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

ILLINOIS

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

Logs about the compromise are crucial for forensics!

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

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

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:

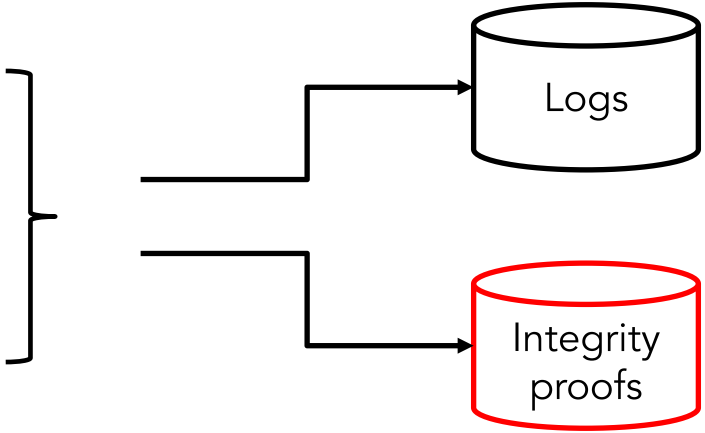
1. Initial Access
2. Establish Foothold
3. Download Exploit
4. Privilege Escalation
5. Log Tampering
6. Lateral Movement

Central Server?

Attack Model

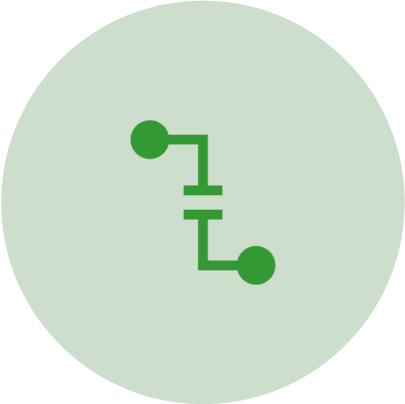
Attack pattern:

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



Design Overview

Design Overview

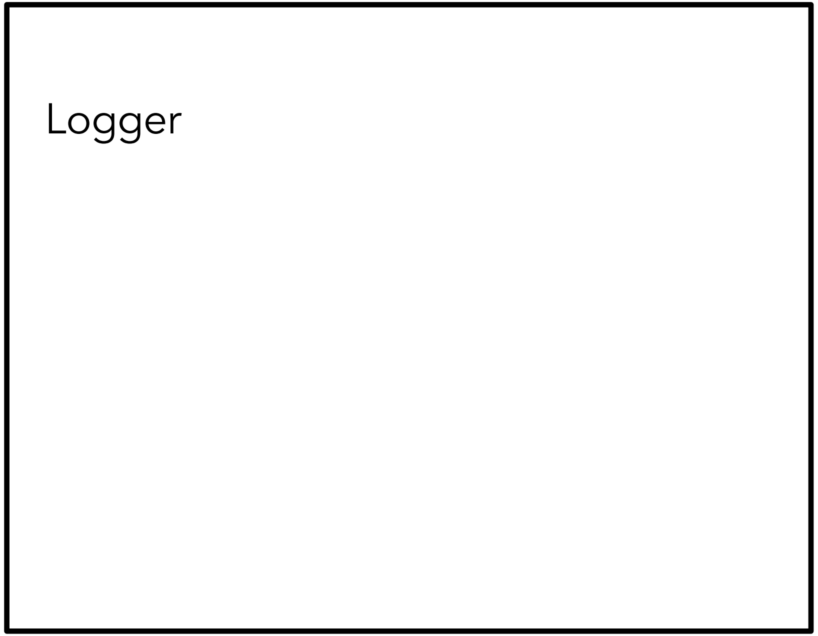


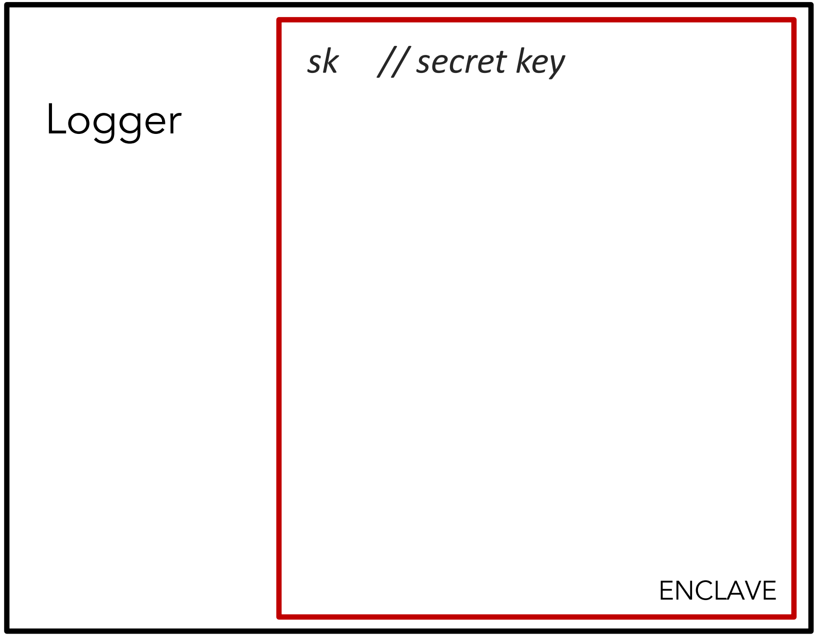
1) TAMPER-EVIDENT LOGGING

Design Overview

1) TAMPER-EVIDENT LOGGING

2) AUDITING


Logger


Logger
sk // secret key
ENCLAVE

$$\boldsymbol{\sigma} = \boldsymbol{Sig_{sk}}\,(\underline{\boldsymbol{Hash}(\,\boldsymbol{m_1}||\ \ldots\ ||\,\boldsymbol{m_h}||\boldsymbol{c}})\,)$$

Logger

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

Logging:

H.Update(m_i)

ENCLAVE

$$\sigma = Sig_{sk}\,(\underline{Hash(\,m_1 || \,\dots\, || \,m_h || c)})$$

Logger

m_1 →

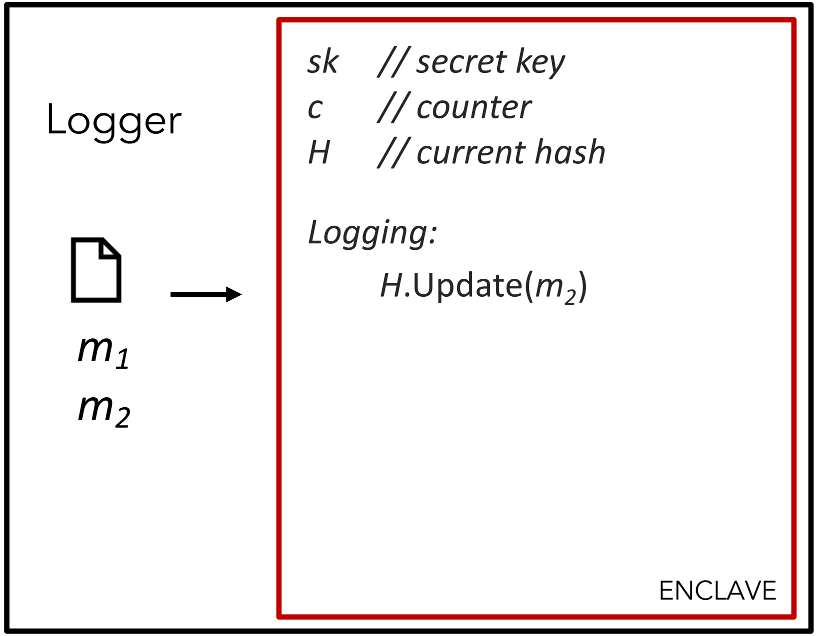
sk // *secret key*
c // *counter*
H // *current hash*

Logging:

H.Update(m_1)

ENCLAVE

$$\sigma = Sig_{sk}\left(\underline{Hash(m_1 || \dots || m_h || c)}\right)$$



$$\sigma = Sig_{sk}\,(\underline{Hash(\,m_1 || \,\ldots\, || \, m_h || c)})$$

Logger

m_1
m_2
…
m_h

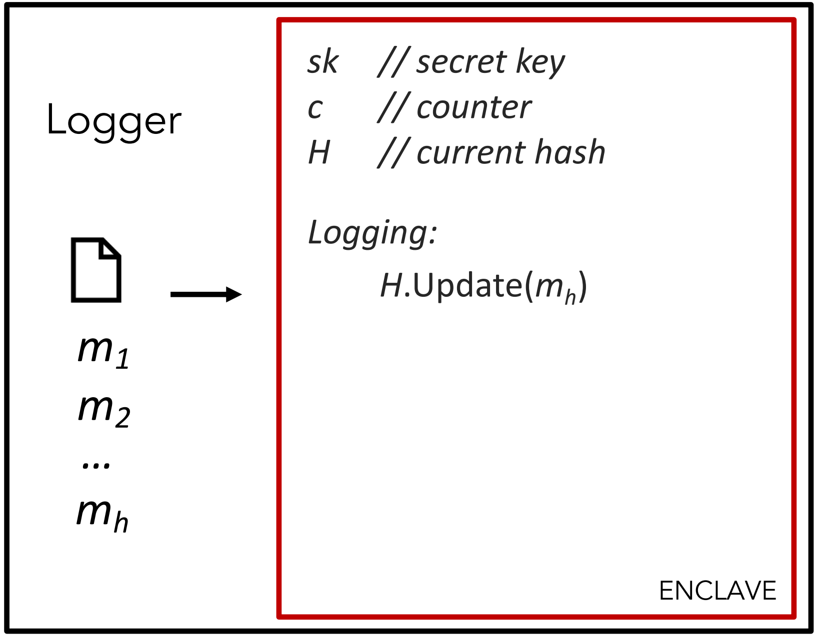
sk *// secret key*
c *// counter*
H *// current hash*

Logging:

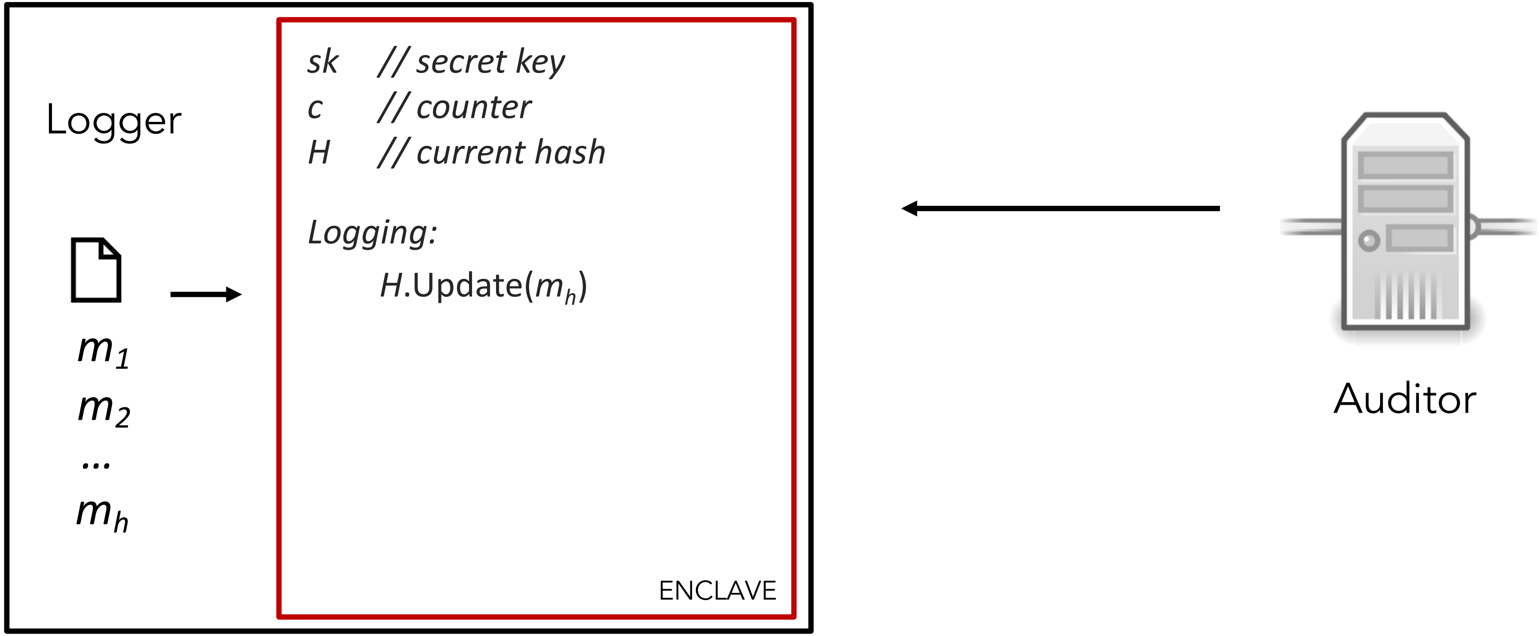
H.Update(m_h)

ENCLAVE

$$\sigma = \underline{Sig_{sk}} (Hash(m_1 || \ldots || m_h || c))$$



$$\boldsymbol{\sigma} = \boldsymbol{Sig_{sk}}\,(\boldsymbol{Hash}(\,\boldsymbol{m_1}||\ \dots\ ||\,\boldsymbol{m_h}||\boldsymbol{c}))$$



$$\sigma = \underline{Sig_{sk}} \left(Hash(m_1 || \ldots || m_h || c)\right)$$

Logger

m_1
m_2
…
m_h

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

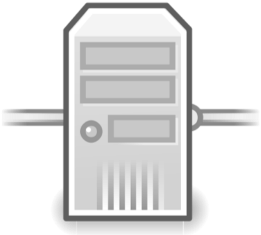
Logging:

H.Update(m_h)

Commitment:

H.Update(c)
σ = Sig$_{sk}$(H)
H.Init()
c++

ENCLAVE



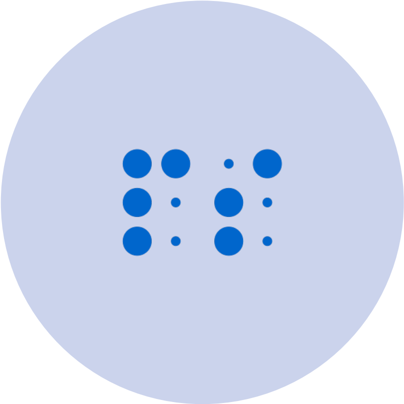
Auditor

Auditing

1) CENTRALIZED AUDITING

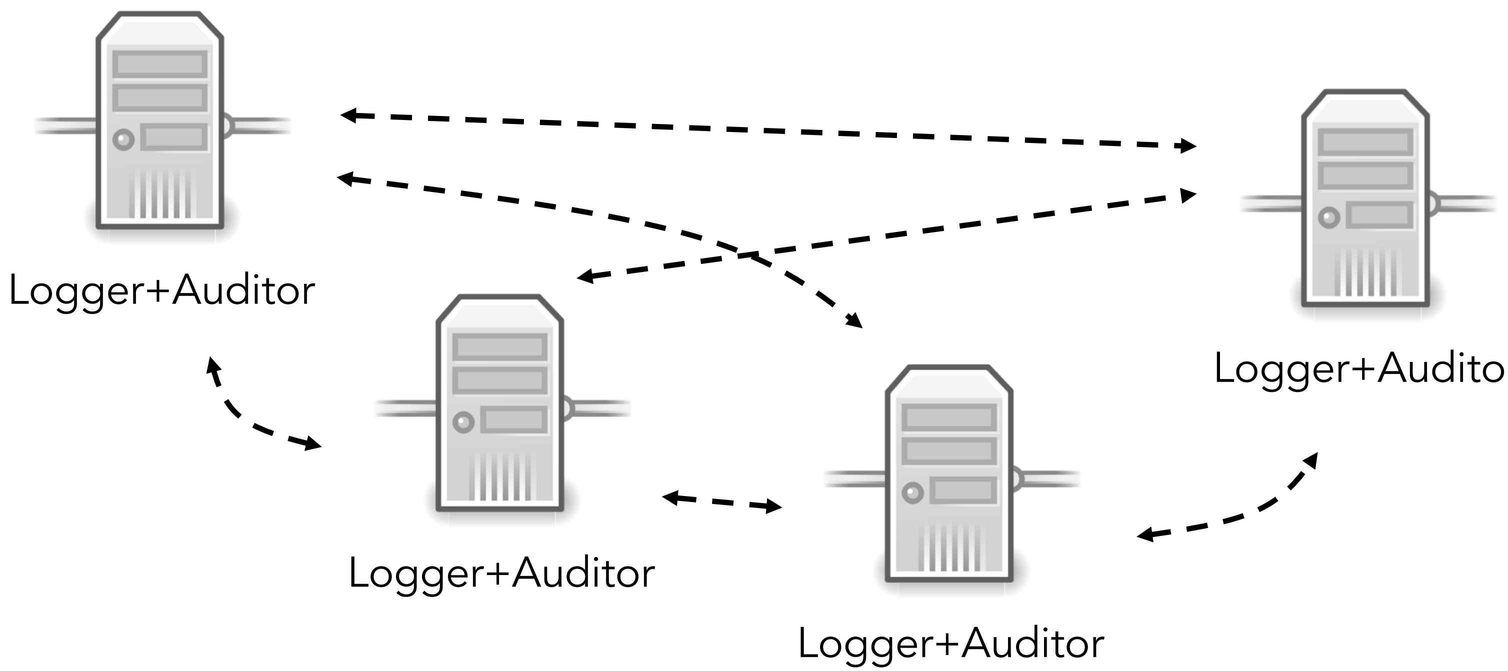
Auditing

1) CENTRALIZED AUDITING



2) DECENTRALIZED AUDITING

Decentralized Auditing

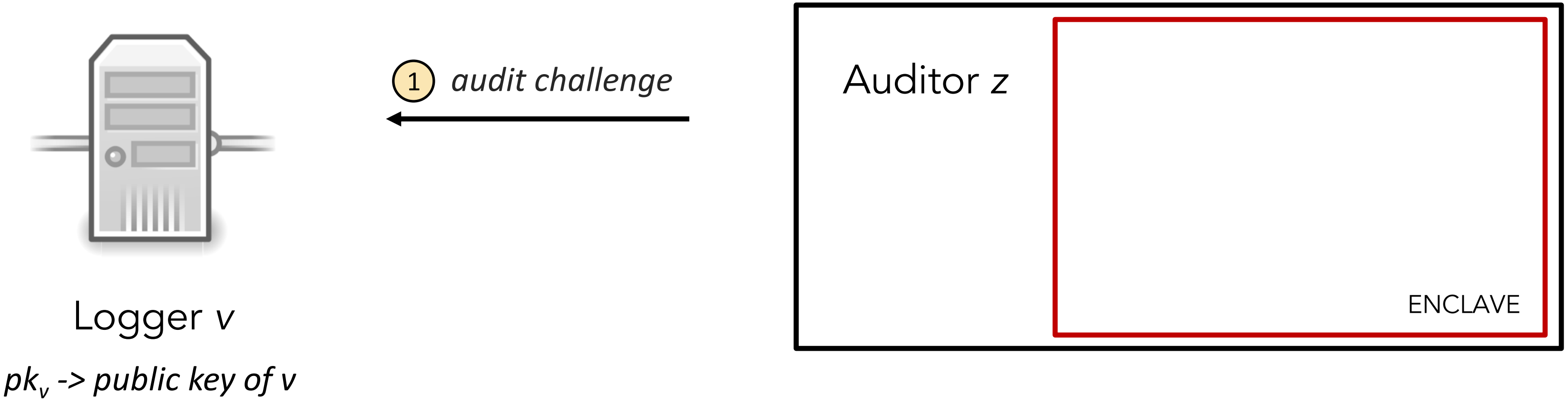


Logger *v*

pk_v -> public key of v

Auditor *z*

ENCLAVE


1 audit challenge
Auditor z
ENCLAVE
Logger v
pk_v -> public key of v

$$\sigma = Sig_{sk_v}(Hash(m_1 || \ldots || m_h || c))$$

Logger *v*

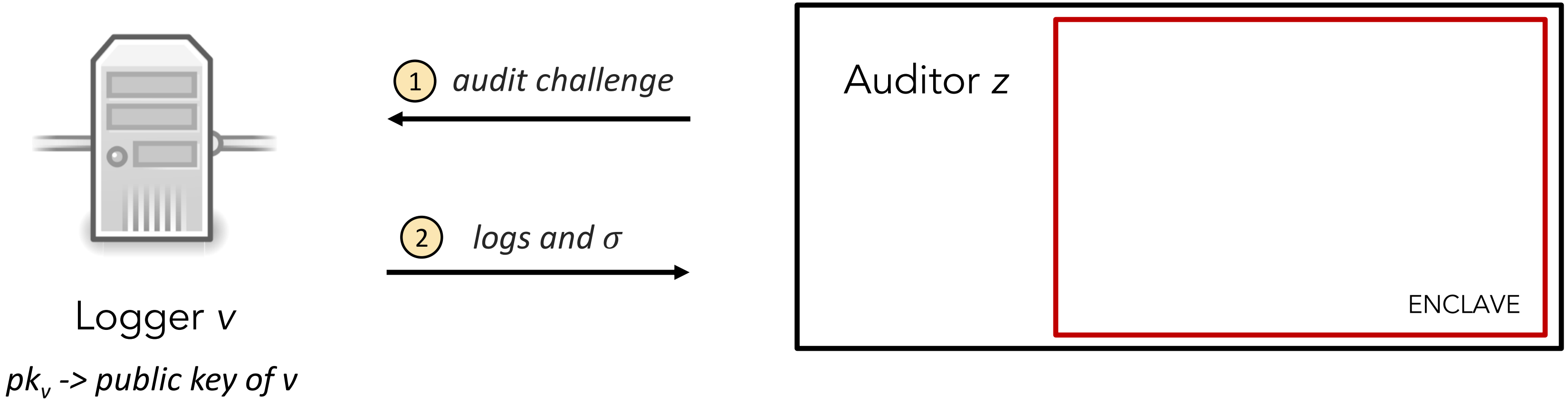
pk_v -> public key of v

1 *audit challenge*

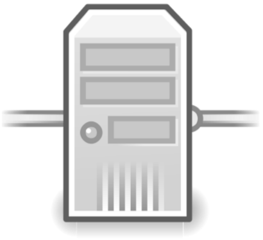
Auditor *z*

ENCLAVE

$$\sigma = Sig_{sk_v}(Hash(m_1 || \ldots || m_h || c))$$



$$\sigma = Sig_{sk_v}(Hash(m_1 || \ldots || m_h || c))$$



Logger v

pk_v -> public key of v

① *audit challenge* ←

② *logs and σ* →

Auditor z

Verification (σ, m_1, …, m_h, c):

$H = Hash(m_1 || \ldots || m_h || c)$

result = $Ver_{pk\_v}(\sigma, H)$

ENCLAVE

Security Analysis

Security Analysis

Logger v

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

Logging:

H.Update(m_i)

ENCLAVE

Security Analysis

Logger *v*

m_1

m_2

…

m_h

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

Logging:

H.Update(m_h)

ENCLAVE

Attack pattern:

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

Security Analysis

Logger v

$\cancel{m}_1$ m'_1
$\cancel{m}_2$ m'_2
… …
$\cancel{m}_h$ m'_k

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

Logging:

H.Update(m_h)

ENCLAVE

Attack pattern:

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

Security Analysis

Logger v

$\cancel{m_1}$ m'_1
$\cancel{m_2}$ m'_2
… …
$\cancel{m_h}$ m'_k

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

Logging:

H.Update(m_h)

ENCLAVE

Auditor

ENCLAVE

Attack pattern:

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

Security Analysis

Logger v

$\cancel{m}_1$ m'_1
$\cancel{m}_2$ m'_2
… …
$\cancel{m}_h$ m'_k

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

Logging:

H.Update(m_h)

Commitment:

H.Update(c)
σ = Sig$_{sk}$(H)
H.Init()
c++

ENCLAVE

Auditor

ENCLAVE

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

Security Analysis

Logger v

$\cancel{m_1}$ m'_1
$\cancel{m_2}$ m'_2
… …
$\cancel{m_h}$ m'_k

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

Logging:

H.Update(m_h)

Commitment:

H.Update(c)
$\sigma = \mathrm{Sig}_{sk}(H)$
H.Init()
c++

ENCLAVE

Auditor

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

$H = Hash(m'_1 \,||\, \ldots \,||\, m'_k \,||\, c)$
result = $\mathrm{Ver}_{pk\_v}(\sigma, H)$

ENCLAVE

Attack pattern:

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

Security Analysis

Logger v

$\cancel{m_1}$ m'_1
$\cancel{m_2}$ m'_2
… …
$\cancel{m_h}$ m'_k

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

Logging:

H.Update(m_h)

Commitment:

H.Update(c)
σ = Sig$_{sk}$(H)
H.Init()
c++

ENCLAVE

Auditor

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

$H = Hash(m'_1 \,||\, \ldots \,||\, m'_k \,||\, c)$
result = Ver$_{pk_v}$(σ, H)

ENCLAVE

Attack pattern:

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

Security Analysis

Logger v

$\cancel{m_1}$ m'_1
$\cancel{m_2}$ m'_2
… …
$\cancel{m_h}$ m'_k

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

Logg

H.Update(m_h)

Commitment:

H.Update(c)
σ = Sig$_{sk}$(H)
H.Init()
c++

ENCLAVE

Full security analysis on the paper!

Auditor

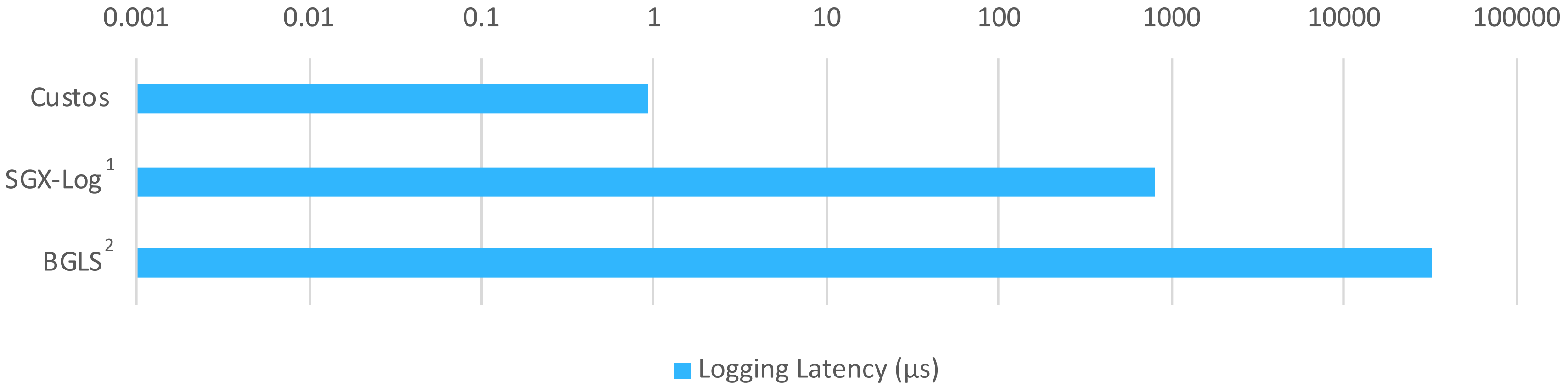
Verification $(\sigma, m'_1, \dots, m'_k, c)$:
$H = Hash(m'_1 \| \dots \| m'_k \| c)$
result = Ver$_{pk_v}(\sigma, H)$

ENCLAVE

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

Microbenchmarks

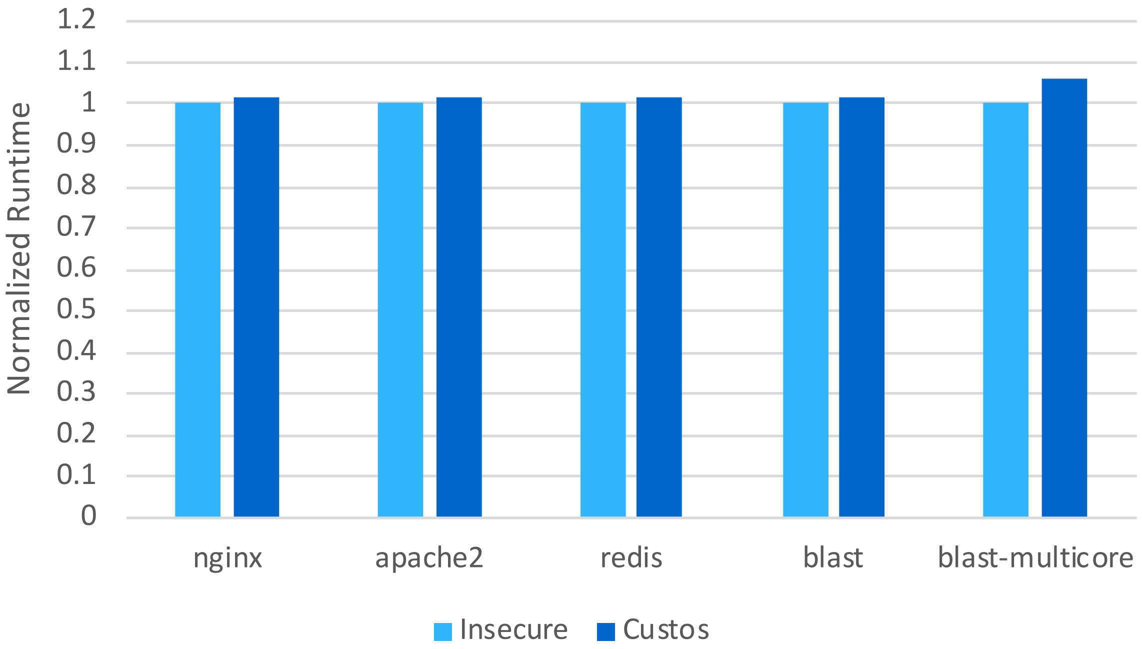
Microbenchmarks



[1] Karande et al. ”SGX-log: Securing System Logs With SGX." ASIACCS 2017.
[2] Hartung et al. “Practical and Robust Secure Logging from Fault-Tolerant Sequential Aggregate Signatures”, ProvSec 2017

Application Benchmarks

Application Benchmarks



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.

10:52

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

10:52

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

10:52

11:42

11:46

11:46:17

4. Download Drakon

11:46:44

5. Privilege Escalation (through Drakon binary)
6. Log Tampering

11:46:47

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.

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

Conclusion

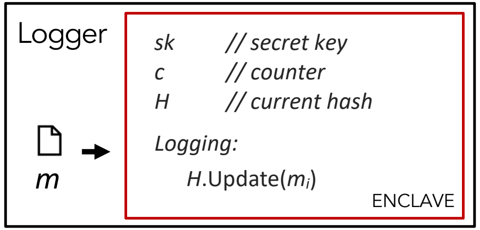
- Log integrity is important.

- 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, 2018 -- 16:36 GMT (09:36 PDT) | Topic: Security



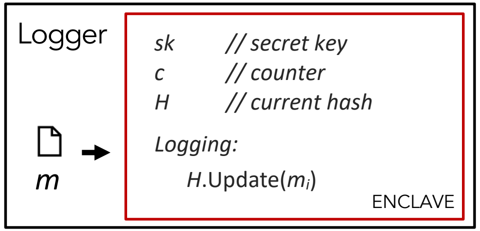
Conclusion

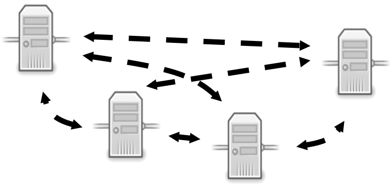
- Log integrity is important.
- Custos is a **practical** solution for log integrity.
- Custos can discover log tampering in near real-time.

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.

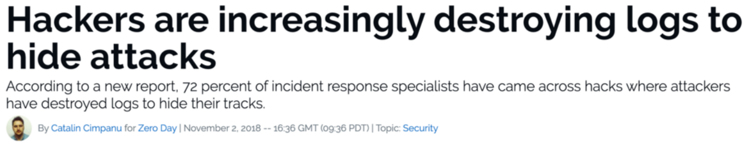
By Catalin Cimpanu for Zero Day | November 2, 2018 -- 16:36 GMT (09:36 PDT) | Topic: Security





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



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