Auditability and Accountability

SecAppDev 2007

Quiz

Here’s the scenario

- Your software has just suffered a major security breach
- The CEO has called in law enforcement and a Computer Security Incident Response Team (CSIRT) to help clean up the mess

Now what?

What are the CSIRT's top priorities?

How will your software help the CSIRT do its job?

Who will write the company's *next* software?

Here’s my answer to #1

- Highest priority is to determine the business impact
- Second highest is situational awareness throughout the incident
- Third is to recommend a course of action to take, and then to coordinate execution of that plan

In reality

In my 20+ years of incident response experience:
- The CSIRT is called up after the fact
- Often, the attackers have come and gone
- CSIRT has to assemble the puzzle from available data
  - There's never enough–or the right–available data

CSIRT needs to…

- Determine the who, what, when, where, and how (WWWWH)
  - Using existing records of the events
  - Disk and network forensics often not terribly useful
    - After the fact may be too late
  - Auditability vs. accountability

Event logging

In a production data processing environment, there can be many sources of log data

- With luck, they're sent to a central log concentrator
- Consider the per-source perspective
  - What did the (router, firewall, web server, Java container, database) see?
  - Now, what did they *report*?
  - How do they speak to WWWWH?

Synthesizing the data

From our event data, what can we tell?

- How is it all used to figure out what happened?
- How do our encrypted network protocols affect what we see?
- Examples

Examples –1

Does this mean anything (useful) to you?

```
Jan 24 12:00:21 example-host ipop3d[40900]: pop3s SSL service init from 10.1.2.3
Jan 24 12:00:22 example-host ipop3d[40901]: pop3s SSL service init from 10.1.2.3
Jan 24 12:00:43 example-host imapd[40914]: imap service init from 127.0.0.1
Jan 24 12:00:47 example-host imapd[40929]: imap service init from 127.0.0.1
Jan 24 12:00:47 example-host imapd[40936]: imap service init from 127.0.0.1
Jan 24 12:00:48 example-host imapd[40949]: imap service init from 127.0.0.1
Jan 24 12:00:49 example-host imapd[40952]: imap service init from 127.0.0.1
Jan 24 12:00:49 example-host imapd[40953]: imap service init from 127.0.0.1
Jan 24 12:01:21 example-host ipop3d[40967]: pop3s SSL service init from 10.1.2.3
Jan 24 12:01:22 example-host ipop3d[40968]: pop3s SSL service init from 10.1.2.3
Jan 24 12:02:21 example-host ipop3d[40993]: pop3s SSL service init from 10.1.2.3
```

Examples –2

How about this one?

```
Jan 20 10:48:25 example-host sshd[64110]: Accepted publickey for krvw from
    10.1.2.3 port 33494 ssh2
Jan 20 11:10:39 example-host sshd[64761]: Invalid user patrick from
    216.144.225.211
Jan 20 11:10:40 example-host sshd[64763]: Invalid user patrick from
    216.144.225.211
Jan 20 11:10:49 example-host sshd[64775]: Invalid user rolo from 216.144.225.211
Jan 20 11:10:50 example-host sshd[64777]: Invalid user iceuser from
    216.144.225.211
Jan 20 11:10:51 example-host sshd[64779]: Invalid user horde from 216.144.225.211
Jan 20 11:10:54 example-host sshd[64785]: Invalid user wwwrun from
    216.144.225.211
Jan 20 11:10:55 example-host sshd[64787]: Invalid user matt from 216.144.225.211
Jan 20 11:10:59 example-host sshd[64795]: Invalid user test from 216.144.225.211
```

Examples –3 (last one…)

And what does this one tell you?

```
64.4.8.137 - - [24/Jan/2007:06:15:09 -0500] "GET /robots.txt HTTP/1.0" 200 0 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:15:09 -0500] "GET /rss.xml HTTP/1.0" 200 8613 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:38:41 -0500] "GET /robots.txt HTTP/1.0" 200 0 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:38:41 -0500] "GET /about.php HTTP/1.0" 200 9770 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:42:21 -0500] "GET /index.php HTTP/1.0" 200 5080 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:42:38 -0500] "GET /courses.php HTTP/1.0" 200 7509 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:06:50:07 -0500] "GET /whats_new.php HTTP/1.0" 200 12404 "-"
    "msnbot/1.0 (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:10:09:30 -0500] "GET /contact.php HTTP/1.0" 200 3526 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:10:09:34 -0500] "GET /consulting.php HTTP/1.0" 200 4936 "-"
    "msnbot/1.0 (+http://search.msn.com/msnbot.htm)"
64.4.8.137 - - [24/Jan/2007:10:09:34 -0500] "GET /sclist.php HTTP/1.0" 200 3139 "-" "msnbot/1.0
    (+http://search.msn.com/msnbot.htm)"
```

Wow, those were ugly

So, what is missing from our logging?

- Meaningful data about the software
- Pretty much all of WWWWH

Wouldn't it be neat if they looked more like…

What if…

Isn't this a little more meaningful?

```
Jan 24 12:00:20 example-host apache[32767]: 10.1.2.3 "GET /webmail.php HTTP/1.0"
Jan 24 12:00:47 example-host imapd[40929]: imap service init from localhost:apache
Jan 24 12:00:47 example-host imapd[40929]: user wvrk authenticated
Jan 24 12:00:48 example-host imapd[40929]: user wvrk downloaded 3 new messages
Jan 24 12:00:49 example-host imapd[40929]: user wvrk messageids: 1234, 2345, 3456
Jan 24 12:00:49 example-host imapd[40929]: user wvrk
Jan 24 12:01:21 example-host apache[32767]: webmail.php: user wvrk session terminated
```

What can dev to to improve the status quo?

Dev’s role in infosec

How do you contribute to infosec today?

- Protect
- Detect
- React

Now, let’s consider some alternatives

Protect –1

Designing the software

- Think about the application-to-OS interface
- Compartmentalization of function
  - Make full use of OS security features
    - E.g., separate identities/credentials for different functions
  - Virtual machines for some functions
- Case study: Postfix MTA

Protect –2

Deploying the software

- File and directory access controls
- Apply the principle of least privilege

"Every program and every user of the system should operate using the least set of privileges necessary to complete the job."
– Saltzer and Shroeder, 1975

Protect –3

Security states

- Many infosec organizations use a tiered priority scale for security incidents
  - E.g., "Infocon", priority 1-5, green through red scale
- Consider how your software might be aware and interact
  - E.g., heightened logging, defensive posture

Protect –4

App-level intrusion detection

- Consider viability of a "mini IDS" in your software
  - E.g., statistical profiling of normative behavior
  - Possibly in conjunction with extant IDS
- Case study: hotel credit card processing "velocity checker"

Detect –1

- Event logging and monitoring
  - Client and server considerations
    - Reporting agent and log concentrator
  - Highest risk components of software
    - Your architectural risk analysis should guide you here
  - Consider how your application works with your Ops team’s monitoring
    - What would you need to know if you had to analyze an intrusion?
    - Do they use any standardized data fields, nomenclature, or other practices?

Detect –2

More thoughts on logging…

- Integrate with existing event logging systems
  - E.g., Windows, syslog, syslog-ng
    - Also consider SNMP, CMIP or other management protocols
  - Watch out for free-form data
  - Cross-site scripting opportunity?
- Be careful about what is logged
  - Privacy laws
  - Coordinate this thoroughly with legal counsel
  - Case study: US retailer

Detect –3

- Support for "Infocon" tiers
  - Is state read-only or can your app affect elevated levels?
  - Log additional data for high tiers?
  - Random security questioning of users?
- Example: NTFS event logging options

React –1

Ok, the alarm has been pulled–now what?

- Evidence handling support
  - Log query and search tools
  - Tamper-evident seal for log extracts
  - Digital signatures of log data to enhance non-repudiation capabilities
- Quarantine of suspect data
  - Data uploaded to app stored away in a safe place

React –2

- Support for honeypots
  - Shunting attacker to alternate system
  - Stateful export to honeypot
    - Sanitized system data
    - Account credentials
    - Realistic

Infrastructure

- Be sure your logging architecture is solid
  - Centralized log server
    - Monitored by security team
  - Secure protocols
  - Non-repudiation of logged data
  - Mutual authentication
    - How do you know you're talking to the logger
  - What's the performance impact?
    - Separate admin data from production

Legacy apps

How do you improve the auditability of your legacy apps?

- Application firewalls can help to a degree
  - Most are exclusively for web apps
- Must have intimate knowledge of how the app works in order to be useful
- Event logging is a trivial and natural add-on this way

Getting started

Don’t wait for “them” to come to you

- Seek out the CSIRT at earliest stage of the dev process
  - Coordinate features, logging, etc.
  - Inventory of what gets logged is vital
  - Interface with IDS data/team to ensure compatibility with app logging data
- Seek out General Counsel or privacy officer
  - Ensure logging is in compliance
  - May need to be different by region

The difference

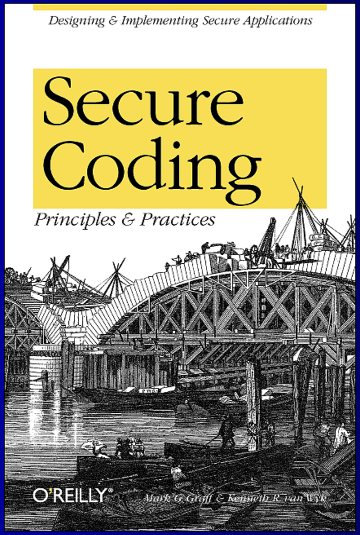
You're thinking that a lot of these things seem pretty extreme. Some are.

However, all of them are borne from experience in the trenches.

A well auditable app will make an enormous difference when the alarms go off.

Kenneth R. van Wyk
KRvW Associates, LLC

Ken@KRvW.com
http://www.KRvW.com


Designing & Implementing Secure Applications
Secure Coding
Principles & Practices
O'REILLY
Mark G. Graff & Kenneth R. van Wyk