Progress Report Meeting
May 2015

On device Anomaly Detection for resource-limited systems

Maroua Ben-Attia
Chamseddine Talhi, Abdelwahab Hamou-Lhadj, Babak Khosravifar

École de technologie supérieure (ÉTS)
Computer System Architecture Research Lab (LASI)
Department of software engineering and IT
Montreal, QC, Canada
Introduction: **Malware Evolution**

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

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

General-purpose small devices

- 2009: Psyb0t
  - Linux-based routers, DSL modems

- 2010: Stuxnet
  - Industrial control systems (ICS)

- 2012: DNSChanger
  - Computers and routers

- 2013: GPS attack
  - GPS based systems

- 2013: Linux. Darlooz
  - Cameras, set-top boxes

- 2014: The Moon
  - Linksys routers
Security Issue: Just in 2014!

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

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

Resource Limitations

Low power CPUs
- Lightweight processing
- Limited multitasking

Memory

Battery life

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>600 MHz</td>
<td>512 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Storage</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>microSD slot</td>
<td>Linux, Android</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>RAM</th>
<th>Storage</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>720MHz</td>
<td>256 MB</td>
<td>4GB microSD</td>
<td>Android, Linux</td>
</tr>
</tbody>
</table>
Objective

Security
- Detection rate
- FP/FN rate
- Real-time detection

Usability
- Battery life
- CPU usage
- Memory consumption
1. Benign applications that loads, for benign reasons, additional code that can be replaced with malicious ones by the attacker.

2. Malicious application that does not contain initially any clearly malicious code, but downloads additional faked code after being installed on a device.

# Intrusion Detection Techniques

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

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

Lookahead pairs

N-gram Tree

Varied-length N-grams

Finite State Machines
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

Angry birds space
Loads additional code to locate the device, steal contacts and send text messages.

Candy Star
Loads a shared library and DEX file
Read/modify/delete the contents of the SD card.

Ninja Chicken
Loads a shared library and DEX file
Read/modify/delete the contents of the SD card.
Read phone state + identify running applications.

http://contagiominidump.blogspot.fr/
https://www.virustotal.com/intelligence/
Experimental results

-Creating Normal profile-

- RAM Overhead

Finite State Machines

- RAM Overhead

Lookahead and Tree models

Varied-length N-grams

% of RAM

length of n-gram

% of RAM

Threshold
Experimental results
-Creating Normal profile-

- CPU Overhead

![Finite State Machines graph](image)

![Lookahead and Tree models graph](image)

![Varied-Length N-gram model graph](image)
Experimental results
-Creating Normal profile-

- **Storage Overhead**

![Finite State Machines Graph](image)

![Memory Overhead (7 traces) Graph](image)

![Varied-Length N-grams (7 traces) Graph](image)
Experimental results
- Scanning 1, 2 and 3 applications in parallel-

• RAM Overhead

Lookahead

N-Gram Tree

VL N-gram

Finite State Machines

% RAM

% RAM

% RAM

% RAM
Experimental results
- Scanning 1, 2 and 3 applications in parallel-

- **CPU Overhead**

  **Lookahead**

  ![Lookahead graph]

  **N-Gram Tree**

  ![N-Gram Tree graph]

  **VL N-gram**

  ![VL N-gram graph]

  **Finite State Machines**

  ![Finite State Machines graph]
Experimental results

Accuracy = \((TP+TN)/(TP+TN+FP+FN)\)
Experimental results

- Accuracy = \(\frac{(TP+TN)}{(TP+TN+FP+FN)}\)
Storage:
-Zopfli compression algorithm-

Size of Traces before and after compression with Zopfli

Size of Models before and after compression with Zopfli (with 500 Traces)
Profiling:
-Profiling parameters-

- Network
  - Sources (wifi/3G)
  - Availability
  - Bandwidth

- Battery
  - Usage
  - Level

- Storage Area
  - % Free

- RAM/CPU
  - Usage
  - Speed
### Profiling:

- **Trace management**

<table>
<thead>
<tr>
<th>network interface</th>
<th>Free memory space</th>
<th>Decisions</th>
</tr>
</thead>
</table>
| \( B_{max} > \alpha_B \) | \(-\) | - 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) |
# Profiling:
- Model and Scan management -

<table>
<thead>
<tr>
<th>Battery</th>
<th>RAM</th>
<th>CPU</th>
<th>Decisions</th>
</tr>
</thead>
</table>
| $B > \alpha_{Batt}$ | $R < \alpha_{RAM}$ | $C < \alpha_{CPU}$ | **Scan using more than one model.**  
Maximize accuracy:  
Increase the size of n-grams (window) for Tree model,  
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 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 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). |
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, FSM
- Storage module:
  - Zopfli compression algorithm
- Profiling module:
  - Profiling parameters: Network status, Battery, RAM/CPU, Storage
Designing a Trade-Off Between Usability and Security:

- **Platform**: Android, **Linux**
- **Security module**:
  - Data Collection $\rightarrow$ system calls, **LTTng**
  - Data Processing
  - Scan/Model Management
    - Signature based detection VS Anomaly based detection
    - Anomaly based algorithms: Lookahead, Tree, Varied-length N-grams, FSM, **other algorithms**
- **Storage module**:
  - Zopfli compression algorithm
- **Profiling module**:
  - Profiling parameters: Network status, Battery, RAM/CPU, Storage
  - **Dynamic decision maker**
    - Monitoring system behavior and selecting the best anomaly detection Algorithm.