Generative Intrusion Detection and Prevention on Data Stream

HyungBin Seo and MyungKeun Yoon
Kookmin University

USENIX Security 2023
Motivation

• Different Types of Intrusion Detection
  • Persistent zero-day attack, signature-group generation, evidence support

<table>
<thead>
<tr>
<th>Type</th>
<th>Accuracy</th>
<th>Automation</th>
<th>Zero-day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misuse-detection (signature)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signature generation [9, 48]</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Signature-group generation (GIPS)</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Anomaly-detection (ML)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervised learning [19, 52]</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Unsupervised learning [54, 62]</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>
Motivation

• Previous signature-generation schemes [9], [24], [35], [39], [48]
  • high-volume attacks such as DDoS, worms

Attack data ratio = 8/10 → Big Group

Generated Signature

Suffix tree, Hash table, …
Motivation

• Previous signature-generation schemes [9], [24], [35], [39], [48]
  • high-volume attacks such as DDoS, worms

<table>
<thead>
<tr>
<th>signature</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;getRunti...&quot;</td>
<td>8</td>
</tr>
<tr>
<td>&quot;exec().&quot;</td>
<td>6</td>
</tr>
<tr>
<td>&quot;POST&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;HTTP/1.1&quot;</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Generated Signatures

Dataset with high attack ratio

Hash table
Motivation

• Previous schemes would cause false-positives for **persistent zero-day attacks**.
Motivation

• Previous schemes would cause false-positives for **persistent zero-day attacks**.

<table>
<thead>
<tr>
<th>Signature</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;HTTP/1.1&quot;</td>
<td>8</td>
</tr>
<tr>
<td>&quot;POST&quot;</td>
<td>5</td>
</tr>
<tr>
<td>&quot;/index..&quot;</td>
<td>3</td>
</tr>
<tr>
<td>&quot;application&quot;</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Dataset with low attack ratio | Hash table | Generated Signatures
Motivation

- **GIPS**
  - Generative Intrusion detection and Prevention on data Stream
  - Data analysis tool

Attack data ratio = 2/10 → Still Big Group

MinHashed Virtual Vector → {"getRuntime().exec", "/bin/busybox"} → Generated Signature-Groups

7
Overall Process of GIPS

Streaming Data → MV2 → JIG → SG2 → AWL → "getRuntime() .exec", "/bin/busybox"

- MV2: Minhashed Virtual-Vector
- JIG: Jaccard-Index Grouping
- SG2: Signature-Group Generation
- AWL: Automatic WhiteListing

Big-group identification

Signature-group generation

HTTP/1.1…
"getRuntime…"
"POST /index…"
"POST /index2.."
"getRuntime…"

“..HTTP/1…”
“HTTP… abed”
“POST /assets…”
“…application/..”
“application/…”
Overall Process of GIPS

Streaming Data → MV2 → JIG → SG2 → AWL → "getRuntime().exec", "/bin/busybox"
Overall Process of GIPS

Signature-group generation

Streaming Data ➔ MV2 ➔ JIG ➔ SG2 ➔ AWL ➔ “getRuntime().exec”, “/bin/busybox”
MV2: Minhashed Virtual-Vector

(a) Content-Defined Chunking

```
"HTTP" "root" "admin" "1234@
```

{k=3 $h_0(\cdot), h_1(\cdot), h_2(\cdot)$, …}

(b) Minhashed Virtual-Vector

```plaintext
\left\{ \min_{x \in \text{chunks}} \{h_0(x)\} \right\} \mod m
\left\{ \min_{x \in \text{chunks}} \{h_2(x)\} \right\} \mod m
0 1 0 1 1 0 0 0
```

MV2: Minhashed Virtual-Vector

\{"HTTP", "root", "admin", "1234@"\}

\[
h_0("HTTP") = 71
\]
\[
h_0("root") = 48
\]
\[
h_0("admin") = 9
\]
\[
h_0("1234@") = 57
\]

\[
h_1("HTTP") = 87
\]
\[
h_1("root") = 11
\]
\[
h_1("admin") = 95
\]
\[
h_1("1234@") = 24
\]

\[
m_\text{min}_{x \in \text{chunks}} h_0(x) = 9
\]

\[
m_\text{min}_{x \in \text{chunks}} h_1(x) = 11
\]

9 mod 8

0100000000

11 mod 8

010100000
**JIG: Jaccard-Index Grouping**

- Big-group of $d_i$ from data stream

$$sg(d_i) = \{d_j | j(d_i, d_j) > \theta_J \} \cup \{d_i\} \quad b_r(d_i) = \frac{|sg(d_i)|}{i} > \theta_B$$

As their similarity to $d_5$ is greater than $\theta_J$,

$d_5$ belongs to a Big-group

Naïve clustering on data stream requires a large time & space-complexity!
JIG: Jaccard-Index Grouping

d0 = “GET /index.html HTTP/1.1”
d1 = “HTTP rootadmin1234@”
d2 = “GET / HTTP/1.1 admin”
d3 = “HTTP root directory”
d4 = “GET root12, admin123@”
d5 = “GET root12, admin123@”

(a) Data stream

(b) Counter array

\[ J = 0.6 \]
JIG: Jaccard-Index Grouping

\( d_0 \) = “GET /index.html HTTP/1.1”
\( d_1 \) = “HTTP rootadmin1234@”
\( d_2 \) = “GET / HTTP/1.1 admin”
\( d_3 \) = “HTTP root directory”
\( d_4 \) = “GET root12, admin123@”
\( d_5 \) = “GET root12, admin123@”

\[ \theta_J = 0.6 \]

(a) Data stream

(b) Counter array
JIG: Jaccard-Index Grouping

\[ \theta_j = 0.6 \]

\( d_0 = \text{“GET /index.html HTTP/1.1”} \)

\( d_1 = \text{“HTTP rootadmin1234@”} \)

\( d_2 = \text{“GET / HTTP/1.1 admin”} \)

\( d_3 = \text{“HTTP root directory”} \)

\( d_4 = \text{“GET root12, admin123@”} \)

\( d_5 = \text{“GET root12, admin123@”} \)

(a) Data stream

(b) Counter array
JIG: Jaccard-Index Grouping

\[ J = 0.6 \]

\( d_0 = \text{“GET /index.html HTTP/1.1”} \)
\( d_1 = \text{“HTTP rootadmin1234@”} \)
\( d_2 = \text{“GET / HTTP/1.1 admin”} \)
\( d_3 = \text{“HTTP root directory”} \)
\( d_4 = \text{“GET root12, admin123@”} \)
\( d_5 = \text{“GET root12, admin123@”} \)

(a) Data stream

(b) Counter array

\( \theta_J = 0.6 \)
JIG: Jaccard-Index Grouping

\[ \theta_J = 0.6 \]

(a) Data stream

\[
\begin{align*}
d0 &= \text{"GET /index.html HTTP/1.1"}
d1 &= \text{"HTTP rootadmin1234@"}
d2 &= \text{"GET / HTTP/1.1 admin"}
d3 &= \text{"HTTP root directory"}
d4 &= \text{"GET root12, admin123@"}
d5 &= \text{"GET root12, admin123@"}
\end{align*}
\]

(b) Counter array
**JIG: Jaccard-Index Grouping**

- $d_0$ = “GET /index.html HTTP/1.1”
- $d_1$ = “HTTP rootadmin1234@”
- $d_2$ = “GET / HTTP/1.1 admin”
- $d_3$ = “HTTP root directory”
- $d_4$ = “GET root12, admin123@”
- $d_5$ = “GET root12, admin123@”

(a) Data stream

<table>
<thead>
<tr>
<th>MV</th>
<th>1</th>
<th>4</th>
<th>0</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
</table>

(b) Counter array

Mean + 6 * Sigma = 2.8
(Iterative Outlier Removal Algorithm)

- $S = \{d_4\}$
- Big-group
- $2 / 3 > \theta_f$ (Overlap ratio > $\theta_f$)
- $\theta_f=0.6$
- Big-counter
- Counter > 2.8
JIG: Jaccard-Index Grouping

d0 = “GET /index.html HTTP/1.1”
d1 = “HTTP rootadmin1234@”
d2 = “GET / HTTP/1.1 admin”
d3 = “HTTP root directory”
d4 = “GET root12, admin123@”
d5 = “GET root12, admin123@”

(a) Data stream

S={d4, d5}

Mean + 6 * Sigma = 3.1
(Iterative Outlier Removal Algorithm)

(b) Counter array

2 / 3 > θ_j
(Overlap ratio > θ_j)
θ_j=0.6

Counter > 3.1

Big-counter

Big-group
JIG: Jaccard-Index Grouping

**Big-group identification**
- small fixed amount of memory
- constant number of hash operations

(a) Data stream

\[ S = \{d_4, d_5\} \]

(b) Counter array

\[ \text{Mean} + 6 \times \text{Sigma} = 3.1 \]

(Iterative Outlier Removal Algorithm)
SG2: Signature-Group Generation

• Signature-group generation from \( S \)
  • Clustering & Signature extraction (Triple-Heavy-Hitter*)
  • Other schemes can be used instead

Clustering (DBSCAN)

*THH

```
/bin/busybox”, “echo”
```

*THH

```
“getRuntime().exec”, “touch /tmp/test.txt”
```

AWL: Automatic WhiteListing

- AWL
  - (SG2 THH) – (Global THH)

(a) SG2

{“HTTP/1.1”, “application”, “/bin/busybox”, “getRuntime().exec”}

(b) AWL

{“HTTP/1.1”, “POST”, “application”}

(c) GIPS

{“/bin/busybox”, “getRuntime().exec”}
Experiments

• Dataset
  • SIM1-1 ~ SIM3-3
    • Simulated dataset for big-group identification
    • [https://tinyurl.com/2gfz4f7v](https://tinyurl.com/2gfz4f7v)
  • IoT1~8
  • IDS1~8
  • ISP1~3
    • Private dataset
    • Suspicious packets from IPS
Experiments

- SIM1-1 ~ SIM3-3
  - [https://tinyurl.com/2gfz4f7v](https://tinyurl.com/2gfz4f7v)

Figure 6: Precision (P) and Recall (R) of JIG for SIM datasets with different big-group ratios, $k$, and $m$. 
Experiments

- IoT1~8
  - IoT23, [https://www.stratosphereips.org/datasets-iot23](https://www.stratosphereips.org/datasets-iot23)
  - Table4 for Precision & Recall

![Graph showing F1-Score for IoT datasets](image)

*Figure 7: F1-Score of GIPS, THH [9], Earlybird [48], and Polygraph [39] for IoT datasets.*
Experiments

• IoT1~8
  • IoT23, https://www.stratosphereips.org/datasets-iot23
  • Table4 for Precision & Recall

GIPS $\approx$ THH
Attack ratio = 0.99

No repetitive contents

Figure 7: F1-Score of GIPS, THH [9], Earlybird [48], and Polygraph [39] for IoT datasets.
Experiments

- IDS1~8
  - Table5 for Precision & Recall

```
"/dv/vulnerabilities", "Cookie: security=low"

"GET / HTTP/1.0


```

Figure 10: F1-Score of GIPS, THH [9], Earlybird [48], and Polygraph [39] for IDS datasets.
Experiments

• **IDS1~8**
  - Table5 for Precision & Recall

Figure 10: F1-Score of GIPS, THH [9], Earlybird [48], and Polygraph [39] for IDS datasets.
Experiments

- ISP1~3
  - Private dataset
  - Table 6 for Precision & Recall

Table 7: GIPS Signatures and Attacks for ISP Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Signature</th>
<th>Related Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISP1</td>
<td><code>getRuntime().exec(&quot;touch /tmp/test.txt&quot;,&quot;/bin/busybox</code></td>
<td>CVE-2022-22963 [40]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CVE-2017-17215 [37]</td>
</tr>
<tr>
<td>ISP2</td>
<td><code>0002736c0000ff</code></td>
<td>CVE-2010-1574 [36]</td>
</tr>
<tr>
<td>ISP3</td>
<td><code>\x00\x00\x00DHIP\x00...&quot;Port&quot; : 37777,&quot;RemoteVideo...</code></td>
<td>Reflection attack [41]</td>
</tr>
</tbody>
</table>
Experiments

• Processing time and memory usage
Conclusion

• A new generative IDS/IPS to mitigate persistent zero-day attacks
• Fast and compact identification of big-groups from data streams
• Automatic generation of robust group-signatures
• Work from a high to a low attack ratio, for example from 0.99 to 0.001

• Limitations
  • Encrypted data, repetitive contents
  • Data analysis tool rather than real-time system
Thank You