

Lab 9 – NIDS/NIPS and Web Proxy Analysis

Sai Sasaank Srivatsa Pallerla

University of Maryland, Baltimore County

Presented to: Gina Scaldaferri

Date: 11/12/2019

About the Lab

Snort network intrusion detection system (IDS) and intrusion prevention system (IPS). It has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort performs protocol analysis, content searching and matching. It can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, semantic URL attacks, buffer overflows, server message block probes, and stealth port scans.

Squid is a caching and forwarding HTTP web proxy. It has a wide variety of uses, including speeding up a web server by caching repeated requests, caching web, DNS and other computer network lookups for a group of people sharing network resources, and aiding security by filtering traffic. Although primarily used for HTTP and FTP, Squid includes limited support for several other protocols including Internet Gopher, SSL, TLS and HTTPS

A hex editor or binary file editor or byte editor is a computer program that allows for manipulation of the fundamental binary data that constitutes a computer file.

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible. It is one of the most famous tools that is used to monitor network traffic and protocols used. It lets us monitor the network at a microscopic level, both wired and wireless.

Part 1. IDS/IPS Log Analysis

1. Examine the alert's data to understand the logistical context

[1:10000648:2] SHELLCODE x86 NOOP [] [Classification: Executable code was detected]
[Priority: 1] 5/18-08:01:45.591840 172.16.16.218:80 -> 192.168.1.169:2493 TCP TTL: 63
TOS:0x0 ID:53309 IpLen: 20 DgmLen: 1127 DF *AP* Seq: 0x1B2C3517 Ack: 0x9F9E0666
Win: 0x1920 TcpLen: 20

SHELLCODE x86 NOOP: This is the message that the user gave to display when an alert pops up.

[Classification: Executable code was detected]: This the category of the alert belongs to, generally this can also be deduced from the alert file the snort rule was written in.

[Priority: 1]: The priority level of the alert, so that the SOC can focus and divide their attention.

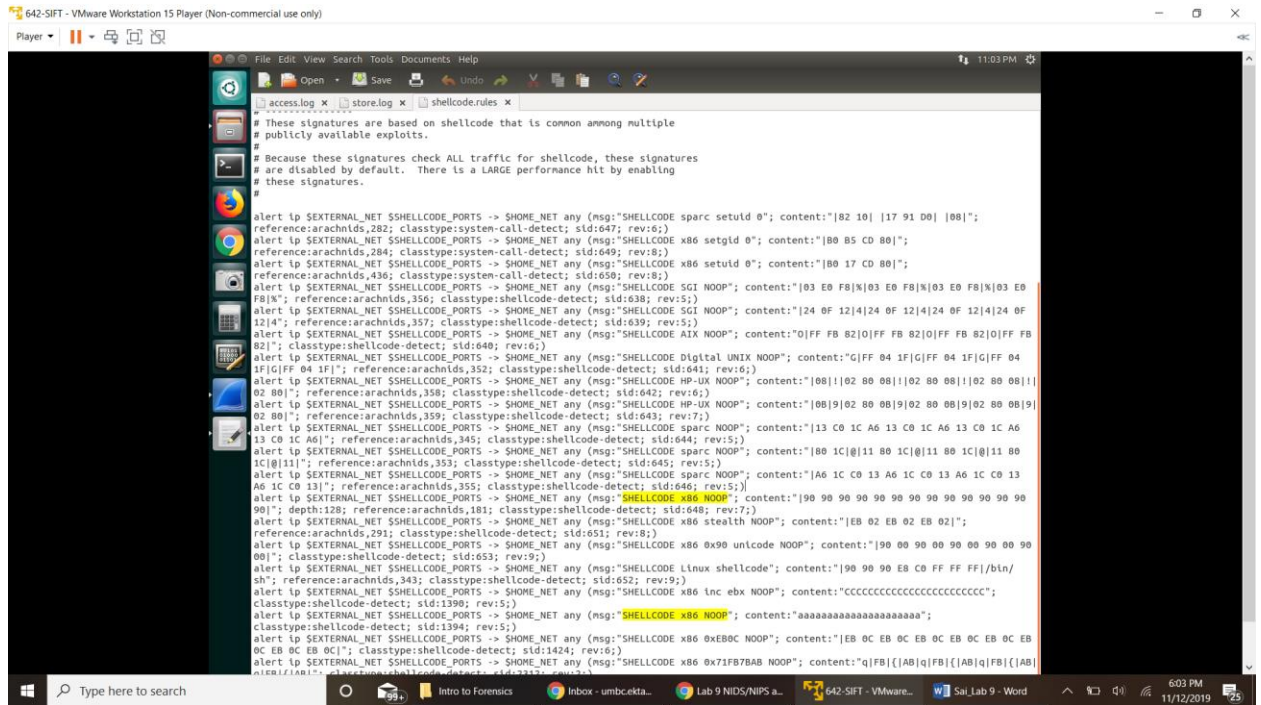
5/18-08:01:45.591840: The time stamp when the alert/ rule was triggered.

172.16.16.218:80 -> 192.168.1.169:2493: From and to IP address of the packet that was detected/ triggered the rule.

TCP TTL: 63 TOS:0x0 ID:53309 IpLen: 20 DgmLen: 1127 DF *AP* Seq: 0x1B2C3517 Ack: 0x9F9E0666 Win: 0x1920 TcpLen: 20: This mentions the details of the packet, these details help the SOC pinpoint the packet and conduct further investigation.

2. Compare the alert to the rule, in order to better understand WHAT it has been built to detect

There are many rules mentioned in the rules folder of snort, to narrow down our search we can simply look at the SHELLCODE rules written the shellcode.rules file.



Two of the relevant alerts were:

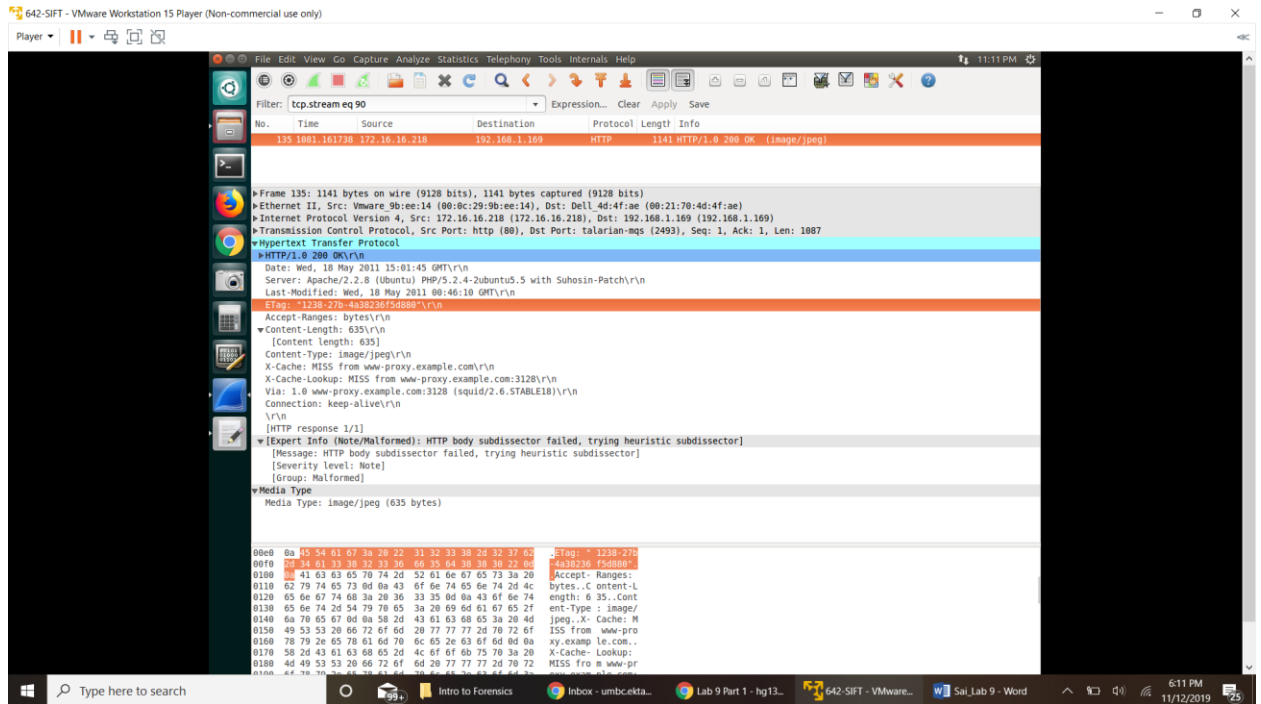
```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
depth:128; reference:arachnids,181; classtype:shellcode-detect; sid:648; rev:7;)
```

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any
(msg:"SHELLCODE x86 NOOP"; content:"aaaaaaaaaaaaaaaaaaaaaa"; classtype:shellcode-
detect; sid:1394; rev:5;)
```

These two rules basically serve the same purpose, When inspecting a packet these rules check of "aaaaaaaaaaaaaaaaaaaaaa" or its hex value "|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|". The intention behind this is when the attacker sends executable code, they try to mask the code by simply XORing the code and the output is usually aaaaaaaaaaaaaaaaaaaaaa or any repeated characters.

3. Retrieve the packet that triggered the alert

To get the packet that triggered the alert we need to inspect the elements of the alert, we can see in field "ID:53309" which is the ID of the IP. So, to get the packet we open the log file in Wireshark and put a filterer "ip.id == 53309" and we get the packet, it's a image/jpeg file.



4. Compare the rule to the packet to understand WHY it fired

Clearly the rule was checking for "90 90 90 90 90 90 90 90 90 90 90 90". The packet contained this int eh hex value that triggered the alert.

```
0400 55 37 f2 a3 b3 c3 28 29 d3 e3 f3 84 94 a4 b4 c4 U7....() .....
0410 d4 e4 f4 65 75 85 95 a5 b5 c5 90 90 90 90 90 90 ...eu... .....
0420 90 90 90 90 90 90 90 90 90 90 57 67 77 87 97 a7 ..... ..Wgw...
0430 b7 c7 d7 e7 f7 38 48 58 68 78 88 98 a8 b8 c8 d8 .....8HX hx.....
0440 e8 f8 39 49 59 69 79 89 99 a9 b9 c9 d9 e9 f9 2a ..9IYiy. ....*
0450 3a 4a 5a 6a 7a 8a 9a aa ba ca da ea fa ff da da .17i7
```

Subsequently, you'll want to determine if there are any other activities that are related to the original event.

1. Construct a timeline of alerted activities involving the potentially malicious outside host
2. Construct a timeline of alerted activities involving the target

To prepare a timeline we need to inspect the IP addresses involved the suspicious packets, which are 172.16.16.218 and 192.168.1.169. The timeline involving internal and external hosts are as follows, (for this I used `grep -r 172.16.16.218` and `grep -r 192.168.1.169`)

```
sansforensics@l5ftworkstation:~/Desktop/cases/Evidence/NIDS Analysis$ grep -r 172.16.16.218
alert: 05/18-08:01:45.591840 172.16.16.218:80 -> 192.168.1.169:2493
sansforensics@l5ftworkstation:~/Desktop/cases/Evidence/NIDS Analysis$ grep -r 192.168.1.169
alert: 05/18-07:45:09.351488 207.46.140.21:80 -> 192.168.1.169:2127
alert: 05/18-07:45:09.834604 207.46.193.178:80 -> 192.168.1.169:2132
alert: 05/18-07:45:51.204813 65.54.95.43:80 -> 192.168.1.169:2142
alert: 05/18-07:46:15.877801 207.46.140.21:80 -> 192.168.1.169:2151
alert: 05/18-07:47:23.571802 208.185.127.35:80 -> 192.168.1.169:2160
alert: 05/18-07:49:10.528302 64.236.85.181:80 -> 192.168.1.169:2194
alert: 05/18-07:49:13.168434 208.185.127.35:80 -> 192.168.1.169:2204
alert: 05/18-07:49:22.868619 66.199.151.142:80 -> 192.168.1.169:2201
alert: 05/18-07:49:41.005265 69.172.216.55:80 -> 192.168.1.169:2208
alert: 05/18-07:50:49.393211 204.203.18.139:80 -> 192.168.1.169:2211
alert: 05/18-07:50:50.999308 64.4.21.39:80 -> 192.168.1.169:2215
alert: 05/18-07:50:51.148406 204.203.18.139:80 -> 192.168.1.169:2212
alert: 05/18-07:51:49.929617 204.203.18.147:80 -> 192.168.1.169:2238
alert: 05/18-07:51:49.935588 204.203.18.147:80 -> 192.168.1.169:2239
alert: 05/18-07:51:49.942308 204.203.18.136:80 -> 192.168.1.169:2240
alert: 05/18-07:51:49.960526 204.203.18.147:80 -> 192.168.1.169:2238
alert: 05/18-07:51:50.660559 66.199.151.142:80 -> 192.168.1.169:2248
alert: 05/18-07:51:50.808069 66.199.151.142:80 -> 192.168.1.169:2248
alert: 05/18-07:51:51.651178 208.71.198.133:80 -> 192.168.1.169:2251
alert: 05/18-07:51:51.736993 207.46.148.35:80 -> 192.168.1.169:2253
alert: 05/18-07:51:51.769689 64.94.107.20:80 -> 192.168.1.169:2254
alert: 05/18-07:51:51.880458 208.71.198.133:80 -> 192.168.1.169:2256
alert: 05/18-07:51:51.897291 208.71.198.133:80 -> 192.168.1.169:2257
alert: 05/18-07:52:08.448729 204.203.18.154:80 -> 192.168.1.169:2260
alert: 05/18-07:52:10.768211 66.114.50.47:80 -> 192.168.1.169:2265
alert: 05/18-07:52:10.944676 66.114.50.47:80 -> 192.168.1.169:2264
alert: 05/18-07:52:32.441844 208.71.198.133:80 -> 192.168.1.169:2251
alert: 05/18-07:52:36.272357 208.71.198.133:80 -> 192.168.1.169:2256
alert: 05/18-07:52:36.893492 207.46.148.35:80 -> 192.168.1.169:2253
alert: 05/18-07:52:36.965692 64.94.107.20:80 -> 192.168.1.169:2281
alert: 05/18-07:52:37.003169 69.194.244.11:80 -> 192.168.1.169:2280
alert: 05/18-07:52:38.338816 98.137.51.254:80 -> 192.168.1.169:2228
alert: 05/18-07:52:38.433211 138.108.5.10:80 -> 192.168.1.169:2283
alert: 05/18-07:52:38.941344 208.71.198.133:80 -> 192.168.1.169:2257
alert: 05/18-07:52:50.108083 74.122.140.121:80 -> 192.168.1.169:2276
alert: 05/18-07:52:50.754182 74.122.140.121:80 -> 192.168.1.169:2276
alert: 05/18-07:52:51.117587 207.171.195.201:80 -> 192.168.1.169:2300
alert: 05/18-07:52:51.394218 184.24.130.77:80 -> 192.168.1.169:2301
alert: 05/18-07:52:51.586201 204.11.109.24:80 -> 192.168.1.169:2302
alert: 05/18-07:53:24.823879 208.71.198.133:80 -> 192.168.1.169:2251
alert: 05/18-07:53:25.656624 184.24.130.77:80 -> 192.168.1.169:2301
alert: 05/18-07:53:25.672738 184.24.130.77:80 -> 192.168.1.169:2301
alert: 05/18-07:53:25.843290 184.24.130.77:80 -> 192.168.1.169:2301
alert: 05/18-07:53:26.118069 207.46.148.35:80 -> 192.168.1.169:2312
alert: 05/18-07:53:26.176160 204.11.109.24:80 -> 192.168.1.169:2314
alert: 05/18-07:53:26.205129 64.94.107.20:80 -> 192.168.1.169:2316
alert: 05/18-07:53:26.239849 69.194.244.11:80 -> 192.168.1.169:2315
alert: 05/18-07:53:26.376435 208.71.198.133:80 -> 192.168.1.169:2256
```

07:45:09 - NIDS alerts for 192.168.1.169 begin.

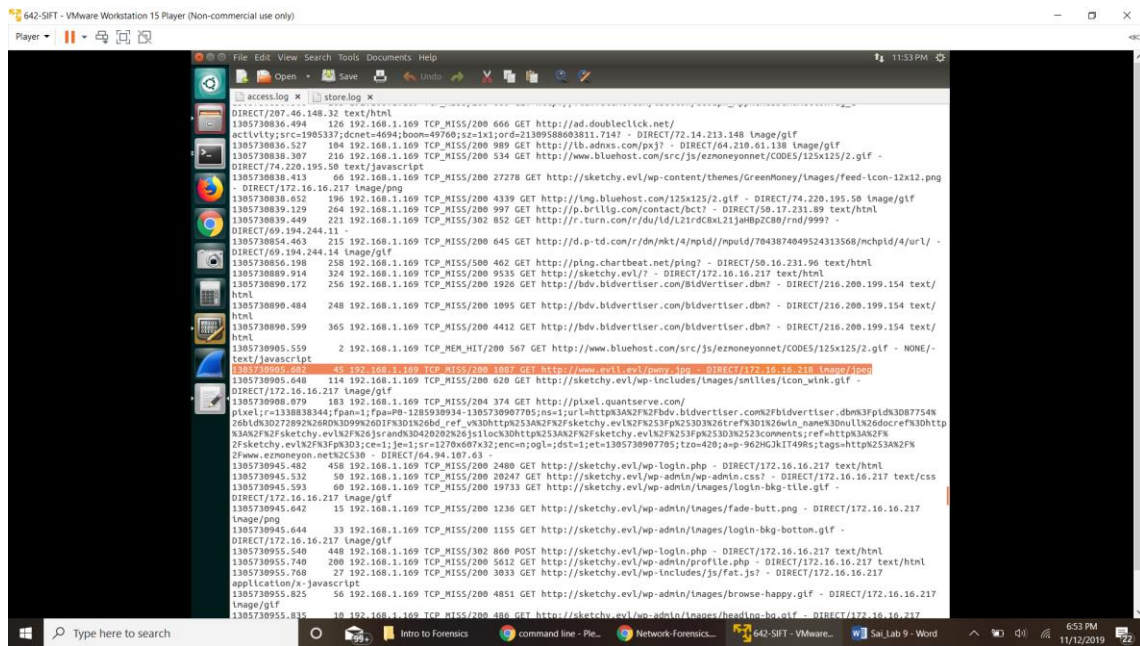
08:01:45 - NIDS alert for possible shellcode being downloaded

08:04:28-08:04:38 - Multiple NIDS alerts

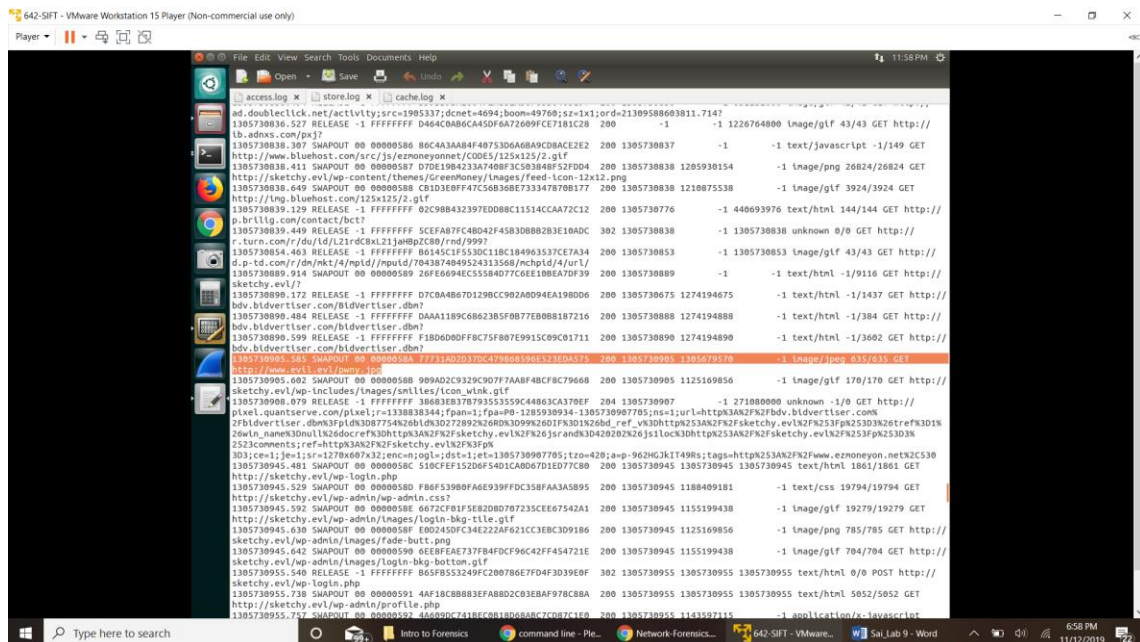
Part 2. Proxy Log Analysis

1. Determine whether the evidence extracted from the Squid cache corroborates our findings from the Snort logs.

Yes, the evidence extracted from Squid cache corroborated our finding. First we search for the IP address 172.16.16.218 in access.log

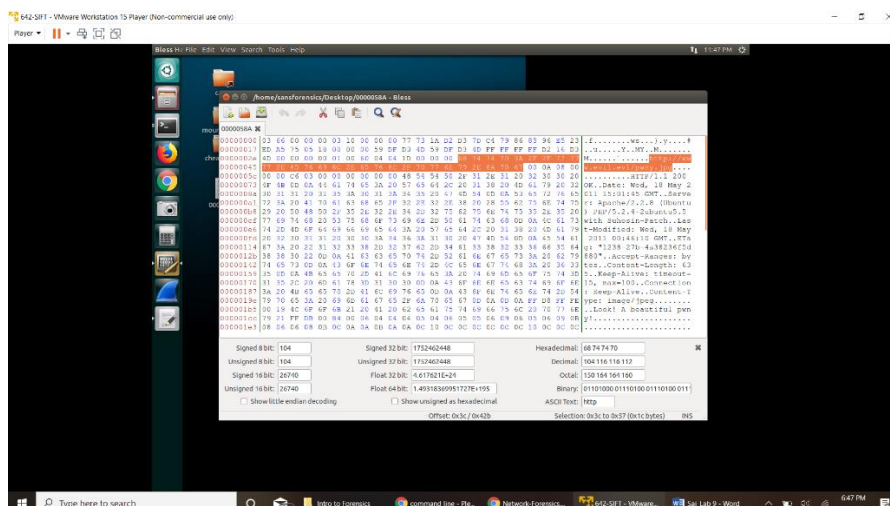
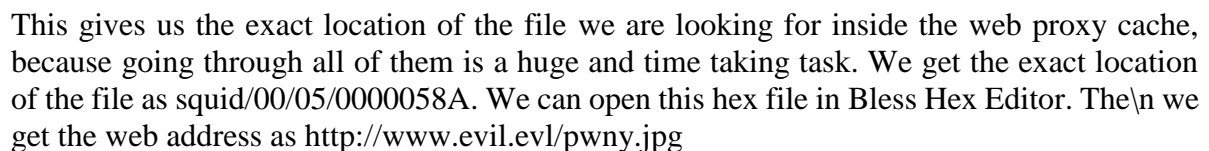


Then we get the website as <http://www.evil.evl/pwny.jpg> then we search for pwny.jpg in store.log



1. Present any information you can find regarding the identity of any internal users who have been engaged in suspicious activities.

We use `grep -r Etag-value`, the value we got from inspecting the packet.



The screenshot shows a VMware Workstation 15 Player window titled "642-SIFT - VMware Workstation 15 Player (Non-commercial use only)". Inside the VM, a Linux desktop environment is visible. A web browser window displays a page titled "Welcome to sKetch!". The page content includes a "Linux affiliate banner" with links to "Home", "About", and "Credit Card Number Recycling". Below this is a "Welcome to sKetch!" heading, a date "February 26th, 2009 | Posted in General", and a message: "Welcome to sKetchy Kredit, your #1 source of perfectly legitimate credit cards". There is a "1 Comment" section with a comment from "l0ser" dated "April 29th, 2011 at 2:28 am" which says "luv the site! hope u get lots of traffic lol.". A "Leave a Comment" form is at the bottom with fields for Name, Mail, and Website, and a "Submit Comment" button.

Overlaid on the browser window is a code editor window titled "pwny.html (-/Desktop) - gedit". The editor shows HTML code for a comment form. A red box highlights a section of the code:


```

    <div class="commenttext">
      <p>luv the site! 
  
```

 The rest of the visible code in the editor is:


```

    <li class="alt1"
    id="comment-3" >

    <div class="commentcount">
      <a href="#comment-3" title="">1</a>
    </div>

    <strong>l0ser</strong> // April 29th, 2011 at
    <br />

    </li>

    </ol>
  
```

 The status bar at the bottom of the code editor shows "HTML", "Tab Width: 8", "Ln 92, Col 25", and "RHS".

Conclusion

From this lab we learn how to read through snort alerts and go through web proxy to find relevant data from the hex dumps. In a way we also learn how to write snort rules. We learn how view the logs to make inferences, match timelines to proves hypothesis. In short, this lab tells us that logs are important to look back at how the incident happened.

References

- [1] <https://forum.netgate.com/topic/138615/snort-alert-log-format/7>
- [2] <https://github.com/Graylog2/graylog-guide-snort>
- [3] <https://resources.infosecinstitute.com/snort-rules-workshop-part-one/#gref>