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

Riccardo Paccagnella, Pubali Datta, Wajih Ul 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.\textsuperscript{1}

\textsuperscript{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

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

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
// secret key

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

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

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

Logger

\[
\begin{array}{l}
\text{ENCLAVE} \\
\frac{\text{sk} }{\text{c} } \quad \text{// secret key} \\
\frac{\text{c} }{\text{counter}} \quad \text{//} \\
\frac{H}{\text{current hash}} \\
\text{Logging:} \\
\quad H.\text{Update}(m_2)
\end{array}
\]
\[ \sigma = \text{Sig}_{sk}(\text{Hash}(m_1|| \ldots ||m_h||c)) \]

Logging:

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

**Logger**

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

**Logging:**

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

Logger

\[
\begin{align*}
\text{sk} & \quad /\!\!/ \text{secret key} \\
\text{c} & \quad /\!\!/ \text{counter} \\
\text{H} & \quad /\!\!/ \text{current hash} \\
\text{Logging:} \\
H.\text{Update}(m_h)
\end{align*}
\]

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

Logger

\[
\begin{align*}
sk & \quad \text{// secret key} \\
c & \quad \text{// counter} \\
H & \quad \text{// current hash} \\
\text{Logging:} & \\
& H.\text{Update}(m_h) \\
\text{Commitment:} & \\
& H.\text{Update}(c) \\
& \sigma = \text{Sig}_{sk}(H) \\
& H.\text{Init()} \\
& c++
\end{align*}
\]

Auditor
Auditing

1) CENTRALIZED AUDITING
Auditing

1) CENTRALIZED AUDITING
2) DECENTRALIZED AUDITING
Decentralized Auditing

Logger+Auditor

Logger+Auditor

Logger+Auditor

Logger+Auditor
Logger $v$

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

$pk_v \rightarrow public\ key\ of\ v$

1. *audit challenge*

Auditor $z$
\[ \sigma = \text{Sig}_{sk_v}(\text{Hash}(m_1|| \ldots ||m_h||c)) \]
\[ \sigma = \text{Sig}_{sk_v}(\text{Hash}(m_1 \parallel \ldots \parallel m_h \parallel c)) \]

1. **audit challenge**

2. **logs and \( \sigma \)**

**Logger v**

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

**Auditor z**

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

**Audit Challenge**

1. **Logger v**
   - \( pk_v \rightarrow \text{public key of v} \)

2. **Logs and \( \sigma \)**

**Auditor z**

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

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

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

Logger

\texttt{sk} // secret key
\texttt{c} // counter
\texttt{H} // current hash

\textbf{Logging:}

\texttt{H.Update(m_i)}
Security Analysis

Logger v

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

Logging:

$$H.\text{Update}(m_h)$$

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

Logger $v$

$\text{ENCLAVE}$

$sk$  // 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
5. Log tampering
Security Analysis

Logger

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

Logging:

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

Auditor

Attack pattern:

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

Logger v

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

Logging:
- $H.\text{Update}(m_h)$

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

Auditor

ENCLAVE

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

Logger

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

Logging:

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

Commitment:

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

Auditor

\[ \text{Verification } (\sigma, m'_1, \ldots, m'_k, c): \]
\[ H = \text{Hash}(m'_1 \parallel \ldots \parallel m'_k \parallel c) \]
\[ \text{result } = \text{Ver}_{pk_v}(\sigma, H) \]

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

Logger

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

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

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

Auditor

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

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

Logger

\[
\begin{align*}
\text{sk} & \quad \text{// secret key} \\
\text{c} & \quad \text{// counter} \\
H & \quad \text{// current hash} \\
\text{Log}\!\text{g}: & \\
& H.\text{Update}(m_h) \\
\text{Commitment:} & \\
& H.\text{Update}(c) \\
& \sigma = \text{Sig}_{sk}(H) \\
& H.\text{Init}() \\
& c++
\end{align*}
\]

Auditor

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

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

Full security analysis on the paper!
Microbenchmarks
Microbenchmarks

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

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

![Normalized Runtime Chart]

- nginx
- apache2
- redis
- blast
- blast-multicore

Normalized Runtime

- Insecure
- Custos
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

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

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

11:42

Complete the attack

11:46
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 came across hacks where attackers have destroyed logs to hide their tracks.

Conclusion

• Log integrity is important.

• Custos is a **practical** solution for log integrity.
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](https://bitbucket.org/sts-lab/custos)