Experiences with High Performance Intrusion Detection in the HPC Environment

Cray User’s Group 2011 Presentation
Jim Mellander
Scott Campbell
NERSC
• NERSC Description
  – Operational Philosophy
  – Science-Driven Cybersecurity
• Cybersecurity Solutions
  – Bro IDS
    • Clustering Solution
  – Inspection of Encrypted Sessions
    • Instrumented SSHd
• Wrapup and Questions
• Open Science
  – Maximum productivity for Users
  – Minimum restrictions on Usage

• Cybersecurity Operational Principles
  – No firewall for access to Big Iron
    • Too restrictive, prohibitive at high-bandwidth
  – Reusable Credentials
    • We presume that hackers will gain access to them
  – Rapid Response and Mitigation, rather than a priori restrictions
• **Bro has been continually under development since 1996**
  – Open-source platform for in-depth monitoring on commodity hardware
  – Used for production IDS operations throughout this timeframe

• **Focus is on:**
  – Application-level semantic analysis (rather than analyzing individual packets)
  – Tracking information over time

• **Strong separation of mechanism and policy**
  – The core of the system is *policy-neutral* (no notion of “good” or “bad”)

• **Activity-based analysis model**
  – Operators program local policy using *domain-specific language*
  – Bro logs all activity comprehensively
The Bro Network Intrusion Detection System (2)

• **Bro’s analysis model differs fundamentally from other NIDS**
  – Doesn’t (primarily) rely on Snort-style signatures nor on anomaly detection
  – Can be used to monitor non-network traffic as well

• **Bro is specifically well-suited for scientific environments**
  – Extremely useful in networks with liberal (“default allow”) policies
    • Can reactively block threats
  – High-performance on commodity hardware
  – Supports intrusion prevention schemes
  – Open-source (BSD license)

• **It does however require some effort to use effectively**
  – Fairly complex, script-based system
  – Requires understanding of the network
  – No GUI, just ASCII logs
global ssh_hosts: set[addr];

event connection_established(c: connection)
{
    local responder = c$id$resp_h;  # Responder’s address
    local service = c$id$resp_p;    # Responder’s port

    if ( service != 22/tcp )
        return;  # Not SSH.

    if ( responder in ssh_hosts )
        return;  # We already know this one.

    add ssh_hosts[responder];  # Found a new host.
    print "New SSH host found", responder;
}
• NIDSs have reached their limits on commodity hardware
  – Need to do more analysis on more data at higher speeds
  – Single commodity system just cannot cope with >1 Gig packet streams
• Key to overcoming current limits is parallel analysis
  – Volume is high but composed of many independent tasks
  – Need to exploit parallelism to cope with load
• To address the challenge, we present the Bro Cluster
  – Allows us to continue operating the Bro NIDS on commodity hardware
The Bro Cluster Approach

- Load-balancing approach: use many boxes instead of one
- Most NIDS provide support for multi-system setups
- However, instances tend to work independent
  - Central manager collects alerts of independent NIDS instances
  - Aggregates results instead of correlating analysis
- **The Bro cluster works transparently like a single NIDS**
  - Gives same results as single NIDS would if it could analyze all traffic
  - No loss in detection accuracy
  - Scalable to large number of nodes
  - Single system for user interface (log aggregation, configuration changes)
Cluster Components

• **Backend – worker nodes**
  – Running Bro as their analysis engine
  – Using essentially the same configuration as before, just on a slice of traffic
  – Bro provides extensive communication facilities for sharing of low-level state
    • Just mark an analysis variable as *synchronized* and its value will be propagated

• **Frontend**
  – Distributes traffic across backends
    • Software based on open-source Click modular router platform or BPF filtering, or
    • Customized appliance implementing MAC address rewriting in hardware, then putting traffic on a switch
• **Proxy**
  – Communicates state changes throughout nodes.
    • Communication mesh is $O(n)$ vs. $O(n^2)$ connections.

• **Manager**
  – Interactive interface for installation, configuration, tuning, logging, ...
  – Distributes traffic across backends
• Allows for Expansion of IDS capabilities to 10G and beyond
  – Adding nodes allows for splitting increasing traffic across more analysis nodes.
  – Multicore systems can run multiple analysis nodes.

• 100G still presents challenges
  – Individual nodes cannot keep up with a single high-speed flow.
    • Perhaps decide that a flow is uninteresting from a forensic standpoint, and stop analyzing.
      – How much can you get from an encrypted SSH session anyway?
IDS monitoring of clear-text sessions was highly effective.
- Ex: unset HISTFILE
- Indicators of ‘hackish’ activity

Capturing interactive data was also quite helpful forensically.
- Capture tools.
- Capture the state of files before/after editing.
- Files that were edited
- Command sequence executed.
• SSH encrypts entire session in transit
  – Hackers can no longer sniff useful session traffic off the wire.
  – Unfortunately, IDS operations no longer have insight into session traffic either.

• Traffic is necessarily decrypted at the endpoints
  – However, the IDS no longer has visibility into the session, until …
As we control the Server side, why not fork off a copy of the session after decryption to our IDS?

– Devil is in the details, but …
– … we can then leverage the IDS capabilities used in the clear-text era
  • Yes, hackers still use `unset HISTFILE`
– Don’t want to impede the user experience in any way.
  • Preserve user experience.
  • Enhance network throughput by incorporating PSC performance mods.
  • Avoid introducing additional failure modes or security exposure.
Instrumented SSHd
• Failure of downstream networking or software can not effect the users ssh experience.
  – Designed to lose security data before degrading the user experience.
  – Non-blocking write in sshd for sending data to a local stunnel socket.
  – Stunnel socket has aggressive timeout to avoid buffering issues on the sshd side.
Transcript of login (1)

```
1286227677.446452 #52374 - 128.55.128.185 128.55.128.187 127.0.0.1  25471
  ssh_connection_start 128.55.19.91:54703/tcp > 128.55.128.185:22/tcp
1286227677.861749 #52375 - 128.55.128.185 128.55.128.187 127.0.0.1  657910655
... auth_ok client publickey 128.55.19.91:54703/tcp > 128.55.128.185:22/tcp
... new_session SSH2
... new_channel_session pty-req
... new_channel_session shell
... data_server Last login: Mon Oct  4 11:31:18 2010 from 128.55.19.91
... data_server
... data_server NOTICE TO USERS
... data_server
... data_server States Government. It is for authorized use only. Users (authorized or
... data_server unauthorized) have no explicit or implicit expectation of privacy.
... data_server
... data_server intercepted, monitored, recorded, copied, audited, inspected, and
disclosed
... data_server to authorized site, Department of Energy, and law enforcement personnel,
... data_server as well as authorized officials of other agencies, both domestic and
foreign.
```
... data_server By using this system, the user consents to such interception, monitoring,
... data_server recording, copying, auditing, inspection, and disclosure at the discretion
... data_server of authorized site or Department of Energy personnel.
... data_server... data_server disciplinary action and civil and criminal penalties. By continuing to use
... data_server this system you indicate your awareness of and consent to these terms and
... data_client ka\^h\^hls
... data_server ls
... data_server instrumented-ssh.tar
... data_client exit
... data_server [35m[clant@[1msg2[m[35m] [32m[1m~[m[0m[1m >[m exit
... data_server
1286227709.639910 #52375 - 128.55.128.185 128.55.128.187 127.0.0.1
657910655
ssh_connection_end 128.55.19.91:54703/tcp > 128.55.128.185:22/tcp
Sample Bro Alerts

Mar 4 19:55:44 SSHD_Hostile #5068 0 53183_host_22 6529
user @ 0.0.0.0 -> 0.0.0.0:22/tcp command: unset HISTFILE

Mar 4 20:10:23 SSHD_Hostile #5068 0 53183_host_22 6529
user @ 0.0.0.0 -> 0.0.0.0:22/tcp command: shellcode=( # by intropy <at> caughq.org

Mar 4 20:10:23 SSHD_Hostile #5068 0 53183_host_22 6529
user @ 0.0.0.0 -> 0.0.0.0:22/tcp command: "x40x82xffxfd" # bnel <shellcode>

Mar 4 20:10:23 SSHD_Hostile #5068 0 53183_host_22 6529
user @ 0.0.0.0 -> 0.0.0.0:22/tcp command: execve("/usr/bin/passwd",[],{"EGG":egg+shellcode,"LC_TIME":bof})
... data_server user@host:/tmp/.tmp> rcp
1p@0.0.0.0:forker.c .
... data_server user@host:/tmp/.tmp> gcc -o f forker.c
... data_server forker.c: In function 'main':
... data_server forker.c:19: warning: incompatible implicit
declaration of built-in function 'exit'
... data_server forker.c:27: warning: incompatible implicit
declaration of built-in function 'exit'
... data_server forker.c:39: warning: incompatible implicit
declaration of built-in function 'exit'
Bro and Instrumented SSHd

- Have proven highly effective at protecting NERSC assets
  - Bro, as a network monitor, acts as a reactive firewall, inserting ACLs into router upon signs of trouble, …
  - … and in conjunction with Instrumented SSHd, allows very rapid detection and response to hacking activities.

- Reliance on rapid response and mitigation, rather than prevention
  - Except when it’s a no-brainer
    - Windows traffic hitting our Big Iron – no thanks.
• Please contact me at:

jmellander@lbl.gov

Questions?