Custos: Practical Tamper-Evident Auditing of Operating Systems Using Trusted Execution

Riccardo Paccagnella, Pubali Datta, Wajih Ull Hassan, Adam Bates, Christopher W. Fletcher, Andrew Miller, Dave Tian
Logs Are Useful
Logs Are Useful

• 75% of incident response specialists said logs are the most valuable artifact during an investigation.¹

¹ Carbon Black Quarterly Incident Response Threat Report April 2019
Logs Are Useful

• 75% of incident response specialists said logs are the most valuable artifact during an investigation.¹

¹ Carbon Black Quarterly Incident Response Threat Report April 2019
CAPEC-268: Audit Log Manipulation

**Attack Pattern ID:** 268  
**Abstraction:** Standard

**Presentation Filter:** Complete

**Description**

The attacker injects, manipulates, deletes, or forges malicious log entries into the log file, in an attempt to mislead an audit of the log file or cover tracks of an attack.
CAPEC-268: Audit Log Manipulation

Attack Pattern ID: 268
Abstraction: Standard

Presentation Filter: Complete

Description

The attacker injects, manipulates, deletes, or forges malicious log entries into the log file, in an attempt to mislead an audit of the log file or cover tracks of an attack.

Hackers are increasingly destroying logs to hide attacks

According to a new report. 72 percent of incident response specialists have came across hacks where attackers have destroyed logs to hide their tracks.

By Catalin Cimpanu for Zero Day | November 2, 2018 -- 16:36 GMT (09:36 PDT) | Topic: Security
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering

Logs about the compromise are crucial for forensics!
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering

Logs about the compromise are crucial for forensics!

If the attacker does not tamper with them, we can detect the attack.
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering

If the attacker tampers with them, we can’t detect the attack.

If the attacker does not tamper with them, we can detect the attack.

Logs about the compromise are crucial for forensics!
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
6. Lateral Movement
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
6. Lateral Movement

Central Server?
Attack Model

Attack pattern:

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
6. Lateral Movement
Design Overview
Design Overview

1) TAMPER-EVIDENT LOGGING
Design Overview

1) TAMPER-EVIDENT LOGGING

2) AUDITING
Logger
sk  // secret key
\[ \sigma = \text{Sig}_{sk} \left( \text{Hash}( m_1 \| \ldots \| m_h \| c) \right) \]

**Logger**

- \(sk\)  // secret key
- \(c\)  // counter
- \(H\)  // current hash

**Logging:**

- \(H.\text{Update}(m_i)\)

ENCLAVE
\[ \sigma = \text{Sig}_{sk} \left( \text{Hash}(m_1 \parallel \ldots \parallel m_h \parallel c) \right) \]
\[ \sigma = \text{Sig}_{sk}(Hash(m_1 || \ldots || m_h || c)) \]

- \( sk \) // secret key
- \( c \) // counter
- \( H \) // current hash

Logging:

- \( H.\text{Update}(m_2) \)
\[ \sigma = \text{Sig}_{sk}(\text{Hash}(m_1 || \ldots || m_h || c)) \]

Logger

\[ \begin{align*}
& m_1 \\
& m_2 \\
& \ldots \\
& m_h
\end{align*} \]

---

ENCLAVE
\[ \sigma = \text{Sig}_{sk} (\text{Hash}( m_1 \| \ldots \| m_h \| c)) \]

- \( sk \) // secret key
- \( c \) // counter
- \( H \) // current hash

**Logging:**
- \( H.\text{Update}(m_h) \)
\[
\sigma = \text{Sig}_{sk}(\text{Hash}(m_1 \| \ldots \| m_h \| c))
\]
\[ \sigma = \text{Sig}_{sk}(\text{Hash}(m_1 || \ldots || m_h || c)) \]

Logger

\[ \sigma = \text{Sig}_{sk}(\text{Hash}(m_1 || \ldots || m_h || c)) \]

Auditor

\[ \sigma = \text{Sig}_{sk}(\text{Hash}(m_1 || \ldots || m_h || c)) \]
1) CENTRALIZED AUDITING
Auditing

1) CENTRALIZED AUDITING
2) DECENTRALIZED AUDITING
Decentralized Auditing
Logger $v$

$pk_v \rightarrow \text{public key of } v$
Logger \( v \)

\[ pk_v \rightarrow \text{public key of } v \]

1. audit challenge
\[ \sigma = \text{Sig}_{sk_v}(\text{Hash}(m_1|| \ldots ||m_h||c)) \]

1. audit challenge

Logger v

pk_v \rightarrow public key of v

Auditor z

ENCLAVE
\[ \sigma = \text{Sig}_{sk_v}(\text{Hash}(m_1 | \cdots | m_h | c)) \]

Logger \( v \)

1. audit challenge

2. logs and \( \sigma \)

Auditor \( z \)

\( pk_v \rightarrow \text{public key of } v \)
\[ \sigma = \text{Sig}_{sk_v}(\text{Hash}(m_1 \parallel \ldots \parallel m_h \parallel c)) \]

1. audit challenge

2. logs and \( \sigma \)

Auditor \( z \)

Verification \((\sigma, m_1, \ldots, m_h, c)\):

\[
H = \text{Hash}(m_1 \parallel \ldots \parallel m_h \parallel c)
\]

result = \( \text{Ver}_{pk_v}(\sigma, H) \)

ENCLAVE

Logger \( v \)

\( pk_v \rightarrow \text{public key of } v \)
Security Analysis
Security Analysis

Logger $v$

- $sk$ // secret key
- $c$ // counter
- $H$ // current hash

**Logging:**

$H$.Update($m_i$)
Security Analysis

Logger $v$

- $m_1$
- $m_2$
- ...
- $m_h$

$sk$  // secret key
$c$  // counter
$H$  // current hash

Logging:

$H$.Update($m_h$)

Attack pattern:
1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
Security Analysis

Logger v

\[ sk \quad \text{// secret key} \]
\[ c \quad \text{// counter} \]
\[ H \quad \text{// current hash} \]

Logging:
\[ H.\text{Update}(m_h) \]

Attack pattern:
1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log tampering

\[ m'_1 \]
\[ m'_2 \]
\[ \ldots \]
\[ m'_k \]
Security Analysis

Logger

\[ \begin{align*}
    & sk \quad // \text{secret key} \\
    & c \quad // \text{counter} \\
    & H \quad // \text{current hash} \\

    \text{Logging:} & \\
    & H.\text{Update}(m_h)
\end{align*} \]

Auditor

\[ \text{Attack pattern:} \]

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log tampering
Security Analysis

Logger

\[ \begin{align*}
  sk & \quad \text{// secret key} \\
  c & \quad \text{// counter} \\
  H & \quad \text{// current hash}
\end{align*} \]

Logging:
\[ H.\text{Update}(m_h) \]

Commitment:
\[ H.\text{Update}(c) \]
\[ \sigma = \text{Sig}_{sk}(H) \]
\[ H.\text{Init}() \]
\[ c++ \]

Auditor

Attack pattern:
1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log tampering

\[ \begin{align*}
  \times_1 m'_1 \\
  \times_2 m'_2 \\
  \cdots \\
  \times_h m'_k
\end{align*} \]
Security Analysis

Logger

\[ \text{sk} \quad // \text{secret key} \]
\[ c \quad // \text{counter} \]
\[ H \quad // \text{current hash} \]

**Logging:**

\[ H.\text{Update}(m_h) \]

**Commitment:**

\[ H.\text{Update}(c) \]
\[ \sigma = \text{Sig}_{sk}(H) \]
\[ H.\text{Init}() \]
\[ c++ \]

Auditor

**Verification** \((\sigma, m'_1, \ldots, m'_k, c)\):

\[
H = \text{Hash}(m'_1 || \ldots || m'_k || c)
\]

\[ \text{result} = \text{Ver}_{pk_v}(\sigma, H) \]

**Attack pattern:**

1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log tampering
Security Analysis

Logger $v$

| $m'_1$ | $m'_2$ | $...$ | $m'_k$ |

- $sk$ // secret key
- $c$ // counter
- $H$ // current hash

Logging:
- $H$.Update($m_h$)

Commitment:
- $H$.Update($c$)
- $\sigma = \text{Sig}_{sk}(H)$
- $H$.Init()
- $c++$

Auditor

Verification ($\sigma, m'_1, ..., m'_k, c$):
- $H = \text{Hash}(m'_1 \| ... \| m'_k \| c)$
- result = $\text{Ver}_{pk_v}(\sigma, H)$

Attack pattern:
1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log tampering

ENCLAVE

Custos: Practical Tamper-Evident Auditing of Operating Systems Using Trusted Execution – Riccardo Paccagnella
Security Analysis

Logger v

\[ \begin{align*}
    & sk \quad // \text{secret key} \\
    & c \quad // \text{counter} \\
    & H \quad // \text{current hash} \\
\end{align*} \]

\[ \begin{align*}
    & \text{Log} \quad H.\text{Update}(m_h) \\
    & \text{Commitment:} \\
    & \quad H.\text{Update}(c) \\
    & \quad \sigma = \text{Sig}_{sk}(H) \\
    & \quad H.\text{Init}() \\
    & \quad c++
\end{align*} \]

Auditor

\[ \begin{align*}
    & \text{Verification (} \sigma, m'_1, ..., m'_k, c \text{)}: \\
    & \quad H = \text{Hash}(m'_1 \mid ... \mid m'_k \mid c) \\
    & \quad \text{result} = \text{Ver}_{pk_v}(\sigma, H)
\end{align*} \]

\[ \begin{align*}
    \text{Attack pattern:} \\
    & 1. \quad \text{Initial Access} \\
    & 2. \quad \text{Establish Foothold} \\
    & 3. \quad \text{Download Exploit} \\
    & 4. \quad \text{Privilege Escalation} \\
    & 5. \quad \text{Log tampering}
\end{align*} \]

Full security analysis on the paper!
Microbenchmarks
### Microbenchmarks

<table>
<thead>
<tr>
<th></th>
<th>Logging Latency (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custos</td>
<td></td>
</tr>
<tr>
<td>SGX-Log¹</td>
<td></td>
</tr>
<tr>
<td>BGLS²</td>
<td></td>
</tr>
</tbody>
</table>

² Hartung et al. “Practical and Robust Secure Logging from Fault-Tolerant Sequential Aggregate Signatures”, ProvSec 2017
Application Benchmarks
Application Benchmarks

Normalized Runtime

Insecure  Custos

nginx  apache2  redis  blast  blast-multicore
Realistic Case Study
Realistic Case Study

• Deploy Custos on 100 nodes.
Realistic Case Study

• Deploy Custos on 100 nodes.

• Replay attack from DARPA Transparent Computing engagement:
  – Professional red-team emulating a nation state attacker.
1. Failed Compromise Attempt (Exploit of Firefox 54.0.1)

2. Initial Access (Exploit of Firefox 54.0.1)
   - 3. Unprivileged Shell

11:42

11:46

Complete the attack
1. Failed Compromise Attempt (Exploit of Firefox 54.0.1)

11:42
2. Initial Access (Exploit of Firefox 54.0.1)
3. Unprivileged Shell

11:46
Complete the attack
4. Download Drakon

5. Privilege Escalation (through Drakon binary)

6. Log Tampering

Custos’ auditing discovered log tampering!
Conclusion
Conclusion

• Log integrity is important.

Hackers are increasingly destroying logs to hide attacks

According to a new report, 72 percent of incident response specialists have come across hacks where attackers have destroyed logs to hide their tracks.

By Catalin Ciopanu for ZDNet (November 1, 2023 — 11:46 GMT (13:46 PDT)) | Topic: Security
Conclusion

- Log integrity is important.
- Custos is a **practical** solution for log integrity.

Hackers are increasingly destroying logs to hide attacks

According to a new report, 72 percent of incident response specialists have came across hacks where attackers have destroyed logs to hide their tracks.

By Catalin Cimpanu for Zero Day | November 2, 2023 -- 11:45 GMT (10:45 PT/7:45 UT) | Topic: Security

```
Logger

// secret key
sk

// counter
c

// current hash
H

Logging:
H.Update(m)
```
Conclusion

• Log integrity is important.

• Custos is a **practical** solution for log integrity.

• Custos can discover log tampering in near real-time.
Conclusion

• Log integrity is important.

• Custos is a practical solution for log integrity.

• Custos can discover log tampering in near real-time.

• https://bitbucket.org/sts-lab/custos