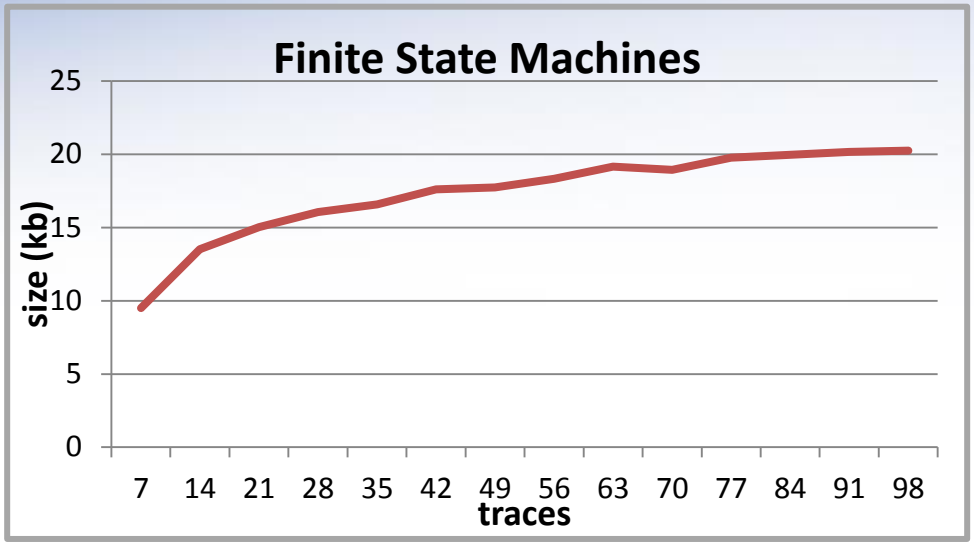


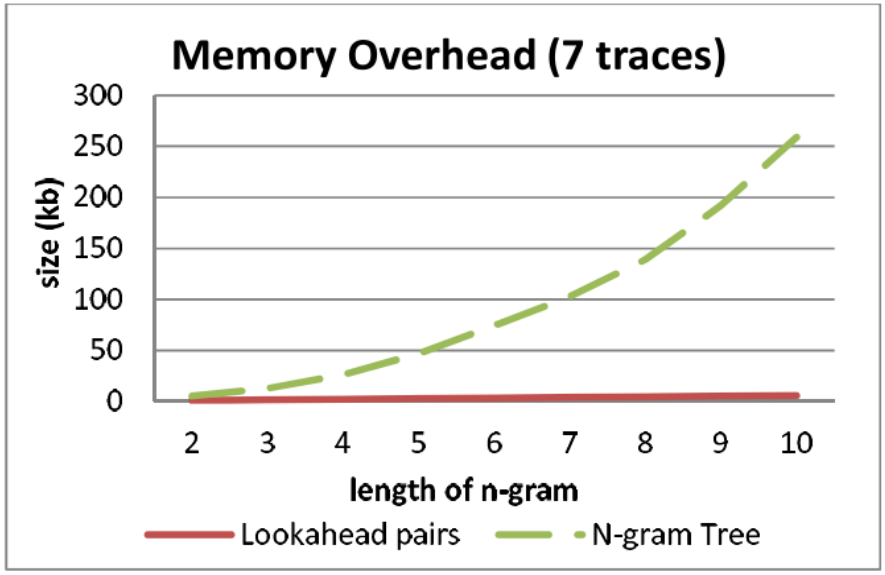


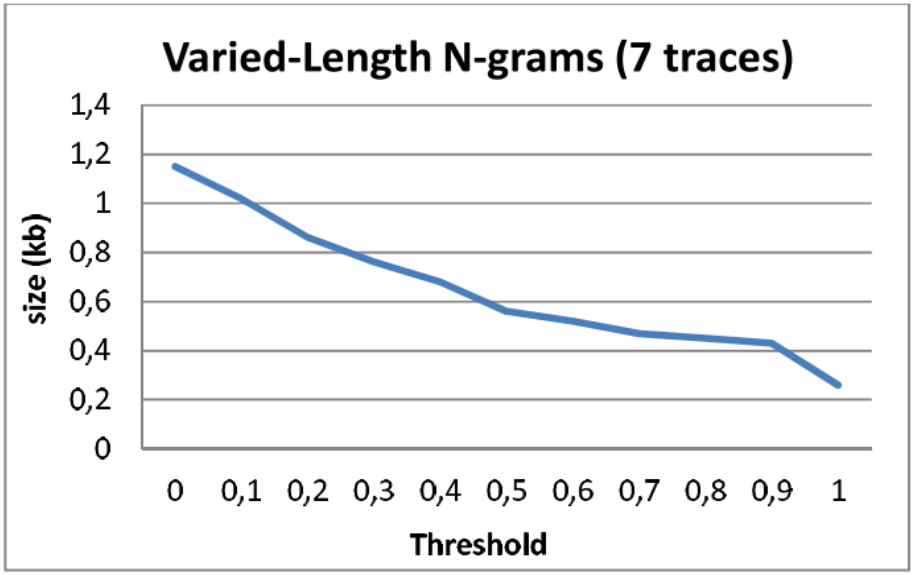


Experimental results
-Creating Normal profile-

- **Storage Overhead**





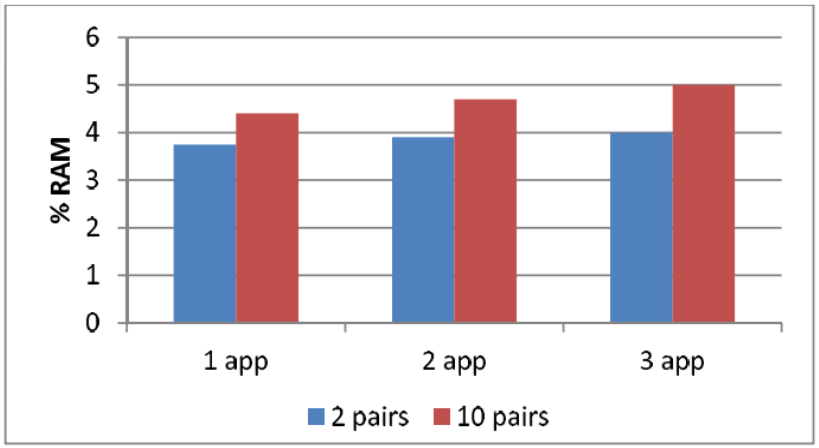


Experimental results

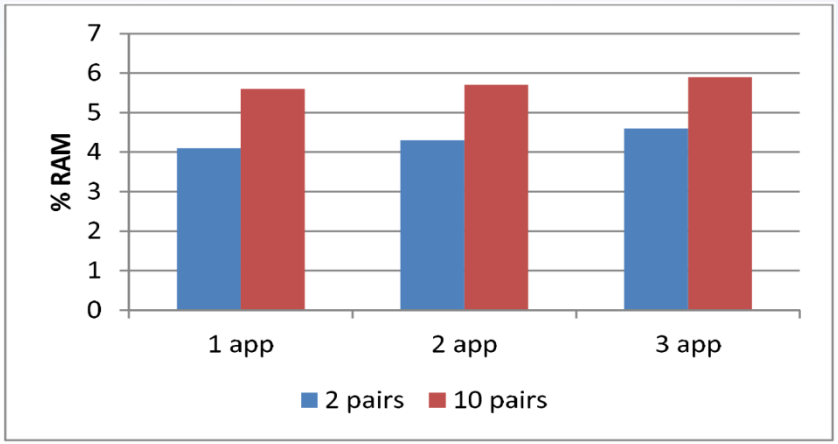
-Scanning 1, 2 and 3 applications in parallel-

- **RAM Overhead**

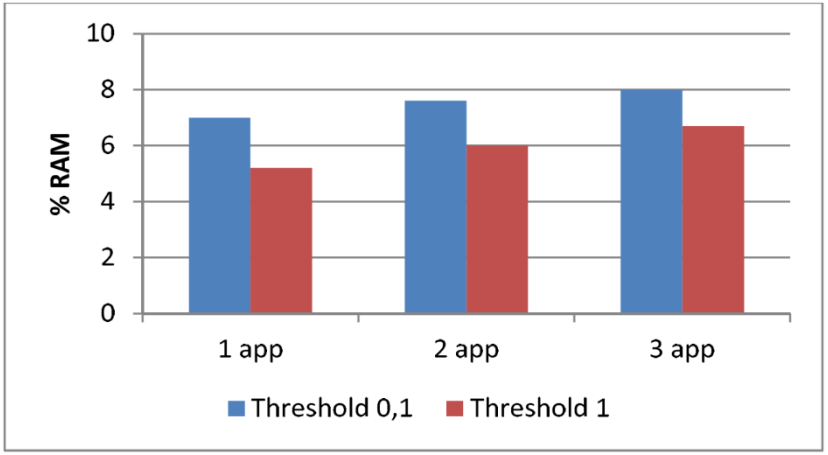
Lookahead



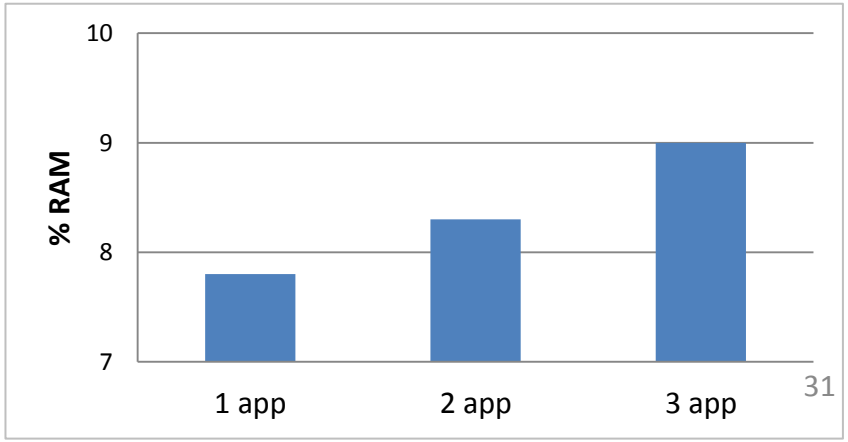
N-Gram Tree



VL N-gram



Finite State Machines

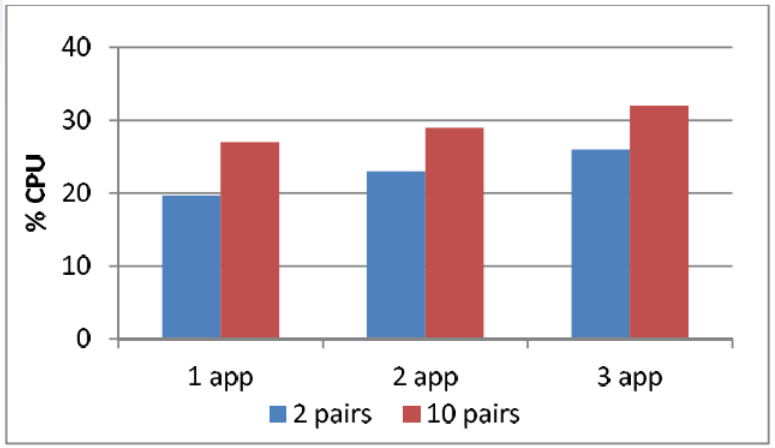


Experimental results

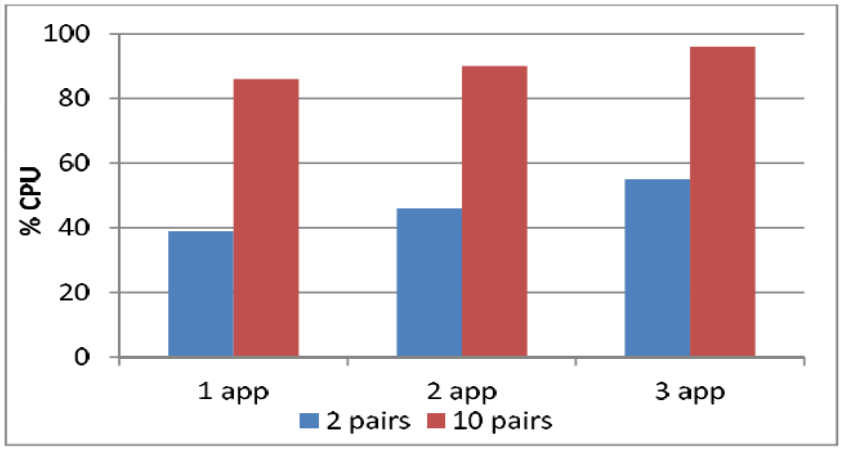
-Scanning 1, 2 and 3 applications in parallel-

- **CPU Overhead**

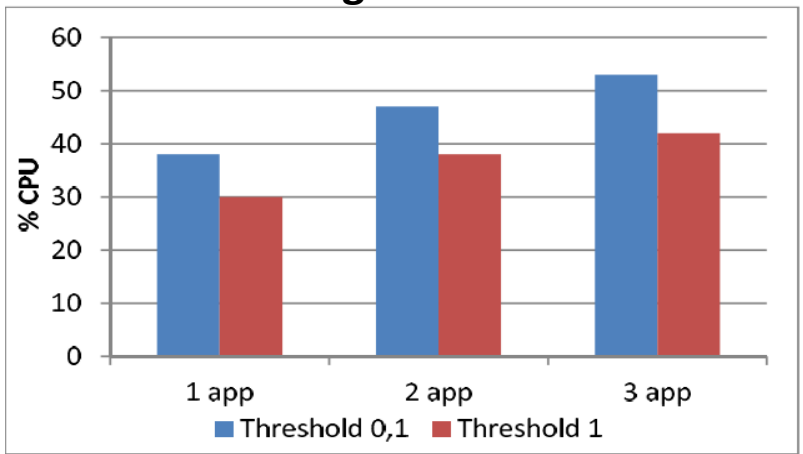
Lookahead



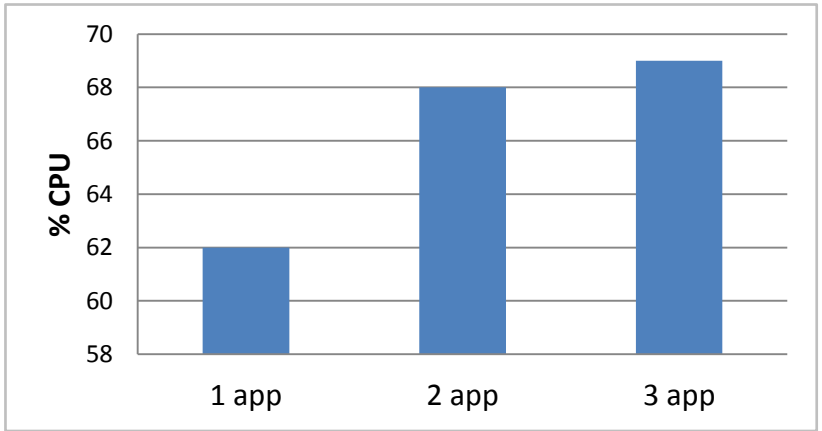
N-Gram Tree



VL N-gram

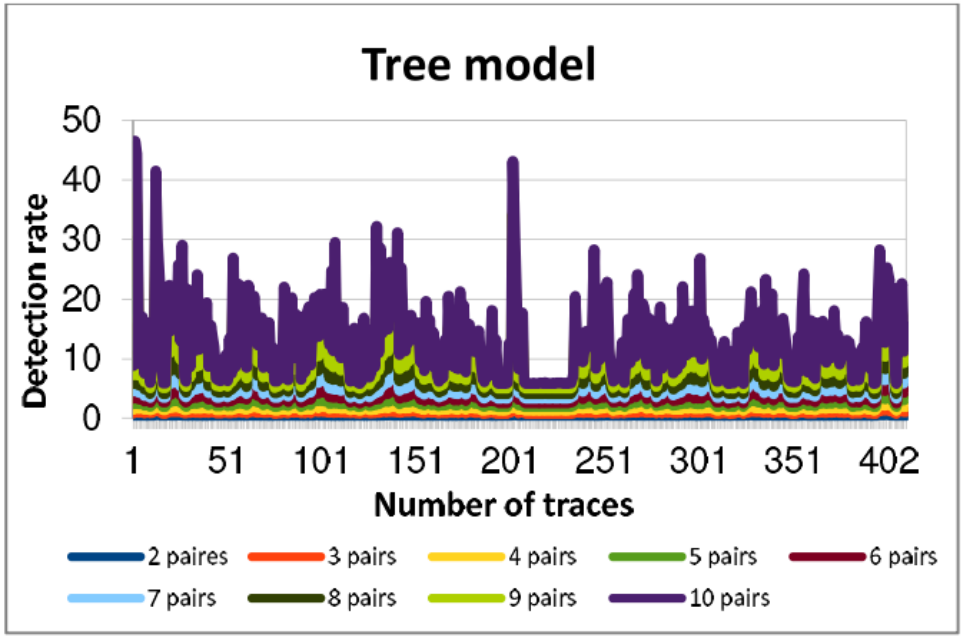


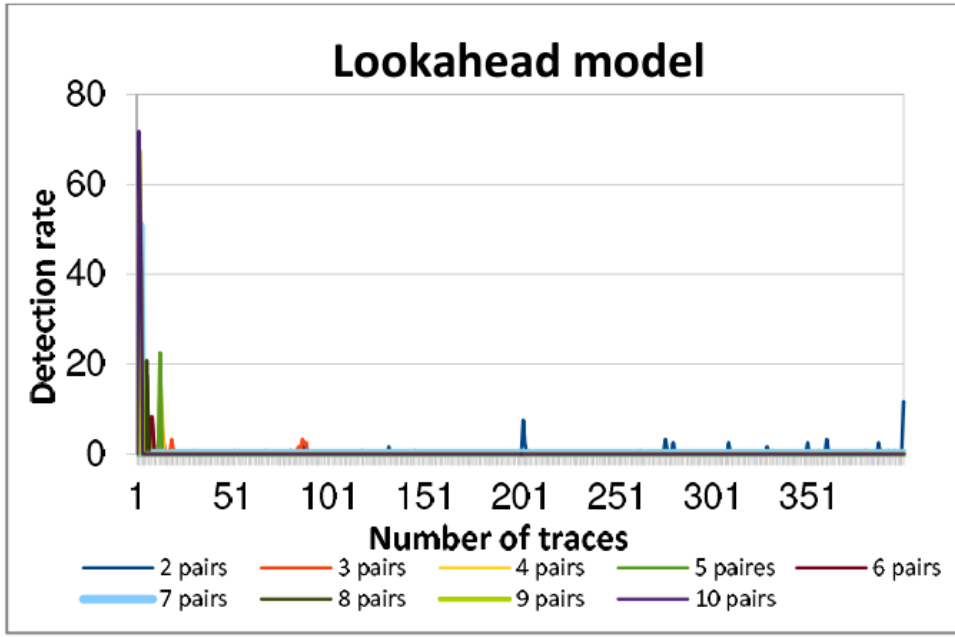
Finite State Machines



Experimental results

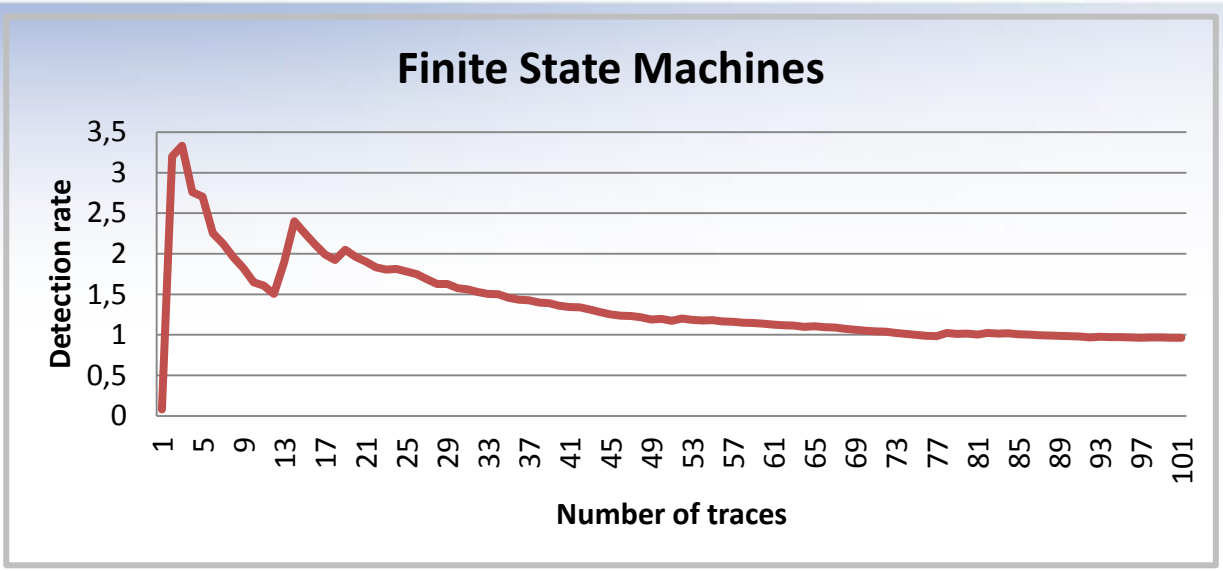
-Detection rate-



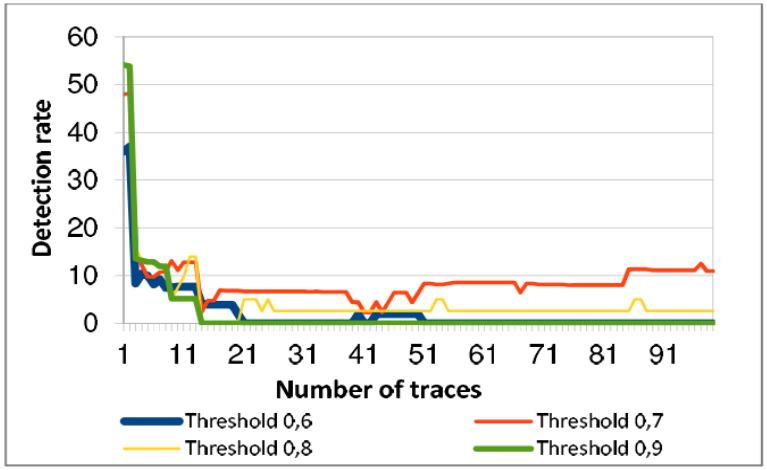


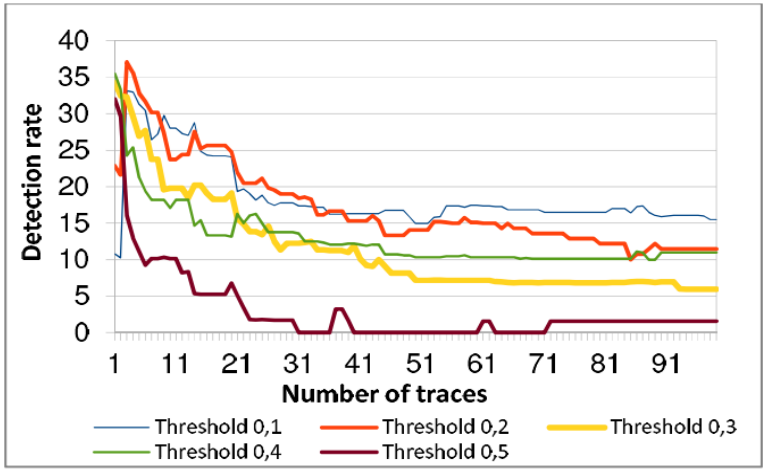
Experimental results

-Detection rate-



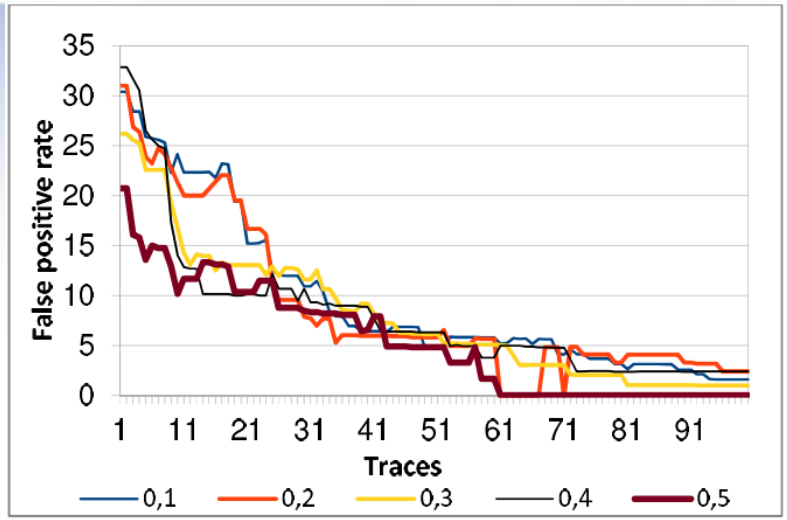
Varied-length N-grams

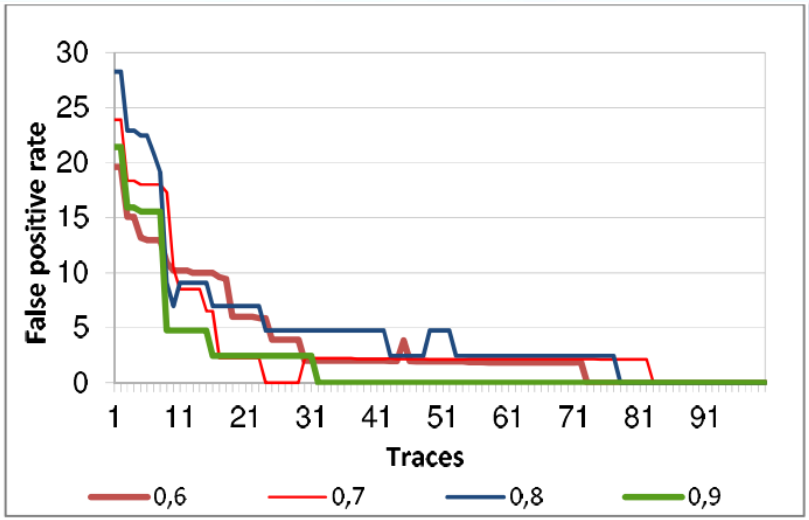




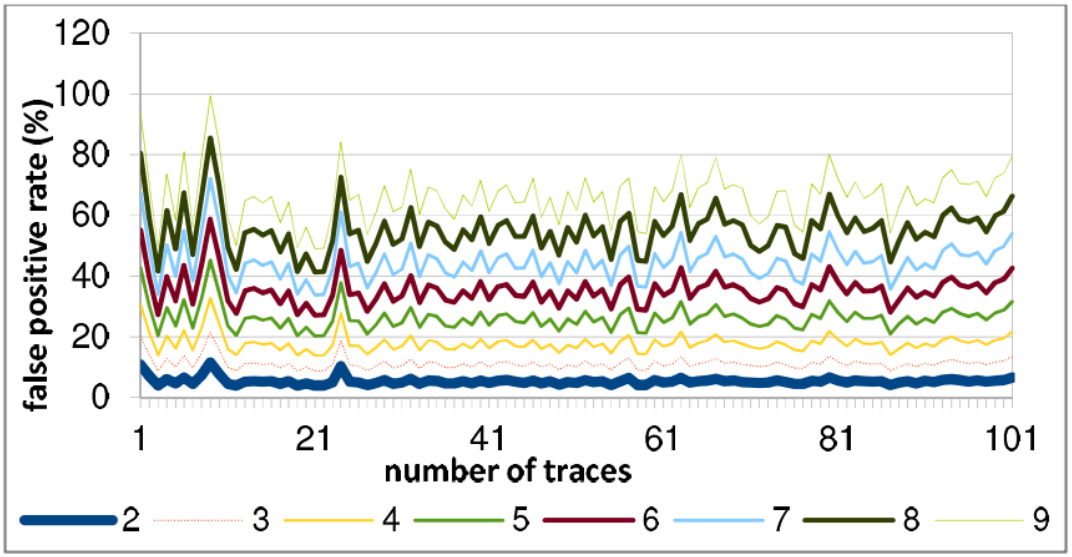
Experimental results

-False positive rate-

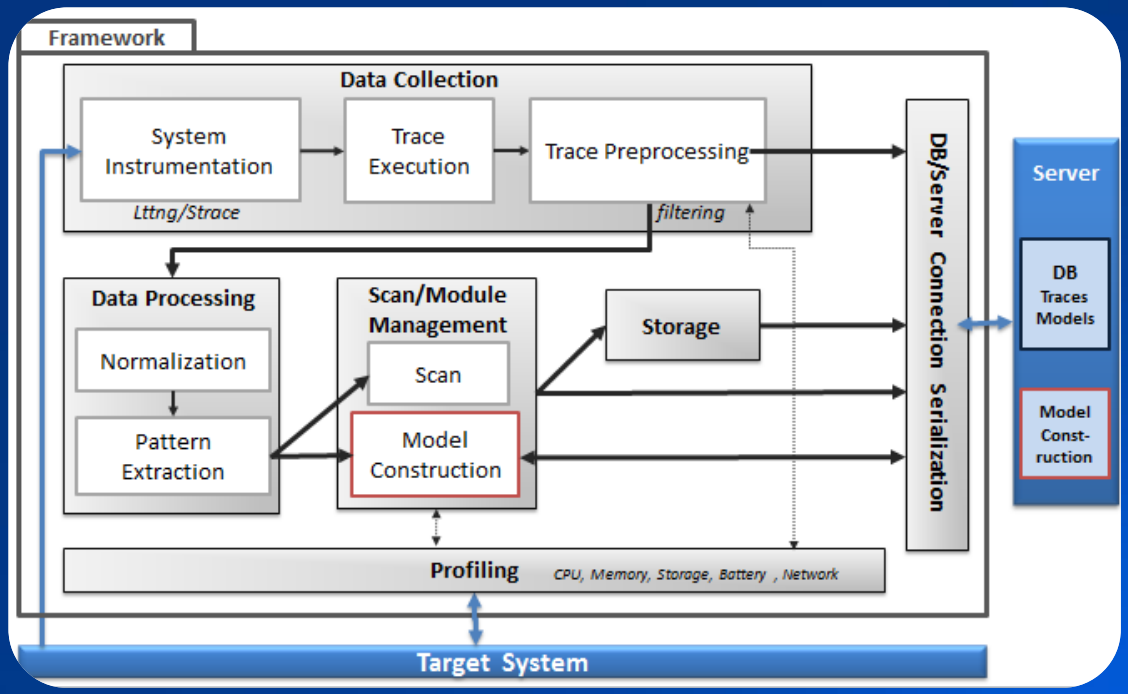




Varied-length N-grams



N-gram Tree


Framework
Data Collection
System Instrumentation
Trace Execution
Trace Preprocessing
Lttng/Strace
filtering
Data Processing
Normalization
Pattern Extraction
Scan/Module Management
Scan
Model Construction
Storage
Profiling
CPU, Memory, Storage, Battery , Network
Target System
DB/Server Connection Serialization
Server
DB Traces Models
Model Const-ruction

STORAGE

Storage: Compression

- Data compression works by eliminating statistical redundancy from the data.

- Benefits of higher compression density:
  - Storing more items in less space
  - Speeding up data transmission
  - Reducing data transfer fees and battery use for mobile

Storage:
Compression techniques

- Mainly they are 2 types of compression techniques:

1. **Lossy less Compression:** only an approximation of the original data are reconstructed from the compressed data. It is used in images, movies and sounds compression.

2. **Loss less Compression:** original data are perfectly reconstructed from the compressed data. It is used in text data, executable program compression.

   Example: Gzip, 7-zip, kzip, Zopfli

Storage:
Compression techniques

- Mainly they are 2 types of compression techniques:

1. **Lossy less Compression:** only an approximation of the original data are reconstructed from the compressed data. It is used in images, movies and sounds compression.

2. **Loss less Compression:** original data are perfectly reconstructed from the compressed data. It is used in text data, executable program compression.

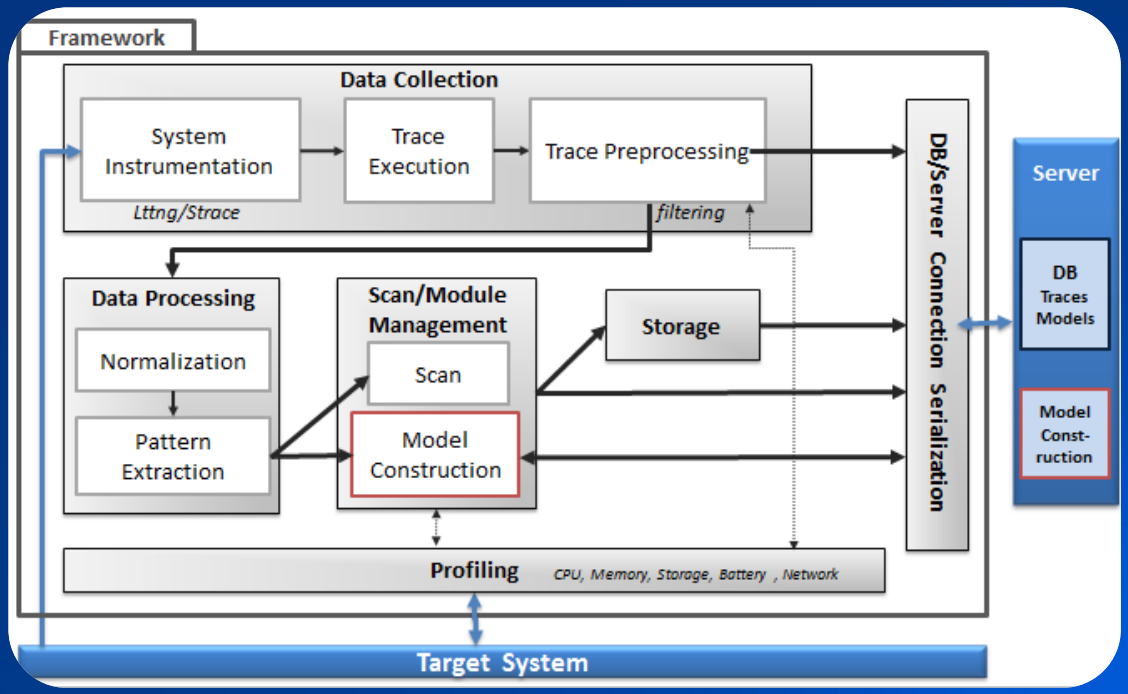
   Example: Gzip, 7-zip, kzip, Zopfli

Storage:
Zopfli compression algorithm

- Open source, released by Google in 2013
- General purpose compression algorithm
- Uses Lossless compression technique (useful for DATA compression)
- Compatible with zlib and gzip.
- Written in C for portability.
- It allows higher data density (it compresses ~ 5% better than any zlib-compatible compressor)
- It makes the compression a lot slower (~100x slower)
- The decompression speed is not affected

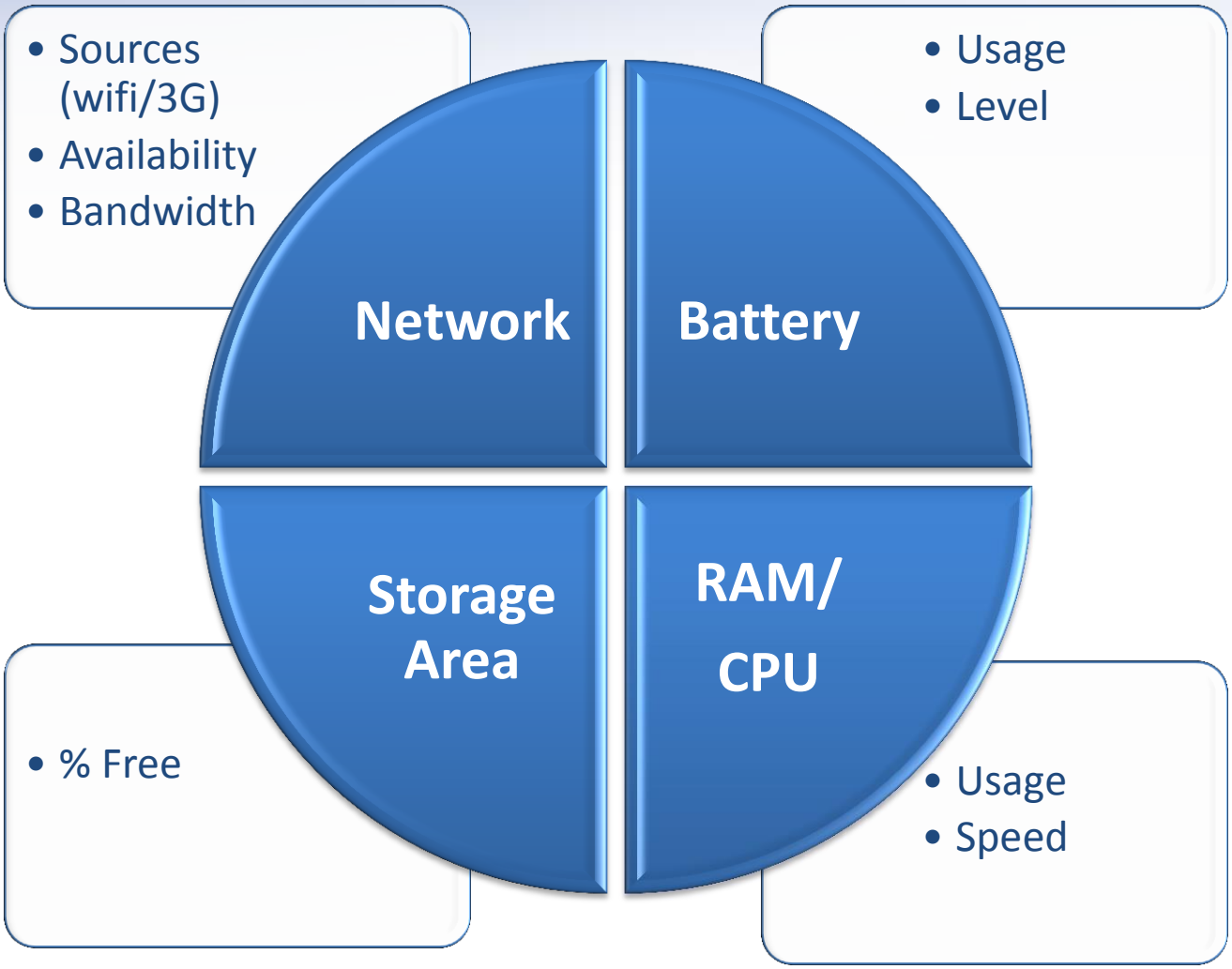
	gzip	7-Zip	kzip	Zopfli
Compressed Size	36,445,248 bytes	35,102,976 bytes	35,025,767 bytes	34,995,756 bytes
Compression Time	5.60 s	128 s	336 s	454 s
Decompress Time	934 ms	949 ms	937 ms	926 ms

SOURCE: James Sanders, “Google's Zopfli Compression Algorithm: Extract higher performance from your compressed files”, 2013, http://www.techrepublic.com/blog/the-enterprise-cloud/googles-zopfli-compression-algorithm-extract-higher-performance-from-your-compressed-files/


Framework
Data Collection
System Instrumentation
Trace Execution
Trace Preprocessing
Lttng/Strace
filtering
Data Processing
Normalization
Pattern Extraction
Scan/Module Management
Scan
Model Construction
Storage
Profiling
CPU, Memory, Storage, Battery , Network
Target System
DB/Server Connection Serialization
Server
DB Traces Models
Model Const-ruction

PROFILING

Profiling parameters



Scenarios and decisions

Trace management

network interface	Free memory space	Decisions
$B_{max} > \alpha_B$	----------	- Send current and compressed traces to the server - Update model
$B_{max} < \alpha_B$	$S_{free} < \alpha_S$	- Increase the threshold of the model "Varied-length N-grams" in order to reduce the size of the model to be saved. - Decrease the size of n-grams (window size) for lookahead and tree models.
$B_{max} < \alpha_B$	$S_{free} > \alpha_S$	- Save traces in the device - Compress the traces when they reach a certain number (the compression is slow but it saves more space and reduce the cost of data transfer and battery use)

Scenarios and decisions

Model and Scan management

Battery	RAM	CPU	Decisions
$B > \alpha_{Batt}$	$R < \alpha_{RAM}$	$C < \alpha_{CPU}$	**Scan using more than one model.** **Maximize accuracy :** Increase the size of n-grams (window) for lookahead and tree (ordered by increasing frequencies) models. Decrease the threshold of the model "Varied-length N-grams."
$B > \alpha_{Batt}$	$R > \alpha_{RAM}$	$C < \alpha_{CPU}$	**Scan using just one model.** **Minimize the amount of data being processed:** Decrease the size of n-grams (window) for lookahead and tree (ordered by increasing frequencies) models. Increase the threshold of the model "Varied-length N-grams." **Decrease depth analysis with the model tree:** During scanning with tree model, handles only a part of the tree n-gram (a sub-tree)
$B > \alpha_{Batt}$	----------	$C > \alpha_{CPU}$	**Scan using just one model.** **Minimize the amount of data being processed:** Decrease the size of n-grams (window) for lookahead and tree (ordered by increasing frequencies) models. Increase the threshold of the model "Varied-length N-grams." **Decrease depth analysis with the model tree:** During scanning with tree model, handles only a part of the tree n-gram (a sub-tree). **Minimize the number of treatments** Do not send traces to the server Do not compress the traces
$B < \alpha_{Batt}$	----------	----------	**Scan only** **Decrease depth analysis with the model tree:** During scanning with tree model, handles only a part of the tree n-gram (a sub-tree).

Project Summary

Designing a Trade-Off Between Usability and Security:

- Platform: Android
- Security module:
  - Data Collection → system calls
  - Data Processing
  - Scan/Model Management
    - Signature based detection VS Anomaly based detection
    - Anomaly based algorithms : Lookahead, Tree, Varied-length N-grams , Finite State Machines
- Storage module:
  - Compression techniques
  - Zopfli compression algorithm
- Profiling module:
  - Profiling parameters : Network status , Battery, RAM/CPU, Storage
  - Decisions


Any
Questions