Honeynet Data Analysis:
A technique for correlating sebek and network data

Edward G. Balas
Indiana University
Advanced Network Management Lab
6/15/2004
About the Author

• Edward G. Balas
  – Security Researcher at Indiana University’s Advanced Network Management Lab.
  – Honeynet Project Member
    • Sebek project lead
    • Honeywall User Interface project lead

• Research Sponsorship
  • This materials based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-2-0221. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.
Roadmap

• Honeynets are an idealized forensic testbed
• These testbeds have lead to a new data capture tool called Sebek.
• The volume of data has precluded use in operational environments.
• Describe efforts to solve issue by enhancing Sebek.
• Hope to provide quicker examination of data
• May yield a viable tool for forensics.
Introduction to Sebek

• Sebek Data Capture tool
  – kernel space tool that monitors sys_read call
  – covertly exports data to server.
  – used to monitor keystrokes, recover files, and other related activities even when session encryption used.
  – http://www.honeynet.org/tools/sebek/
Sebek Illustrations

- top left shows general architecture
- bottom left provides illustration of how Sebek gains access to sys_read data.
What the data “looks” like
Existing Capabilities

• What this gives you
  – Keystrokes
  – Files copied to system with session encryption
  – Burneye passwords
  – Read activity for each process.

• What is missing
  – Way to filter or navigate the volume of data
  – Sense of relationship between processes
  – Correlation to IDS or other network events.
  – Names of Files associated with File Descriptor
Enhancements to Sebek

• Record Socket Information
  – allows us to correlate network events to the associated process, user and even file descriptor on a box running sebek.

• Record Fork and Parent PID information
  – allows us to rebuild the process tree
  – combined with Socket Info, provides a fault tree.

• Record all files Opened
  – identify all files “touched” in association with an event.
Socket Monitoring

• To correlate network connections to process / file number we added the ability to monitor the sys_socket call.
  – in Linux, all socket calls are multiplexed through one generic socket call.
  – gained access using the same technique as used with sys_read.
  – this provided a mapping of:
    • src/dst ip endpoints for a connection
    • src/dst ports and protocol
    • state of connection.
    • Related Process, File No, etc.
Parent PID tracking

• Record the process inheritance tree by reporting the Parent PID along with the PID
  – Each sys_read provides the Parent PID
  – Each sys_fork provides a mapping as well.
    • needed because not all processes read before forking.
Data Analysis

- Honeynet data analysis and the analysis of network based intrusions are quite similar.
- Multiple Data types examined
  - Network traffic logs
  - IDS / Event logs
  - Disk Analysis
  - Sebek or other keystroke logs
- Time consuming and error prone.
Three steps in analysis

- Collect/Screen
  • Identify raw data of interest

- Coalesce
  • Combine data from different data sources, identifying cross data source relations and providing some type of normalized access to the data.

- Report
  • Identify central themes, screen out superfluous data.
How it is done today

• Each data type has its own analysis tool
  – causing a stovepipe effect.
  – each data set goes through the 3 steps in isolation.
• Switching data sources causes wetware context switch.
• Relations manually discovered and expressed to each tool for screening by analyst.
• No automatic way to track interesting sequences across data sources.
Why this is no good

• Labor intensive
  – I am lazy
• Error Prone
  – I am sloppy
• Lots of menial work being done by a human
  – I paid a lot for this computer
Where we want to be

• Shift the Screening and Coalescing burden to the computer.
• Focus human effort on tasks best suited to the human.
• Provide an interface that supports the analyst’s workflow.
• Provide a system that may have use in production networks.
Improving Data Analysis

• The new data coming from sebek allows us to automatically relate network and sebek data.

• To automate coalescing we developed a backend daemon called Hf bw.

• To demonstrate the impact of these capabilities on reporting, we developed a web based user interface named Walleye.
The challenge facing Hf bw
Hf bw Overview

• Fancy perl daemon, which consumes multiple data streams.
• Automates the process of Data Coalescing.
• Inputs:
  − Argus data
  − Snort IDS events.
  − Sebek socket records.
  − p0f OS fingerprints.
• Outputs:
  − normalized honeynet network data uploaded into relational database.
Hf bw Illustration
What this gives us.

- **Automatic identification**
  - Type of OS initiating a network connection
  - IDS events related to a network connection
  - IDS events related to a process and user on a host.
  - Point where non-root user gained root access.
  - List of files associated with an intrusion
  - Sense of Attribution between 2 related flows on a monitored box.

- **Operate at higher level where we can scale to support operational networks**
  - using Argus central theme of an event sequence can be identified without having to examining packet traces.
  - When packet traces needed, argus info helps facilitate retrieval.
Reporting with Walleye

- Perl based web interface
- Provides unified view
  - Network “f bw” connection records
  - IDS events
  - OS Fingerprints
- Allows user to jump from network to host data.
- Visualizes multiple data types together.
Looking closely

- host x.x.x.31 attacked x.x.x.25 on its https port.
- x.x.x.31 was a linux host.
- The attack matched the OpenSSL worm signature and triggered 2 additional alerts that indicate the attacker gained www and then root access.
- If we click on Proc View, we jump to a high level view of related process activity.
Sebek Data related to Snort Event: SID=1, CID=1520
What you are seeing

• Display shows a process tree and its associated IDS events.
  – created by querying on a single IDS event.
  – Yellow Boxes are root processes
  – Cyan Boxes are non-root processes
  – Red Boxes are IDS events
  – Red Arrow represents direction of flow associated with event
    • Only displaying IDS related flows.

• Graph automatically generated from DB with graphviz tool from ATT.

• Notice anything odd about the graph?
Walleye tracked intrusion across 2 honeypots

• Both the .25 and .26 honeypots were running the enhanced version of Sebek.

• We are able to provide a sense of attribution in situations where all stepping stones are running Sebek.

• Based on fault tree we could then click on a yellow box and then jump into the sebek interface.
Old question made easy

- What happened after the intrusion?
  - Use IDS event as index into process tree.
  - All related files will be liked to that tree.
  - All files “touched” as part of the intrusion will be related to that tree.
  - Sequences that span 2 hosts can be automatically identified via common network connection.
Features

• Identify descendant f bws or sebek events related to a given event.
• Identify ancestral f bws or sebek events related to a given event
• Effectively, the combination of the two allow us to filter all data which can not be related to an event of interest.
• Find all files opened by any process in a process tree.
Current Status

• Sebek
  – socket code in linux client rather stable
  – parent PID tracking currently missing some data for processes that fork and don’t read (easy to fix)

• Hf bw
  – few bugs and it's not syslog friendly

• Walleye interface
  – a few bugs, look and feel not 100% happy with
  – not yet integrated with conventional analysis tools.
  – doesn’t provide way to access raw packets
Future work

- **Sebek**
  - track fork call so that we always get a view of the process tree
  - look at various anti-anti-sebek options.

- **Hf bw**
  - testing, lots of testing.
  - evaluate attack resistance

- **Walleye**
  - get UI to better support workf bw
  - provide alerting
  - provide some summary reports
  - clean, debug, document
  - integrate with existing tools where sensible.

- Get everything to work on the Honeywall CDROM!
Taking this out of the Honeynet context

• Sebek is a good tool for post intrusion intelligence gathering on an intruder
• On a production box it generates great amounts of data, making it difficult to use.
• With previously mentioned enhancements, Sebek may be a more viable tool, due to its improved coalescing and screening.
• The ability to relate 2 flows to and from a host via a common process tree may be more valuable than the ability to record keystrokes?
Related works

• Covert
• Anti Sebek foo
CoVirt

• CoVirt and the BackTracker system
  – Enhanced UML system allows host to monitor guests system call activity.
  – “Automatically identifies potential sequences of steps that occurred in an intrusion.”
BackTracker output
References to attack techniques:
