Node Router Sensors: What just happened?

GIAC (GCIH) Gold Certification

Author: Kim Cary, kim.cary@pepperdine.edu
Advisor: Rich Graves

Accepted:
November 20, 2016

Abstract

When an airliner crashes, one of the most important tasks is the recovery of the flight recorder or black box. This device gives precise & objective information about what happened and when before the crash. When an information security incident occurs on a network, it is equally important to have access to precise information about what happened to the victim machine and what it did after any compromise. A network of devices can be designed, economically constructed and managed to automatically capture and make available this type of data to information security incident handlers. In any environment, this complete record of network data comes with legal and ethical concerns regarding its use. Proper technical, legal and ethical operation must be baked into the design and operational procedures for devices that capture information on any network. These considerations are particularly necessary on a college campus, where such operations are subject to public discussion. This paper details the benefits, designs, operational procedures and controls and sample results of the use of "Node Router Sensors" in solving information security incidents on a busy college network.

1. Introduction

The genesis of this project was in response to network growth. As our network grew, the network team decided to change from a core-routed to distribution node-routed architecture. This decision meant that intra-VLAN and inter-subnet network traffic would no longer have to pass a single control point within the core router. This single core router allowed network traffic capture (aka "sniffing") to determine the cause of problems or the source of attacks. The information security team requested the creation of distribution node routers include the deployment of distribution node capture points, to retain the ability to capture this local network traffic. We originally called these capture points "node sensors," and these are referred to for clarity in this paper as “Node Router Sensors” or NRS.

To get this addition approved, the NRS had to be economical both in capital and maintenance and operational costs. The security team had no budget at the time beyond salaries – which implied piggybacking on the already committed network expansion budget for capital and maintenance. Further, the security team needed a system that had low operational maintenance requirements, as their time was already fully tasked. The design set forth below not only met these cost goals but proved to be a valuable tool in incident handling.

Table 1 shows the six phases of the step-by-step consensus incident handling process (SANS, n.d.):

Phase 1: Preparation	Phase 4: Eradication
Phase 2: Identification	Phase 5: Recovery
Phase 3: Containment	Phase 6: Lessons Learned

Table 1: *Six phases of the incident handling process*

The *creation* of the NRS network was part of the preparation phase of our incident handling process. The preparation phase takes place before any work on a specific incident begins. This phase identifies the tools, authority and procedures to be used in handling, or preventing (Scarfone, 2008), incidents. The maintenance of this NRS network also falls into the preparation phase of incident handling.

Kim Cary, kim.cary@pepperdine.edu

In *operation*, most applications of our NRS would be classed primarily under the identification phase of the incident handling process, showing the exact content of suspected bad traffic. However, an NRS network might also be used to measure the effectiveness containment and eradication efforts by providing visibility into whether incident traffic recurred.

We designed the NRS as a host that would provide our information security team with an interactive point to initiate traffic captures out at the local routers. The NRS would give us a local station to examine such things as denial-of-service source IPs, rogue DHCP servers, the content of local malicious traffic or to identify the exact content and destination of information disclosures. However, as deployed the NRS improved on the single sniffer point available on the core-routed network in two ways. First, the core-routed single point capture was performed using a *temporary* SPAN of the traffic on one or two VLANs, with capture on a laptop temporarily attached to the core router; the distributed NRS network used *permanent* distribution router monitor ports connected to secured servers. Second, with fixed hard drives on each server, it had enough capacity to capture traffic from all user VLANs on the router. This VLAN data was recorded continuously in a first-in-first-out circular set of buffer files. The buffer holds, depending on location, between 72 hours to 3 weeks of our network traffic.

With the NRS systems deployed at the nodes, we were able to not only see what was happening in near real time on the network but also look up to three weeks in the past to assess what had taken place on the network during an incident! This data buffer increased the breadth of types of incidents we could investigate. We went from a system, which was only practical to use for ongoing attacks to one that could be used to investigate overnight IDS alerts and transient event alarms. Our NRS server is a simple design, which is inexpensive and relatively easy to maintain. With the proper planning for security and ethical controls, any security organization can apply this design to organization networks to find out "What just happened?"

Kim Cary, kim.cary@pepperdine.edu

2. Node Router Sensor Network

While the information security office manages NRS as a network, systems in this 'network' do not share the data captured. However, the NRS are all maintained using configuration management software that links them into a loose cluster for targeted maintenance and configuration changes. Each system captures from its SPAN port(s) on the router and retains the data locally. Data extraction is done locally on each system and then downloaded to the analyst's workstation for review. Proper design, construction, and operation of the NRS ensure secure and efficient support of incident investigations.

2.1. System Design

Given the forensic intent of the NRS as a replacement to core router sniffing, as well as the limited financial resources available, keeping the system design simple by matching deployed functions to the purpose was a critical efficiency.

Each NRS system consists of an inexpensive rackmount server, with two or more Ethernet ports -- a management port and gigabit capture port(s) for SPANs from its node router. Each system is configured via the underlying native CentOS utilities, overlaid with Jamie Cameron's Webmin freeware for configuration management and software distribution. A startup script starts and keeps alive a daemonized capture service which captures all data from the SPAN(s) to a set of files in a directory designated for each capture port. Appropriate controls are used to ensure secure and ethical operation. The analysts log into the NRS to extract data and download it to their local encrypted workstation for analysis. The examples in section three show that this simple system is versatile as an incident handling tool, as well as a device that provides data for analyzing network anomalies.

The hardware to run a simple capture system can be much less expensive than other security systems, such as an IDS. If one has intrusion detection capability elsewhere on the network, a simple capture recorder like our NRS could be an inexpensive supplement. Our second-generation rackmount NRS hardware costs about $500. The hardware for an open source IDS, like Security Onion (Burks, 2016), for the equivalent network, requiring eight cores and 128Mb RAM, could be ten times as much. Of course, purpose-built hardware and software from IDS vendors would be even more expensive.

Most importantly, capturing *all* traffic, not just alerted traffic, is the primary value of the system. This full capture allows for post-mortem analysis in the case where, say, a client workstation got infected without triggering an IDS alert. The full capture record can look back in detail at any probe steps, the downloaded binary and the origins or destinations of the attack.

2.2. Construction

All systems utilize two or more 1Gb network ports with one port for management and one or more for network capture. There is also a boot drive for CentOS and its logs, and a higher capacity capture hard drive to hold the captured data.

In the original purchase, the eleven NRS servers were HP DL145 1-U rackmount server systems with 2Ghz Opteron processors, dual gigabit NICs and 2Gb RAM each. The systems were equipped with two 80Gb SATA drives, one for boot and one for capture. Each of these systems cost about $1200. The servers could have been cheaper with a tower form factor, but they needed to mount in the same rack with each node router.

The second generation NRS was deployed in new network areas or to replace failed HP DL145 hardware. These systems are SuperMicro Atom 525 based half-depth 1-U rack mount systems. These proved even more reliable but were still able to capture at full network speed (1Gbps). Each of these systems, including the drives used for capture, cost about $500.

Over time, the security team discovered that NRS systems required larger and more rugged capture drives. All the capture drives were replaced with 1Tb or larger SATA spinning hard drives designed for use in equipment that continuously writes to disk, such as DVRs. Such drives are available from several manufacturers at a small premium over consumer drives. In six years of operation, we have not had to replace any of these upgraded drives due to failure.

Various boot drive strategies have been tried, beginning with 80Gb spinning drives, then internal 16Gb USB flash & mSATA SSD drives on the second generation

Atom boxes. The current standard for boot drives is small 60-80Gb SATA SSD. These drives generate less heat than spinning drives and better reliability than USB flash.

NRS servers deployed in two overseas locations utilize 4-core Xeon-based SuperMicro 1-U half-depth rackmount systems with 32Gb RAMcosting around $1500. This more powerful hardware allows the potential for future functions beyond just capture (e.g. log receiver/forwarder at remote campus) or to accept capture from more than one heavily utilized network (e.g. when one centrally located router replaces four routers in separate locations).

Despite the hot and dusty environment or unprotected power in some of the locations, reliability of the NRS servers has been strong. The fan-less SuperMicro Atom 525 servers seem more robust than the DL145. The terminal output quoted below shows the current uptime champion, which is in a datacenter monitoring our busiest network.

```
# uptime
16:44:36 up 688 days,  6:58,  1 user,  load average: 0.38, 0.26, 0.20
```

Note that despite capturing a high volume of data the load average is low, since the work is mostly writing from network to disk. This measurement justifies the low-cost hardware of the system.

2.3. System Connection

Since new routers were purchased as part of the distribution router network architecture, it was not necessary to buy hardware to tap network cables. Instead, we chose to use a SPAN session to copy data from the appropriate VLANs to the NRS's capture port. There are potential inaccuracies and losses with a SPAN compared to a passive network tap (Fortunato, 2016). However, technically, using SPAN did not significantly impede the purpose of examining traffic and the economics of purchasing tap hardware would have worked against getting the system built. Losing timing accuracy or packets on overloaded links or malformed traffic packets using SPAN was a risk we would accept.

In the SPAN configuration example quoted below VLAN 72 is the user network and VLAN 772 is the network management network. Traffic traversing both networks is recorded, in this configuration

```
TAC-NODE-72-1#show monitor session 1
Session 1
---------
Type: Local Session
Source VLANs:
    Both: 72,772
Destination Ports: Gi2/0/21
    Encapsulation: Native
          Ingress: Disabled
```

SPAN configurations and physical ports used are recorded offline for each of the NRS in the case of configuration changes or accidents that remove the monitor session from the router.

2.4. System Management

The reader may think the preceding paragraphs represent a lot of system administration for a security analysis paper. However, if the security team must perforce become system administrators to create and deploy a valuable security system, it would behoove them to have a good design that does not need constant intervention to keep secure and online. This way, once deployed, they can concentrate on using rather than maintaining the system. Thus, the details managing the NRS machines become relevant for security professionals.

The NRS systems update the OS from a local mirror of the CentOS update repository using the yum utilities and scheduled tasks shipped with the OS.

A configuration automation tool is invaluable in increasing the efficiency of system administration tasks (Ryd, 2016). Efficiency is an aid to security; necessary tasks may be completed before unmaintained systems can be exploited. Obviously, automatic system security updates, such as scheduled *yum* tasks are basic to operating system security patching. However, the challenges with multiple machines are in keeping users, scripts, and configurations in sync. There are several good free-to-use system configuration tools for Linux, such as Salt (SaltStack Documentation, 2016) and CFEngine (CFEngine Community Edition, 2016). Webmin has been our configuration

management tool for NRS systems. Systems can be added to a Webmin cluster, and many necessary operations can be applied to all systems in the cluster (Cluster Webmin Servers, 2010). Common operations can be executed on all NRS systems at once. Sample operations we have performed with Webmin include: removing a user who is no longer with the security team; adding a personal account and password for a new user to each system; updating Linux kernel; updating the Webmin management software; deploying a new version of the capture startup script;

For security reasons the Webmin broadcast discovery method is turned off and the firewalls of each system are set to receive commands only from Webmin on the dedicated master control server for the cluster.

The generation 1 NRS uses a bare bones base CentOS5 Server package configuration to which the team adds other necessary packages. Those systems were cloned from the original disk and configured for their destinations from the console. Generation 2 hardware has used CentOS6, and a tarball archive used to apply the capture components. Given the longevity of these systems – some systems deployed in 2007 are still in production in 2016 – a base operating system with long-term support for security patches is appropriate.

With geographically distributed systems like this, there is no quick way to get to a power button for hard resets. Even the best systems may hang from time to time, and several NRS have become unresponsive due to disk problems, being turned off by telecom techs or power outages leaving the system at a filesystem check prompt. In a case like this, a lights-out management (LOM) system built into the motherboard has saved time and steps beginning with the generation 2 hardware. The LOM is a small operating system that can control power and also provide a remote boot console environment for viewing and interacting with the boot process to diagnose or correct error conditions. The main system can crash, but the LOM is most always still listening on its separate IP address and able to take power management and other commands. The LOM in the SuperMicro systems was easy to set up and configure so that it would only accept commands from secure subnets. The LOM has been used several times to reboot a

hung system or, more frequently, boot ones that got turned off during maintenance of a rack and were not turned back on.

2.5. Autocapture Setup

The NRS's value, like that of an aircraft flight recorder, is in recording as much recent relevant activity as is practical. The NRS is set up to keep as much of the captured data as possible for the network(s) that it is recording. In order not to fill the disk and thereby halt recording, there must be a first-in-first-out (FIFO) data buffer that limits the size of capture to a specified subset of available disk space.

The NRS servers use daemonlogger software to create the FIFO capture buffer of a particular size. Daemonlogger is a packet capture program written by Martin Roesch (Roesch, 2008). This software can record captured data from a network port in a circular disk buffer.

The NRS setup uses the daemonlogger's "write to disk" mode of operation. In this application, the capture file is set to be 100Mb (-s) before rotation, and to delete the oldest files to guarantee that the capture disk is no more than 95% utilized (-M) and no more than 25000 (-m) capture files exist. The 100Mb size is easily manipulated with the packet analysis tool, Wireshark. The command line generated by the customized startup script reflects these limits below.

```
/usr/local/sbin/daemonlogger -i eth1 -u tcpdump -g tcpdump -M 95 -m 25000
-s 100000000 -r -l /data/concap/all -n pcc-all.pcap -f
/usr/local/etc/bpf.conf
```

The other significant switches in the command include the capture port (-i), use of the circular buffer mode (-r, ringbuffer), the directory for the capture files (-l), and the prefix to be used for the serialized capture file names (-n). The capture file names, in this case, would be appended with the epoch timestamp of the creation time of the file (e.g. pcc-all.pcap.1478297811, pcc-all.pcap.1478297858, and so forth.)

Some data in SPAN traffic may need exclusion from capture. A bpf.conf file can hold a packet filter. In the NRS this file is deployed on every system, for simplicity, but is typically empty. One example use of the bpf is on the NRS for the router serving our WiFi access point controllers. Both the encrypted outside of a GRE tunnel and the

decrypted contents of the tunnel are being sent to the NRS capture port at this router. The encrypted tunnel traffic is excluded with the statement "not proto gre" in the bpf.conf file. That exclusion is used to prevent recording duplicate traffic from shortening the length of data retention on the capture sensor.

If multiple capture ports are used, as they are on some of our NRS, corresponding startup files are used to start a separate daemonlogger instance and target directory for capture files. Capture history limits are adjusted accordingly.

The following output shows commands used in a manual check of capacity & data history on an NRS. This type of check is something that is ripe for future automation.

```
# ls /data/concap/all | wc -l
24669
# ls -lt /data/concap/all | head -3
total 2593176196
-rw-r-----. 1 tcpdump tcpdump  10460848 Aug 14 17:36 tac1-all.pcap.1471219531
-rw-r-----. 1 tcpdump tcpdump 100001420 Aug 14 17:05 tac1-all.pcap.1471205772
# ls -lt /data/concap/all | tail -3
-rw-r-----. 1 tcpdump tcpdump 100001500 Jul 20 02:40 tac1-all.pcap.1469007627
-rw-r-----. 1 tcpdump tcpdump 100000465 Jul 20 02:40 tac1-all.pcap.1469007623
-rw-r-----. 1 tcpdump tcpdump 100000086 Jul 20 02:40 tac1-all.pcap.1469007620
# df –h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             2.7T  2.5T  138G  95% /data
```

In the output above, command results show there are 24669 100Mb capture files, covering about 25 days of network data and consuming approximately 90% of the disk space in the 3Tb capture disk.

2.6. System Controls

Any full network capture system presents a target for abuse both by external actors and insiders. As such the NRS requires well-designed technical and operational security controls. Suggested measures include technical and ethical standards.

2.6.1. Physical Controls

All NRS sites have controlled physical access limited to authorized personnel with all but one having a lock on the room containing the sensor. The systems have no optical drives. There are USB ports, retained for keyboard use, which does present a small risk should someone get physical access.

2.6.2. Technical Controls

The management ports for the NRS connect to either management only RFC 1918 networks or inside firewalled data centers to inhibit traffic snooping. All management communications are encrypted SSH or HTTPS. Host firewalls are configured such that they only allow access from secure technician networks. The firewall configurations are uniform and distributed from the master Webmin server.

Security updates are managed automatically via scheduled update jobs against our private CentOS mirror, except for Kernel updates. These are applied as needed via Webmin cluster master.

Capture ports have no IP addresses so that services are not presented that can be attacked. Operators do not do packet interpretation of captured data on the NRS, so flaws in packet analysis software are a low risk since technicians use a properly patched version of Wireshark on analysis workstations.

2.6.3. Ethical Operational Controls

Each IT employee signs a code of conduct that prohibits browsing data and use of data for non-assigned duties. This document applies to our security analysts and the NRS network as well. However, the existence of an exact copy of all data on a network warrants further protection. ISO uses a log to record when we access the sensors to collect data. In our office manual for the NRSs (node-sensor-sysinfo.doc), atop the operational instructions for following access logging procedure is stated:

> **WARNING: before logging in to extract capture data, open a JIRA case for the incident and note the sensor(s), purpose (scope of what you are looking for) and findings of your investigation!**

This procedure applies to each investigation – the reminder is right where it needs to be, with the instructions for extracting data. This instruction can be cross-checked against login and command events recorded on the central Syslog servers. The intent of the incident and system logs is to make the use of the system auditable. It is also a directive and deterrent control, which serves to keep the analyst in mind of the standards for the work and consequences of violating the standards (Hansche, Berti, & Hare, 2003, p. 340).

2.7. Extracting Target Data

The NRS records to disk a simple circular buffer of the latest data sent to it by a router SPAN port. Instructions to extract captured data are below. Of interest is the necessity to use the mergecap utility to concatenate files from the period of interest, before using tcpdump to select the interesting data from that period. Our instruction steps, found under "Data Extraction Procedure" in the Appendix, go through the options of the various utilities for our common use cases. Several aspects of the procedure are worth explaining. The *mergecap* utility, which is installed as part of the Wireshark package, is used to concatenate files for the period under investigation. The *tcpdump* utility is then used to extract traffic from the merged file for the system(s) of interest. Once data is extracted from the capture buffer files, the security analyst downloads it to the encrypted analysis workstation for review.

3. Sample Incidents Handled

Several incidents have been investigated using the NRS systems. Some were serious, and NRS was involved in several incident handling steps. Other examples are shown where the NRS is used in the identification step to prove that the potential security incident is a misconfiguration or false positive. The following synopses of investigations are provided as representative of the type of use found for NRS in incident handling

3.1. Identification of Trojan Symi Security Events

This incident showed the value of NRS in speeding up the accurate characterization of an attack. This *identification* led to the precise *eradication* of the startup file used to compromise workstations before an infection event could be triggered.

The security analyst with event review handling security events and alerts from Friday through the weekend noted some IDS events "MALWARE-CNC Win.Trojan.Symmi variant outbound connection" (SID 25511, from the Sourcefire VRT ruleset), which may have been false positives. The Monday event analyst noted more of the same events. Management review of these separate reports deemed there was enough increased activity present to initiate a system quarantine of one of the detected systems for a closer look at what seemed likely to be a security incident. The workstation

examined had a GATAK Trojan. Other workstations triggering the SYMMI event were examined, and the same finding was made. This increase was deemed enough to configure the network access control (NAC) system to block workstations with that SYMMI event. However, how did the Trojan get there? This question prompted the security team to look at the NRS data.

On consultation, the NRS capture for the workstations showed a common sequence of events.

1. Attacker IP connects and negotiates an SMB connection, mounts C:/, then closes the connection (RST/ACK).
2. Attacker IP then reconnects and tests for Documents and Settings/<userid>/Recent.
3. Attacker IP then tests for <userid>/start menu/programs and then creates a file with a name in the format <9-10numbers>.exe
4. This path test & write sequence repeats for the other users on the machine in alphabetical order.
5. Attacker logs off.

The file creation event is shown in Figure 1. In this case, ten random numbers are used in the filename; analysts on the incident referred to this file as the "numbers.exe" file, for convenience.

```
▸ SMB2 Header
▾ Create Request (0x05)
  ▸ StructureSize: 0x0039
    Oplock: Lease (0xff)
    Impersonation: Impersonation (2)
    Create Flags: 0x0000000000000000
  ▸ Access Mask: 0x00120196
  ▸ File Attributes: 0x00000000
  ▸ Share Access: 0x00000003 SHARE_READ SHARE_WRITE
    Disposition: Overwrite If (if file exists overwrite, else create it) (5)
  ▸ Create Options: 0x00000066
  ▸ Filename: Documents and Settings\cashier\Start Menu\Programs\Startup\1868626304.exe
```

Figure 1. Wireshark display of file creation event

The “numbers.exe” file that NRS showed was created in the start up items was absent on the examined station. However, on examining the behavior of the file when it was executed, it installed the GATAK trojan elements and deleted itself. At this point, about 35 machines had been infected, isolated by the NAC in response to the SYMMI IDS events, and would need to be re-imaged. However, workstations with the “numbers.exe” file did not show the SYMMI event, until the file was executed.

Kim Cary, kim.cary@pepperdine.edu

Further review of the NRS data during the *identification* phase, showed a common Attacker IP on our network. Further, a quick review indicated perhaps 200 workstations that had the "numbers.exe" file installed by the Attacker IP. An interview with the assigned user of the workstation at that IP confirmed risky behavior, namely, downloading "keygen" files to attempt to remove copy protection from programs. Examination of the "keygen" files showed them to be the source of the infection. The elevated administrative privileges of this user allowed the distribution of the file to over 200 other systems.

Thanks to knowing the file name pattern and target location from the NRS, the security team was able to deploy an endpoint management script in the *eradication* phase to clean out the startup folders network-wide. Statistics from the script show this prevented 189 other systems from being infected and needing reimaging.

Security team felt confident in the *recovery* phase NRS data could be examined to determine whether sample cleaned or rebuilt machines continued to exhibit infectious behavior. However, the Symmi IDS indicator did not recur, nor did the “numbers.exe” (according to the software inventory system) nor did the GATAK trojan return.

3.2. iPhone DHCP server

In this incident, our IDS reported DHCP REQUESTs were being made to a non-authoritative DHCP server, but when we went to find the device that may have been making the offers, its IP was not recorded in our network access control (NAC) system and could not be determined. As this event began to recur, there was also some complaint of WiFi clients being mysteriously unable to use the network during a current network session.

Searching for the MAC addresses involved in the NRS data, the device making the DHCP OFFER & ACK turned out to be an iPhone with a NAC registered IP and a second unregistered IP, which was handing out IP addresses in the valid WiFi IP range to other peers on the network. Of course, many of these were already in use and caused individual device network problems. Being able to use the NRS data to connect the MAC addresses our IDS alert was detecting with the IP addresses of the person with the misconfigured phone (he was trying to share his mobile wireless bandwidth with

colleagues) allowed the security team to *identify* the source of the unintended denial of service and address the problem.

3.3. Flash interactions with vietnampig.com

One user was flagged by the IDS for an attempted Flash exploit. "Attempted exploit" security event alerts are common in IDS, and the majority do not indicate real compromises on our network. Was the user infected, warranting a system rebuild, or not? NRS data helped in the *identification* phase by recording that the SWF stream downloaded other files. One of the downloaded files, extracted from the stream of data obtained from the NRS, though not detected in our brand of antivirus at the time, was identified as Troj/JSShell-AF. Therefore, the user's computer was reimaged to *eradicate* the infection.

3.4. IDS Review showed suspicious traffic

An IDS review showed suspicious network traffic originating from a system on an IP range dedicated to departmental servers. The system had generated a large volume of alerts based on public (SBL_DROP) and private intelligence sources indicating the system was accessing networks with a criminal or malware IP reputation. The student in charge of the system claimed this server was used only for research, but the security team wanted to verify that the traffic did not indicate a compromised system. Data from the NRS was part of the confirmation that the computer was spidering the web for legitimate research reasons. It had just run into some IP ranges with bad IP reputation while looking for legal documents, such as contracts, wills, and other agreements, to collect and catalog. The downloaded files were indeed not executable. NRS was used to help with identification that the data in question was not malware or a security incident.

4. Conclusion

Even in an organizational culture that values information security, budgets can be tight. This financial challenge can be true even for large schools, Universities, and non-profit organizations. The Node Router Sensor system is one approach to providing information security incident handling investigations support that is economical and efficient.

Kim Cary, kim.cary@pepperdine.edu

The economy in this design extends both to capital and operational costs. The hardware is less expensive than a typical desktop PC and the software not only free but easy to configure for analysts with some system administration experience (please see Appendix for sample startup and configuration examples). It leverages pre-existing and necessary network hardware (routers) as a source of data. Operationally, utilizing available automation and configuration management utilities the system can be set up to have a very low requirement for hands-on maintenance. It just is deployed and recording data until needed.

Once examined, the NRS network capture buffer neatly and efficiently and accurately answers questions about potential incidents. The system records not only traffic that generates a security alert but all traffic. This full record is vital in identifying incidents where significant damage might go undetected, as with the SYMMI detects. It is also effective in identifying when a detected event is not an incident, as with the legal research spider.

Future work with the NRS system might include further automation improvements. One such improvement could be alerting on the status of captured data (latest file too old, history less than 72-hour threshold) or data storage status (drive errors). Developing a system that is equally economical, relatively at least, for higher capacity (up to 10Gb) or higher utilization networks is another possible future improvement to the NRS. With future improvements, the automated full-capture NRS system can remain a high-value, low-cost aid to incident handlers that need to know "what just happened?"

Kim Cary, kim.cary@pepperdine.edu

5. References

Burks, Doug. (2016, September 21). *Hardware.* Retrieved from https://github.com/Security-Onion-Solutions/security-onion/wiki/Hardware

Cluster Webmin Servers. (2015, February 16). [Wiki] Retrieved from http://doxfer.webmin.com/Webmin/Cluster_Webmin_Servers

CFEngine Community Edition. (2016). Retrieved from https://cfengine.com/product/community/

Fortunato, T. (2016, April 13). *SPAN Port Vs. TAP: The Latency Impact.* Retrieved from http://www.networkcomputing.com/networking/span-port-vs-tap-latency-impact/1358909399

Hansche, S., Berti, J., & Hare, C. (2003). *Official (ISC)² Guide to the CISSP exam*. Boca Raton, Fl: Auerbach Publications.

Roesch, M. (2008). *Daemonlogger (README)*. Sourcefire. Retrieved from http://www.snort.org/snort-downloads/additional-downloads

Ryd, T. (2016, June 6). *How efficient is your IT infrastructure and what can you do?* Retrieved from https://cfengine.com/company/blog-detail/how-efficient-is-your-it-infrastructure-and-what-can-you-do/#sthash.ryXRZh1n.dpuf

SaltStack Documentation. (2016). Retrieved from https://docs.saltstack.com/en/latest/

SANS. (n.d.). *Incident handling step-by-step and computer crime investigation*. Retrieved from https://www.sans.org/security-training/incident-handling-step-by-step-computer-crime-

Scarfone, K. (2008, March). *Computer security incident handling guide*. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-61-rev1/SP800-61rev1.pdf

Kim Cary, kim.cary@pepperdine.edu

6. Appendix

Data Extraction Procedure

Steps: Determine what files you need, then use mergecap and tcpdump to extract the data

1. cd /data/concap
2. ls –lst
   - 2.1. Use this command to see the timestamps on the files in order (otherwise, it will wrap at some point).
   - 2.2. There are up to 2000 files so you should use a selector, like:
     - 2.2.1. | more (to see the format of the names)
     - 2.2.2. | grep vlan72-1[3-7]?? (to select files ending in 1300-1799)
3. **mergecap** [-a] -w - vlan72-1[3-7]?? | **tcpdump** -nn -s0 -r - -w /tmp/dhcp.pcap port 67 &
   - 3.1. **mergecap** collects the data from the files named in the regular expression
     - 3.1.1. –w -: send merged data to standard output.
     - 3.1.2. -a indicates filenames are specified in chronological order. Omit if they are not.
     - 3.1.3. vlan72-1[3-7]??: a selector for what dump files to use, interpreted
       - 3.1.3.1. file starts with vlan72-1
       - 3.1.3.2. next character is a number in the range from 3 to 7
       - 3.1.3.3. next two characters are any alphanumerics to end the file name
       - 3.1.3.4. matches vlan72-1300 through vlan72-1799
   - 3.2. **tcpdump**
     - 3.2.1. –nn: don't decode port or ip into names, leave numeric, not needed with "-w" but useful if you are going to look at the output.
     - 3.2.2. –s0: Write the entire packet, not the first few bytes.
     - 3.2.3. –r - : read from standard input
     - 3.2.4. –w: write to the file at path
     - 3.2.5. port 67: or some other BPF syntax (see man tcpcump) to select certain packets from the stream being generated by mergecap, in this case, those whose destination or source port is 67
4. (optional) **editcap** -d -A 'yyyy-mm-dd hh:mm:ss' -B 'yyyy-mm-dd hh:mm:ss' /tmp/dhcp.pcap /tmp/dhcp-nodup.pcap. The "-d" option removes duplicate packets, and the "-A" and "-B" options select a specific time range. Note that **editcap** cannot accept "-" as an input filename to read from standard input.
5. Transfer the file with Webmin or scp to your workstation from /tmp and analyze using Wireshark.
   - 5.1. E.g. http.request.method == "POST" is a nice Wireshark filter for a Trojan to check what posts may have been copied to the bad guys

Kim Cary, kim.cary@pepperdine.edu

concap Startup File

This is a script for starting up process that runs the script, which sets up the capture.

```
#!/bin/bash
#
# concap       Starts continuous capture with tcpdump.
#
#
# chkconfig: 345 99 10
# description: Concap runs a continuous tcpdump capture to files.
# pidfile: /var/run/concap-tcpdump.pid

### BEGIN INIT INFO
# Provides:          concap
# Required-Start:    $network
# Required-Stop:
# Default-Start:     3 5
# Default-Stop:      0 1 2 6
### END INIT INFO

# Source function library.
. /etc/init.d/functions

progpath=/usr/local/sbin/concap-start
progconf=/usr/local/etc/concap.conf
processname=concap-all
servicename=$processname
lockfile=/var/lock/subsys/$servicename
pidfile=/var/run/$processname.pid

RETVAL=0

start() {
        [ -x $progpath ] || exit 5

        umask 022
        echo -n $"Starting Continuous Capture: "
        daemon --check $processname --pidfile $pidfile $progpath $progconf
        RETVAL=$?
        echo
        if [ $RETVAL -eq 0 ]; then
                touch $lockfile
        fi
        return $RETVAL
}
stop() {
        echo -n $"Shutting down Continuous Capture: "
        killproc -p $pidfile $processname
        echo
        RETVAL=$?
        echo
        if [ $RETVAL -eq 0 ]; then
                rm -f $lockfile
                rm -f $pidfile
        fi
        return $RETVAL
}
rhstatus() {
        status -p $pidfile $processname
}
restart() {
        stop
        start
}
condrestart() {
        if [ -f $lockfile ]; then
                stop
```

```
                start
        fi
}
case "$1" in
  start)
        start
        ;;
  stop)
        stop
        ;;
  status)
        rhstatus
        ;;
  restart)
        restart
        ;;
  condrestart)
        condrestart
        ;;
  *)
        echo $"Usage: $0 {start|stop|status|restart|condrestart}"
        exit 2
esac

exit $?

cat /usr/local/sbin/concap-start
#!/bin/sh
# Start concap continuous capture in background.

conf_fn="$1"

if [ "x$conf_fn" = "x" ]; then
    echo "usage: $0 config_file" >&2
    exit 5
fi

. "$conf_fn" || exit 1

daemonlogger=/usr/local/sbin/daemonlogger

if [ "x$capdir" = "x" ]; then
    echo "$0: no capture directory" >&2
    exit 5
fi

cd "$capdir" || exit 1

if [ ! -x "$daemonlogger" ]; then
    echo "$0: $daemonlogger not found"
    exit 1
fi

touch $logfile
echo "Starting concap at `date`" >>$logfile
# env >>$logfile

umask "$umask"

/usr/local/sbin/run-daemon -p "$pidfile" -0 "$procname" -w 60 -r 15 \
  $daemonlogger -i "$interface" -u "$user" -g "$group" -M "$maxdisk" -m "$nfiles" -s
"$filesize" -r \
    -l "$capdir" -n "$capfilename" -f "$bpf" \
    >>$logfile 2>&1 </dev/null\\
```

concap-start

Script to start the continuous capture of network data.

```
#!/bin/sh
# Start concap continuous capture in background.

conf_fn="$1"

if [ "x$conf_fn" = "x" ]; then
    echo "usage: $0 config_file" >&2
    exit 5
fi

. "$conf_fn" || exit 1

daemonlogger=/usr/local/sbin/daemonlogger

if [ "x$capdir" = "x" ]; then
    echo "$0: no capture directory" >&2
    exit 5
fi

cd "$capdir" || exit 1

if [ ! -x "$daemonlogger" ]; then
    echo "$0: $daemonlogger not found"
    exit 1
fi

touch $logfile
echo "Starting concap at `date`" >>$logfile
# env >>$logfile

umask "$umask"

/usr/local/sbin/run-daemon -p "$pidfile" -0 "$procname" -w 60 -r 15 \
  $daemonlogger -i "$interface" -u "$user" -g "$group" -M "$maxdisk" -m "$nfiles" -s
"$filesize" -r \
    -l "$capdir" -n "$capfilename" -f "$bpf" \
    >>$logfile 2>&1 </dev/null
```

concap.conf

The concap.conf file is read by concap-start to configure and start the daemonlogger.

```
# concap configuration file
bpf="/usr/local/etc/bpf.conf"

# enable=yes enables this instance of concap.
enable=yes

# Location and instance are used to default other values.
location=routerlocation
instance=all
interface=eth1

# filesize is the size in bytes when we rotate capture file.
# nfiles is the number of capture files before deleting oldest.
# maxdisk is the percent of filesystem used before deleting oldest capture file.
filesize=100000000
nfiles=25000
maxdisk=95

# procname is process name.
procname="concap-$instance"

# capdir is directory for capture files.
```

```
capdir="/data/concap/$instance"

# capfilename is filename prefix for capture files.
capfilename="$location-$instance.pcap"

# user and group determine uid/gid or running process.
user=tcpdump
group=tcpdump

# umask is file creation umask for running process.
umask=027

# chroot directory for running process.
chroot=

# pidfile is pid file for tracking process
pidfile="/var/run/$procname.pid"

# logfile is file to receive program output and error messages.
logfile="/var/log/$procname.log"
```

Kim Cary, kim.cary@pepperdine.edu