Unsupervised Clustering
Under Temporal Feature Volatility
in Network Stack Fingerprinting

Zain Shamsi and Dmitri Loguinov

Internet Research Lab
Department of Computer Science and Engineering
Texas A&M University

June 16, 2016
Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison
Introduction

• Classification of large networking datasets is an important topic

• This requires a database of known signatures for a classifier to match against
  – Made of different specimens found in the wild

• Generally, these databases are created manually and must be kept up-to-date
  – Slow process that usually lags behind the discovery of a new specimen
  – Is prone to error, heuristic decisions and poor repeatability
Introduction

- The performance of a classifier depends heavily on the makeup of the underlying database

- E.g., imagine two databases of animal images
  - Database A contains two pictures, 1 rabbit and 1 cat
  - Database B contains 5 rabbits, 5 hares and 5 cats

- Then a classifier trying to identify animals could have:
  - 99% accuracy and a quick runtime using A
  - 40% accuracy and slower performance using B since rabbits/hares are similar, and more comparisons are needed

- Databases should only keep classes which can be differentiated, and drop duplicate specimens
Introduction

• However, data gathered in the wild has more complex elements in reality
  – Each image captured could additionally be disturbed by noise such as lens distortion or motion blur
  – Let's say this is given by some noise model $\mathcal{X}$

• We now plot the features of animals in database B
  – Under a small noise radius, we cannot tell hares and rabbits apart
  – With larger noise, we may not be able to distinguish cats either

• We want a database that can classify specimens correctly under given noise $\mathcal{X}$
• We introduce a framework to describe a classifier and its database – its **dimension** $d(1-\epsilon, \mathcal{X})$
  - This means that the classifier can differentiate between $d$ signatures with probability $1-\epsilon$ under noise model $\mathcal{X}$
  - The database for this classifier must then contain exactly $d$ signatures

• We can use this metric to build databases as well as compare the power of classifiers
  - In our example, if we had almost no noise, our dimension $d = 15$
  - Under small noise radius our dimension $d = 2$
  - Under large noise radius it would be reduced to $d = 1$
Introduction

• We define features to be *volatile* if \( \mathcal{X} \) can distort them, or *deterministic* otherwise

• Our interest is in the problem of specimen *separation*  
  – Deciding whether two observations are too similar for database inclusion

• Separation for deterministic features can be simple  
  – E.g. pick every unique combination as a different signature

• For volatile features, we require a more sophisticated solution  
  – Our goal is to solve this problem  
  – We apply our method to OS fingerprinting
Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison
Consider a measurement of production systems $S_1, \ldots, S_N$ in the wild
- This produces a set of feature vectors

Initially, all vectors are added into one large database
- May contain several duplicates

We want to determine the separability of the specimens in this database according to noise $\mathcal{N}$

We introduce an algorithm called Plata, which refines this database and determines its dimension
- La Plata, Argentina was the first city to use fingerprint databases in 1892
The Plata Algorithm

• Assume $\Delta$ is a vector of features and $\Delta'$ is feature vector in the database

• Any given classifier will have a function $p(\Delta|\Delta', \mathcal{X})$
  – Produces classification probability for $\Delta'$ becoming $\Delta$ under $\mathcal{X}$

• Plata uses the classifier to compare all database signatures to each other under simulated noise

• This constructs a confusion matrix $M$, where each cell is calculated as:
  $$M_{ij} = E[p(\Delta_i + \theta|\Delta_j, \mathcal{X})]$$
  – $\theta$ is random observation noise driven by model $\mathcal{X}$
  – Generally, Monte-Carlo simulations can be used to determine $M_{ij}$
The Plata Algorithm

• Note that the diagonal of $M$ signifies the probability of self classification
  – We want signatures that can be matched back to themselves with probability $1 - \epsilon$, i.e.
    $$E[p(\Delta_i + \theta|\Delta_i, X)] \geq 1 - \epsilon$$

• Plata iteratively eliminates signatures from $M$ starting with the lowest diagonal value
  – Keeps going until all diagonal values are $\geq 1 - \epsilon$


• Instead of re-running expensive Monte-Carlo simulations after each removal, infer the next matrix
  – Removing system $k$ distributes its classifications proportionally amongst the remaining candidates

\[ M_{ij} = M_{ij} + \frac{M_{ij}}{1 - M_{ik}} M_{ik} \]

• $1 - \epsilon$ can be a tuning parameter
  – Higher means smaller database, more uniqueness
  – Lower means larger database, higher risk of duplicates

• For labeling, we only need to know labels for a subset
  – Attach labels to the final clusters based on membership
  – The paper gives more details
Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison
OS Fingerprinting Database

- OS Fingerprinting is a technique to determine the OS of a remote host
- We split previous work by types of features used
  - Packet timing techniques
    - Use volatile and deterministic features
    - Low overhead and not intrusive, suitable for large scans
OS Fingerprinting Database

- Deterministic features are values from packet headers
  - TCP Window, IP TTL, TCP Options etc.

- Volatile features are SYN-ACK Retransmission Timeouts (RTOs)
  - Volatile due to network queuing delays

<table>
<thead>
<tr>
<th>Label</th>
<th>Win</th>
<th>TTL</th>
<th>DF Flag</th>
<th>TCP Options</th>
<th>MSS</th>
<th>SYN-ACK RTOs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7</td>
<td>8192</td>
<td>128</td>
<td>1</td>
<td>MNWST</td>
<td>1460</td>
<td>3  6  12</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>5792</td>
<td>64</td>
<td>1</td>
<td>MSTNW</td>
<td>1460</td>
<td>4.3 6 12 24 48</td>
</tr>
</tbody>
</table>
OS Fingerprinting Database

- We scan our campus network on port 80 and gather feature vectors from ~10K systems
- We build databases using three previous packet timing classifiers: RING, Snacktime, and Hershel
- Our data is initially separated using the deterministic features of each classifier
  - Forms several clusters that need to be further separated
- Plata then separates each cluster’s volatile features
  - The noise model $\mathcal{X}$ is simulated as a packet queue that adds exponential random one way delay to each packet
  - We also set $1-\epsilon = 0.8$ to ensure sufficient duplicate elimination
OS Fingerprinting Database

- RTOs are not only volatile, but also random due to OS timers
  - Example shows two Xerox printers
- Most OS signatures have 3-5 RTOs, some with 20
  - Doing this separation manually is practically impossible
- Example: Plata separates 2 Windows 2003 signatures
OS Fingerprinting Database

- We obtain these dimensions for the previous methods:

<table>
<thead>
<tr>
<th>Grouping</th>
<th>R</th>
<th>ING</th>
<th>Snacktime</th>
<th>Hershel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic Features Only</td>
<td>28</td>
<td>209</td>
<td>344</td>
<td></td>
</tr>
<tr>
<td>Volatile Features Only</td>
<td>23</td>
<td>52</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>Both</td>
<td>39</td>
<td>260</td>
<td>398</td>
<td></td>
</tr>
</tbody>
</table>

- This allows us to directly compare the power of each classifier in separating the same dataset
  - Hershel is clearly the most powerful method

- For labeling, we use banner grabbing
  - Use simhash to form cluster all hosts of similar OS
  - Match the label clusters to database, see paper for details
OS Fingerprinting Database

- Running Monte-Carlo simulations to determine $E[p(\Delta_i + \theta \mid \Delta_j, X)]$ for each cell can be time consuming
  - Our dataset with 10K hosts takes over 24 hours
  - We want easy repeatability and scalability to larger networks

- We can optimize Plata when the noise model allows the expectation to be calculated directly

- Since Hershel calculates probabilities, we can derive the expected similarity between $\Delta_i$ and $\Delta_j$ under $\theta$
  - This gives us a closed form for each cell of the Plata matrix
  - The paper develops the model, omitted here
OS Fingerprinting Database

• We also improve on Hershel’s classifier, calling the new method Hershel+
  – We switch from using jitter to one-way delay, see paper for a detailed explanation
  – Simulations show it performs up to 10% better than Hershel in RTO classification

• Hershel+ now produces 420 signatures (as opposed to Hershel’s 398) for our OS database
  – Improved dimension also confirms its superiority

• Plata’s closed form matches Monte-Carlo results, reduces runtime from 24+ hours to 12 minutes
Agenda

- Introduction
- The Plata Algorithm
- OS Fingerprinting Database
- Internet Scan
- Nmap Comparison
Internet Scan

• We performed a port-80 SYN scan of the Internet in July 2015
  – 2.7B IPs in 6 hours, 125K packets / sec
  – 66.4 million hosts responded, almost double the last study

• On the Internet, observed signatures may undergo several changes
  – Deterministic features might be changed by users, firewalls
  – Packet loss and network delays may change RTOs

• We classify all hosts using Hershel+ as the classifier and our database of 420 signatures built using Plata
  – This is the largest study performed with so many signatures and the first with an automated database
Internet Scan

- Linux most popular, for webservers, also a lot of embedded devices
- Comparing with previous results
  - Linux/embedded have doubled
  - Windows has stayed almost the same
  - Misc has lost 69% of its membership
- Paper shows additional results and distributions

<table>
<thead>
<tr>
<th>Family</th>
<th>OS</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Ubuntu/Redhat/CentOS</td>
<td>14.55 M</td>
</tr>
<tr>
<td></td>
<td>Ubuntu/Redhat/SUSE</td>
<td>2.62 M</td>
</tr>
<tr>
<td></td>
<td>Ubuntu/Debian/Redhat</td>
<td>2.38 M</td>
</tr>
<tr>
<td>Embedded</td>
<td>3Com Routers</td>
<td>2.66 M</td>
</tr>
<tr>
<td></td>
<td>Dell Laser/Xerox Printers</td>
<td>1.98 M</td>
</tr>
<tr>
<td></td>
<td>Cisco Embedded</td>
<td>1.86 M</td>
</tr>
<tr>
<td>Windows</td>
<td>Windows 7/2008/2012</td>
<td>2.18 M</td>
</tr>
<tr>
<td></td>
<td>Windows XP / 2003</td>
<td>822 K</td>
</tr>
<tr>
<td></td>
<td>Windows 2000 / 2003</td>
<td>791 K</td>
</tr>
<tr>
<td>Misc</td>
<td>FreeBSD</td>
<td>480 K</td>
</tr>
<tr>
<td></td>
<td>FreeBSD</td>
<td>107 K</td>
</tr>
<tr>
<td></td>
<td>Novell Netware</td>
<td>37 K</td>
</tr>
</tbody>
</table>
Agenda

• Introduction
• The Plata Algorithm
• OS Fingerprinting Database
• Optimization
• Internet Scan
• Nmap Comparison
Nmap Comparison

- We Nmap 1% of responsive hosts from a separate machine at the same time as our scan
- Nmap sends 10 types of probes to the target host
  - Its features are based on existence of responses and values from header fields of each response
- Hershel+ and Nmap find agreement in the OS family in the majority of cases
  - Comparison on the exact OS/device is made difficult due to a large variety of OS names, especially for embedded devices
- However, some cases have glaring disagreements
  - We focus on four such cases between the two classifiers
Nmap Comparison

• Sampled hosts (S1-S4) and their Hershel+ matches:

<table>
<thead>
<tr>
<th>System / Signature</th>
<th>Win</th>
<th>TTL</th>
<th>TCP Options</th>
<th>MSS</th>
<th>SYN-ACK arrival timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>8192</td>
<td>128</td>
<td>MNWST</td>
<td>1464</td>
<td>0.22 3.22 9.22 21.22</td>
</tr>
<tr>
<td>S2</td>
<td>8192</td>
<td>64</td>
<td>MNWST</td>
<td>1460</td>
<td>0.18 3.17 9.17</td>
</tr>
<tr>
<td>Windows 7/2008 R2</td>
<td>8192</td>
<td>128</td>
<td>MNWST</td>
<td>1464</td>
<td>0.00 2.99 9.00 21.00</td>
</tr>
<tr>
<td>S3</td>
<td>16384</td>
<td>128</td>
<td>MNWNNTNNS</td>
<td>1460</td>
<td>0.21 2.67 9.22</td>
</tr>
<tr>
<td>S4</td>
<td>16384</td>
<td>128</td>
<td>MNWNNTNNS</td>
<td>1370</td>
<td>0.21 3.07 9.63</td>
</tr>
<tr>
<td>Windows 2000/2003</td>
<td>16384</td>
<td>128</td>
<td>MNWNNTNNS</td>
<td>1380</td>
<td>0.00 2.65 9.21</td>
</tr>
</tbody>
</table>

• Hershel+ overcomes packet loss and changes in the TTL/MSS value

• Classifications make sense and are not in doubt
## Nmap Comparison

- Nmap signatures of the same four systems:

<table>
<thead>
<tr>
<th>System / Signature</th>
<th>Win</th>
<th>TTL</th>
<th>TCP Options</th>
<th>Response TCP open port</th>
<th>Response TCP closed port</th>
<th>Response UDP / ICMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>8192</td>
<td>128</td>
<td>MNWST</td>
<td>0000</td>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>Tomato 1.28 (Router firmware)</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>0000</td>
<td>111</td>
<td>01</td>
</tr>
<tr>
<td>S2</td>
<td>8192</td>
<td>64</td>
<td>MNWST</td>
<td>1000</td>
<td>100</td>
<td>11</td>
</tr>
<tr>
<td>OpenBSD 4.3 (8 years old)</td>
<td>00</td>
<td>00</td>
<td>MNWNNNTNNS</td>
<td>1000</td>
<td>100</td>
<td>11</td>
</tr>
<tr>
<td>S3</td>
<td>16384</td>
<td>128</td>
<td>MNWNNNTNNS</td>
<td>0000</td>
<td>000</td>
<td>01</td>
</tr>
<tr>
<td>S4</td>
<td>16384</td>
<td>128</td>
<td>MNWNNNTNNS</td>
<td>0000</td>
<td>111</td>
<td>01</td>
</tr>
<tr>
<td>D-Link DWL624 (2003 54g router)</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>0000</td>
<td>111</td>
<td>01</td>
</tr>
</tbody>
</table>

- Nmap allows null features that match everything, weighs heavily on whether response was received.
Nmap Comparison

• Nmap has no provisions for feature volatility, especially in its response vector
  – Becomes an issue when middleboxes block its packets

• The Tomato signature was matched to 21% of hosts in the entire Nmap data
  – Very doubtful for so many hosts to run this firmware

• Nmap’s results are questionable on public networks where IDS, packet filters and firewalls are abundant

• Future work will determine the dimension of Nmap
Conclusion

• Introduced Plata – an algorithm for separating observed samples under feature volatility

• Applied Plata to OS classification and automatically built a database of 420 OS signatures

• Developed an improved state-of-the-art classifier, Hershel+, and used it to classify every webserver on the Internet

• Compared our results with Nmap, showing that disagreements tend to favor the Hershel+ result

Thank you!