SSH Compromise Detection using NetFlow/IPFIX

Rick Hofstede, Luuk Hendriks
“51 percent of respondents admitted that their organizations have already been impacted by an SSH key-related compromise in the last 24 months.”

–Ponemon 2014 SSH Security Vulnerability Report
SSH Compromise Detection using NetFlow/IPFIX

Rick Hofstede, Luuk Hendriks
SSH attacks

Fig. 2. Temporal visualization of a brute-force SSH scan (a) and variation of packets per flow during the scan (b).

Adifferent viewpoint is given by Figure 2 (a). Each mark in the graph either represents a malicious connection from the attacker to a victim or a response from the victim back to the attacker. The y-axis gives the 65,535 possible destination addresses in the university network. We identify three attack phases. During the scanning phase (first 1000 seconds), the attacker performs a sequential SSH scan spanning over the entire network address space. In this phase, the attacker gathers information on which hosts run a vulnerable SSH service. Only few victims respond to the attack. Once this phase is completed, the attacker initiates a brute-force user/password guessing attack (brute-force phase). In this phase, only a small subset of the hosts in...
SSH attacks

![SSH Attacks Diagram]

- **Scan**
- **Start**
- **Brute-force**
- **Compromise**
- **End**
SSH attacks

- SSH intrusion detection on end hosts is hardly scalable
- Network-based approaches exist, but only inform security operators about the presence of attacks
We perform compromise detection.
We perform **compromise detection.**

All **flow-based.**
SSH attacks

Fig. 2. Temporal visualization of a brute-force SSH scan (a) and variation of packets per flow during the scan (b)

Ad different viewpoints with the attacks given by Figure 2 (a). Each mark in the graph either represents a malicious connection from the attacker to a victim or the answer in a connection from the victim back to the attacker. The y-axis gives the 65,535 possible destination addresses in the university network. We identify three attack phases. During the scanning phase (first 1000 seconds), the attacker performs a sequential SSH scan spanning over the entire network address space. In this phase, the attacker gathers information on which hosts run a vulnerable SSH service. Only few victims respond to the attack. Once this phase is completed, the attacker initiates a brute-force user/password guessing attack (brute-force phase). In this phase, only a small subset of the hosts in
SSH attacks

![Graph showing SSH attacks](image)

Fig. 2. Temporal visualization of a brute-force SSH scan (a) and variation of packets per flow during the scan (b).

Advantages with the attack shown in Figure 2 (a). Each mark in the graph either represents a malicious connection from the attacker to a victim or an answer in the connection from the victim back to the attacker. The y-axis gives the 65,535 possible destination addresses in the university network. We identify three attack phases. During the scanning phase (first 1000 seconds), the attacker performs a sequential SSH scan spanning over the entire network address space. In this phase, the attacker gathers information on which hosts run a vulnerable SSH service. Only few victims respond to the attack. Once this phase is completed, the attacker initiates a brute-force user/password guessing attack (brute-force phase). In this phase, only a small subset of the hosts in...
A bit of history...
A bit of history…

- **SSHCure 1.0 (June ’12):**
  - Purely deviation-based compromise detection

- **SSHCure 2.0 (May ’13):**
  - Notifications, database maintenance, performance profiling, …
A bit of history...

Fig. 2. Temporal visualization of a brute-force SSH scan (a) and variation of packets per flow during the scan (b)

Adifferent view with the attack sigiven by Figure 2 (a). Each mark in the graph either represents a malicious connection from the attacker to a victim or an answer in a connection from the victim back to the attacker. The y-axis gives the 65,535 possible destination addresses in the university network. We identify three attack phases. During the scanning phase (first 1000 seconds), the attacker performs a sequential SSH scan spanning over the entire network address space. In this phase, the attacker gathers information on which hosts run a vulnerable SSH service. Only few victims respond to the attack. Once this phase is completed, the attacker initiates a brute-force user/password guessing attack (brute-force phase). In this phase, only a small subset of the hosts in
A bit of history…

• SSHCure 1.0 (June ’12):
  • Purely deviation-based compromise detection

• SSHCure 2.0 (May ’13):
  • Notifications, database maintenance, performance profiling, …
Recent/upcoming releases
Recent/upcoming releases

- SSHCure 2.4 (July ’14):
  - New compromise detection algorithm (CCR paper release), based on ‘action upon compromise’

- SSHCure 3.0 (January ’14):
  - New frontend, ingress vs. egress attacks
Recent/upcoming releases

SSH Compromise Detection using NetFlow/IPFIX.
In: ACM SIGCOMM Computer Communication Review, October 2014
Recent/upcoming releases

SSH Compromise Detection using NetFlow/IPFIX.
In: ACM SIGCOMM Computer Communication Review, October 2014
Recent/upcoming releases

SSH Compromise Detection using NetFlow/IPFIX.
In: ACM SIGCOMM Computer Communication Review, October 2014
Recent/upcoming releases

- SSHCure 2.4 (July ’14):
  - New compromise detection algorithm (CCR paper release), based on ‘action upon compromise’

- SSHCure 3.0 (January ’14):
  - New frontend, ingress vs. egress attacks
SSH Cure
Validation approach

- Ground truth: `sshd` logs from 93 honeypots, servers and workstations, divided over two datasets:
  - Dataset 1 — easy targets
  - Dataset 2 — more difficult targets

<table>
<thead>
<tr>
<th></th>
<th>Honeypots</th>
<th>Servers</th>
<th>Workstations</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>636</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>0</td>
<td>76</td>
<td>4</td>
<td>10353</td>
</tr>
</tbody>
</table>
SSH Cure
Validation results

• Evaluation metrics:
  • TP / FP — correct / false identification of incident
  • TN / FN — correct / false identification of non-incident

• Detection accuracy close to 100%

<table>
<thead>
<tr>
<th></th>
<th>TPR</th>
<th>TNR</th>
<th>FPR</th>
<th>FNR</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>0,692</td>
<td>0,921</td>
<td>0,079</td>
<td>0,308</td>
<td>0,839</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>—</td>
<td>0,997</td>
<td>0,003</td>
<td>—</td>
<td>0,997</td>
</tr>
</tbody>
</table>
SSH Cure

Deployment

- SSH Cure is open-source and actively developed
  - Download counter SourceForge (Dec. ’14): 3k
  - Recently moved to GitHub (summer ’14)
- Tested in several nation-wide backbone networks
- Many successful deployments already:
  - Web hosting companies
  - National Research and Education Networks (NRENs)
  - Campus networks
  - Governmental CSIRTs/CERTs
Lessons learned
Lessons learned
Lessons learned

• Ease-of-use is key
Lessons learned

- Ease-of-use is key
  - Many potential SSHCure users (e.g., CSIRTs) are less-skilled than we are
Lessons learned

• Ease-of-use is key

  • Many potential SSHCure users (e.g., CSIRTs) are less-skilled than we are

  • Installation scripts are important
Lessons learned

• Ease-of-use is key

  • Many potential SSHCure users (e.g., CSIRTs) are less-skilled than we are

  • Installation scripts are important

• Use of NfSen:
Lessons learned

• Ease-of-use is key

  • Many potential SSHCure users (e.g., CSIRTs) are less-skilled than we are

  • Installation scripts are important

• Use of NfSen:

  • Widely used in (European) NREN community
Lessons learned

• Ease-of-use is key
  • Many potential SSHCure users (e.g., CSIRTs) are less-skilled than we are
  • Installation scripts are important
  • Use of NfSen:
    • Widely used in (European) NREN community
    • Experience with SURFmap [1]

Lessons learned
Lessons learned

- Ingress vs. egress attacks
Lessons learned

• Ingress vs. egress attacks

• Initial focus mainly on ingress attacks
Lessons learned

• Ingress vs. egress attacks

• Initial focus mainly on ingress attacks

• CSIRTs are becoming more responsible *towards* the Internet: Keep it clean!
Lessons learned
Lessons learned

• Integration into workflow is important
Lessons learned

• Integration into workflow is important

• Yet another tool is hard to integrate into CSIRT workflow
Lessons learned

• Integration into workflow is important

• Yet another tool is hard to integrate into CSIRT workflow

• Integration with existing systems is necessary: IODEF, X-ARF, QuarantaineNet, …
Lessons learned
Lessons learned

- Advertizing is important
Lessons learned

- Advertizing is important
- People don’t spot your cool project by themselves
Lessons learned

• Advertizing is important

• People don’t spot your cool project by themselves

• Visit meetings & conferences (FloCon, TERENA TNC, RIPE, etc.)
Lessons learned

- Advertizing is important
- People don’t spot your cool project by themselves
- Visit meetings & conferences (FloCon, TERENA TNC, RIPE, etc.)
- GitHub vs. SourceForge
Lessons learned
Lessons learned

• 1:1 sampling is hardly used by non-academia
Lessons learned

• 1:1 sampling is hardly used by non-academia

• Problem for our algorithms
Lessons learned

• 1:1 sampling is hardly used by non-academia

• Problem for our algorithms

• Admins are ‘afraid’ of increasing sampling rates
Lessons learned
Lessons learned

• Input data quality is hard to predict
Lessons learned

• Input data quality is hard to predict

• Algorithms should be as resilient to various data sources as possible
Lessons learned

• Input data quality is hard to predict

• Algorithms should be as resilient to various data sources as possible

• Examples:
Lessons learned

• Input data quality is hard to predict
  • Algorithms should be as resilient to various data sources as possible
  • Examples:
    • Availability of TCP flags
Lessons learned

• Input data quality is hard to predict

• Algorithms should be as resilient to various data sources as possible

• Examples:

  • Availability of TCP flags

  • Assumptions on flow cache entry expiration
Thanks!
Questions?

https://github.com/sshcure/sshcure