Audit, Logs and Secure Logs

What really happened?
Agenda

1. Auditing
2. Logs
3. Secure Logs
Audit (1/2)

• "Audit, -noun, an official examination and verification of accounts and records, especially of financial accounts." - [http://dictionary.reference.com/browse/audit](http://dictionary.reference.com/browse/audit)

• "Audit, -verb, to make an audit of; examine (accounts, records, etc.) for purposes of verification: The accountants audited the company's books at the end of the fiscal year." - [http://dictionary.reference.com/browse/audit](http://dictionary.reference.com/browse/audit)

• When auditing you try to discover what is really going on, to for example:
  – Find security issues (outdated software, poor security, etc)
  – Evaluate systems, processes, risks and controls
  – Enable people to make informed decisions
Audit (2/2)

• How do you discover what is really going on in a network?
  1. Identify the topology of the network
  2. Identify all hosts on the network
  3. Determine services running on all hosts
  4. The configuration of all services and applications on the host
  5. Look at logs

• An audit log or audit trail
  – ”Chronological record of system activities to enable the reconstruction and examination of the sequence of events and/or changes in an event“ - National Information Assurance Glossary CNSSI 4009
  – “A record showing who has accessed an IT system and what operations the user has performed during a given period“ - NIST Special Publication 800-47 Security Guide for Interconnecting Information Technology Systems
Common Criteria (CC)

• Harmonized criteria for the international community for evaluation and recognition
  – ISO IS 15408

• CC scope covers *Systems* and *Products*

• CC is divided in 3 parts
  – Part 1: Introduction and general model
  – Part 2: Security functional requirements
    • basis to describe the desired security behavior of a target of evaluation, that is, security services
  – Part 3: Security assurance requirements
    • evaluation assurance levels (EAL) and assurance classes
Evaluation Assurance Level (EAL)

- EAL1 - functionally tested
- EAL2 - structurally tested
- EAL3 - methodically tested and checked
- EAL4 - methodically designed, tested and reviewed
- EAL5 - semi-formally designed and tested
- EAL6 - semi-formally verified design and tested
- EAL7 - formally verified design and tested

Evaluation is not a primary goal during design.

TOE is designed with evaluation in mind.
Common Criteria Requirements on Security Audits (1/2)

• "Security auditing involves recognising, recording, storing, and analysing information related to security relevant activities (i.e. activities controlled by the TSF). The resulting audit records can be examined to determine which security relevant activities took place and whom (which user) is responsible for them." – CC Part 2 v3.1R3
  – TSF stands for Target of Evaluation Security Functions

• Requirements on:
  – Security audit automatic response
    • “defines the response to be taken in case of detected events indicative of a potential security violation“ – CC Part 2 v3.1R3

  – Security audit data generation
    • The level of auditing, types of events and minimum information that should be provided with various audit record types
Common Criteria Requirements on Security Audits (2/2)

- Security audit analysis
  - "...defines requirements for automated means that analyse system activity and audit data looking for possible or real security violations." – CC Part 2 v3.1R3

- Security audit review
  - "...defines the requirements for audit tools that should be available to authorised users to assist in the review of audit data." – CC Part 2 v3.1R3

- Security audit event selection
  - "...defines requirements to select the set of events to be audited during TOE operation from the set of all auditable events." – CC Part 2 v3.1R3
    - TOE stands for Target of Evaluation

- Security audit event storage
  - "...defines the requirements for the TSF to be able to create and maintain a secure audit trail." – CC Part 2 v3.1R3 (Bold formatting added)
Agenda

1. Auditing
2. Logs
3. Secure Logs
Logs (1/2)

• A log is a record of sequential data:
  – Audit logs
  – Transaction logs
  – Connections logs
  – ...

• Generated by:
  – Operating system
  – Databases
  – Firewalls
  – Antivirus
  – Routers
  – ...

Source: Using Logs for Breach Investigations and Incident Response, by Dr Anton Chuvakin
Looking at logs helps you figure out:

- **Who** did something
  - But is who a person or a system? Is an IP-address a person?

- **Where** something happened or originated from
  - Is it spoofed? An IP-address belongs to an ISP in China, so what?

- **When** something happened
  - What timezone and according to what clock? Any delay in logging?

- **How** something happened
  - But in what detail and how sure can you be?

- **What** happened
  - Or only what got recorded?

Source: Using Logs for Breach Investigations and Incident Response, by Dr Anton Chuvakin
Log Trustworthiness Hierarchy (1/2)

According to Dr Anton Chuvakin, from worst to best:

1. Compromised system logs (mostly pure distilled crap :-), but might contain bits that attacker missed/ignored

2. Desktop / laptop OS and application logs (possibly changed by users, legitimate systems owners, etc)

3. All logs from others systems where 'root'/Admin access is not controlled (e.g. test servers, etc)

4. Unix application logs (file-based)

5. Local Windows application logs

6. Local Unix OS syslogs
Log Trustworthiness Hierarchy (2/2)

7. Unix kernel audit logs, process accounting records

8. Local Windows server OS (a little harder to change)

9. Database logs

10. Other security appliance logs (located on security appliances)

11. Various systems logs centralized to a syslog server

13. Network device and firewall logs (centralized to syslog server)

14. Logs centralized to a log management system via a real-time feed (obviously, transport encryption adds even more trust)
Trustworthiness in Logs

• Trust in the contents of logs is **key**

• But as we have seen, there are different degrees of trustworthiness for different logs
  – And what is actually stored in the logs can be questioned

• So, what does this mean in practice?

• To gain confidence in events
  – Independently corroborate from multiple sources

• **You increase reliability by increasing the number of sources**
Windows Security Event Log

• Designed to be as trustworthy as possible
  – Access control on the API to log
  – Access control on the log itself
    • Only administrators can modify

• Uses trusted sources, like system state, whenever possible for data to be logged instead of what is supplied to the API
  – Makes entries harder to use
    • Would like "Eric used Excel to change the executive compensation spreadsheet"
    • But looks like “process ID 1289 running in context S-1-5-21-1235-6780-501 performed a WRITE_DATA on \dosdevices\c\shared files\exec_comp.xls”

Syslog

• A standard (RFC 3164) for logging application messages
  – Applications generate messages
  – The syslog deamon directs messages
    • To the console
    • To logfiles in /var/log/
    • To remote syslog deamons

• Simple message structure:
  – Facility, 23 in total
    • kernel, user-level, mail, daemons, auth, internal, audit..
  – Priority level, 8 in total
    • Emergency, Alert, Critical, Error, Warning, Notice, Informational or Debug
  – The message to log
Agenda

1. Auditing
2. Logs
3. Secure Logs
What is a Secure Log?

- **A Log:**
  - A record of sequential data

- **A Secure Log:**
  - Protects the confidentiality and integrity of entries
    - Confidentiality provided by encrypting the data part of entries
    - Integrity provided by using hashes and MACs
  - *Prior to* an attacker compromises the logging system
    - Forward secrecy
    - Forward integrity
How it Works: a Secret

- Each entry has an authentication key

- For an entry with index $i$
  - $A_i = \text{hash}(A_{i-1})$
  - $A_0 = \text{the initial secret}$
How it Works: Integrity

- Each entry has a field that forms a chain
  \[ Y_i = \text{MAC}_{A_i}(\text{all other fields, } Y_{i-1}) \]

- Provides \textit{cumulative verification}
  - Verifying \( Y_i \) also verifies the integrity of all prior entries

- Allows \textit{detection} of any changes made to the entries
How it Works: Confidentiality

• The actual data in each entry is encrypted, two approaches:
  
  – Symmetric encryption with key derived from authentication key
  
  – Asymmetric encryption, using the public key
    • $\text{Data}_i = \text{ENC}_{\text{PK}_{ds}}(\text{data}_i)$
    • How is the public key given to the logging system?

• This *prevents* an attacker from reading the contents

• A form of *access control*
Usage

• Logging on insecure or untrusted systems

• The *prior-to* property
  – For the entries committed to the log prior-to compromise the attacker
    • Cannot read their contents
    • Make *undetectable* modifications
  – Reduces the attacker’s options to:
    • Leave the entries alone
    • Delete them all
Example: The Schneier-Kelsey Secure Log

- **Setting:**
  - An untrusted machine $U$
    - Where the logging takes place
  - A trusted machine $T$
    - Is periodically in contact with $U$
    - Wants to read the log entries on $U$ and verify their integrity
  - Attacker model: an attacker compromises $U$ at time $t$
    - Should not be able to alter entries committed to the log without detection
    - Should not be able to read their contents

Based upon “Secure Audit Logs to Support Computer Forensics”, by Bruce Schneier and John Kelsey
Overview of the Schneier-Kelsey Secure Log (1/2)

- Logging *data*, for entry with index *i*:
  - Fields:
    - $W_i = A$ permission mask for access control
    - $E_i = \text{ENC}_{K_i}(\text{data})$
    - $Y_i = \text{hash}(Y_{i-1}, E_i, W_i)$
    - $Z_i = \text{MAC}_{A_i}(Y_i)$
  - State:
    - $A_i = \text{hash}("\text{Increment Hash"}, A_{i-1})$
  - Key derivation:
    - $K_i = \text{hash}("\text{Encryption Key"}, W_i, A_i)$

Based upon "Secure Audit Logs to Support Computer Forensics", by Bruce Schneier and John Kelsey
Overview of the Schneier-Kelsey Secure Log (2/2)

Based upon the overview figure in "Secure Audit Logs to Support Computer Forensics" by Bruce Schneier and John Kelsey

- Encrypted Data: The actual data to log stored in an encrypted form using a derived key from the authentication key.
- Permission Mask: Used by the surrounding system for permissions and when deriving the key for the data field.
- Hash Chain: Protects the integrity of the entry and all previous entries in the log.
- MAC: Authenticates the hash chain, and thus the log entry and all previous log entries, using the authentication key as a secret.
- Authentication Key: The authentication key for the entry, derived from the previous authentication key.
Validation of the Schneier-Kelsey Secure Log

- Anyone with access to the log can validate the hash-chain $Y$
  - Schneier-Kelsey supports a semi-trusted person $V$ to validate and read parts of the log
  - Knowledge of the authentication key $A_i$ for entry with index $i$ allows verification of the MAC $Z_i$
    1. Generate and compare the hash-chain $Y$
    2. Check the MAC $Z_i$ of the last entry

- Any problem with this?
Attacks on the Schneier-Kelsey Secure Log

• A truncation attack
  – An attacker deletes a continuous subset of tail-end entries in the log on $U$, where the entries have not been validated by $T$
  – The validation process does not involve checking the state of $U$
    • An attacker cannot generate authentication keys used for entries committed to the log prior to compromising $U$

• A delayed detection attack
  – The trusted party $T$ is required to validate the integrity of the log
    • Anyone can generate a new hash-chain $Y$
      – $Y_i = \text{hash}(Y_{i-1}, E_i, W_i)$
    • The authentication key used to key the MAC $Z$ is needed
      – $Z_i = \text{MAC}_{A_i}(Y_i)$
Desirable Properties of Secure Logs Missing From the Schneier-Kelsey Secure Log

• Public Verifiability
  – Anyone can verify the integrity of the log

• Offline Trusted Third Party
  – The trusted third party does not have to be involved for verification

• Immediate Verification
  – Verification without delay

• Resilient to Delayed Detection Attacks
  – Countered by the immediate verification property

• Resilient to Truncation Attacks
  – Inspecting state in the Schneier-Kelsey Secure Log enables detection
Trusted Timestamps

• A way of proving the existence of data prior to a timestamp

• A TimeStamping Authority (TSA)

• Tree types of schemes:
  – Simple
    • A TSA issues a signature on the hash of a piece of data together with a timestamp
    • Shown on the next two slides
  – Linked
    • The TSA issues a signature on data that includes data included in other signatures
  – Distributed
    • Multiple TSAs cooperate to generate a signature

Based on "The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies", by Masashi Une
Trusted timestamping

Within a company

Data

Calculate hash

1011...10101

Send hash to TSA

Timestamping Authority (TSA)

1011...10101 +

Timestamp

Calculate hash

0010...01011

Apply private key of the TSA

This is a digital signature of the hash concatenated to the timestamp

0010...01011 +

Timestamp

Signed timestamp and hash are returned to requester

Store together

Checking the trusted timestamp

Calculation steps:
1. Calculate hash of the data: 1011...10101
2. Calculate hash of the timestamp: 0010...01011
3. Apply the public key of the Timestamping Authority (TSA)

Comparison of hash codes:
- Calculated hash: 0010...01011
- Trusted hash: 0010...01011

If the calculated hash code equals the result of the decrypted signature, neither the document or the timestamp was changed and the timestamp was issued by the TTP. If not, either of the previous statements is not true.
Mending Parts of the Schneier-Kelsey Secure Log

• What we already know:
  − Resilient to truncation attacks
    • Inspecting state enables detection of the attack

• Mending public verifiability and offline trusted third party
  − TSA timestamp on the hash-chain $Y$
  − Done by $T$

• Problems:
  − No immediate verification, so vulnerable to delayed detection attacks
  − $T$ still needed for the entries added to the log in $U$ after the latest contact with $T$
    • The authentication key used to key the MAC $Z$ is needed
      − $Z_i = \text{MAC}_{A_i}(Y_i)$
Summary

• When auditing you try to discover what is *really* going on
  – And looking at logs is part of doing it

• There are different degrees of trustworthiness for different logs
  – Trust is key
  – You increase reliability by increasing the number of sources

• Secure logs deal with the integrity and confidentiality of logs
  – “Prior-to” property
  – Logging on insecure or untrusted systems