CSE 40567 / 60567: Computer Security

Network Security 5
Homework #7 Released
Due: Tonight at 11:59PM Eastern Time

See Assignments Page on the course website for details
Distributed Denial of Service Attacks
Botnets

Carna Botnet March–December 2012  BY-SA 4.0 Cyp
Distributed Denial of Service Attacks

How a botnet works

1. Malware infects computers and turns them into bots.
2. The bots are then controlled by a central command center.
3. The command center sends out commands to the bots.
4. The bots carry out the commands and send back data.

How a botnet works (© BY-SA 3.0 Uploaded by Tom-b~commonswiki)
What does DDoS traffic look like?

General strategy: blast target with as many packets as possible
  ▸ Saturates bandwidth
  ▸ May crash OS

• Flood attacks
• Amplification attacks
• Resource depletion attacks
Flood Attacks

UDP Flood Attack

UDP 10  UDP 11  UDP 12  UDP 65,535

12.3.66.101  128.11.231.7  128.9.21.11  99.41.200.58...

ICMP Flood Attack

ICMP Echo Requests / ICMP Echo Responses

99.41.200.58  128.11.231.7  128.9.21.11  12.3.66.101

Victim

ICMP Dest. Port
Unreachable

Victim
Amplification Attacks

Attacker with 1Mbps

10 compromised triggers

1Gbps connection x10

400 amplifier machines

Amplification factor of 50x

500Gbps from amplifiers hits victim

Resource Depletion Attacks

1. TCP SYN Attack
   - Attacker
   - Web Server
   - Port 80
   - SYN
   - SYN-ACK
   - SYN
   - SYN-ACK
   - ... 
   - Send SYNs until no more connections can be established

2. TCP PSH + ACK Attack
   - Attacker
   - Web Server
   - Port 80
   - PSH + ACK
   - ACK
   - PSH + ACK
   - ACK
   - ... 
   - Send PSH + ACKs until target’s resources are exhausted

Data Unloaded
Port 80
Data Unloaded
Insidious: direct lots of legitimate traffic to site

https://twitter.com/search?q=%22RIP%20Paul%20McCartney%22
Defenses

• Attacks on the decline (?)
  ‣ Reported peak in the early to mid-2000s (Kaspersky Lab)

• Technical countermeasures are now commonplace
  ‣ Firewalls
  ‣ Switches with rate-limiting and ACLs
  ‣ Routers with rate-limiting and ACLs

Botnet DDoS Attacks: Q4 2015

Kaspersky Lab Report

• Resources in 69 countries were targeted by DDoS attacks.
• 94.9% of the targeted resources were located in 10 countries.
• Largest numbers of DDoS attacks targeted victims in China, the US and South Korea.
• Longest DDoS attack lasted for 371 hours (or 15.5 days).
• SYN DDoS, TCP DDoS and HTTP DDoS remain the most common attack scenarios.
• The proportion of DDoS attacks from Linux-based botnets was 54.8%.

Firewalls
Firewalls

- Monitor and control all incoming and outgoing packets
- Firewall behavior is defined by a ruleset
- Two different categories
  - Network-based
  - Host-based
Firewall placement

An illustration of where a firewall would be located in a network. © BY-SA 3.0 Bpedrozo
Network Layer / Packet Filtering

Ruleset applied at these layers

OSI Packet Filter  BY-SA 3.0 Willysaef
Application Layer

Split transaction in two:

To the client, the firewall appears to be the server
To the server, the firewall appears to be the client
Proxies

**Proxy** forwarding requests to and from the Internet

**Reverse proxy** takes requests from Internet and forwards them to internal network servers

Likely unaware of internal servers
Network Address Translation

- Client 10.10.10.25
  - Src: 2.25.191.5
- NAT 128.124.1.5
- Server 1 9.5.19.141
  - Dst: 128.124.1.5
- Server 2 2.25.191.5
  - Dst: 128.124.1.5

NAT categorization according to RFC 3489

© BY-SA 3.0 Christoph Sommer
iptables

• Interface to Linux kernel firewall
  ▪ iptables applies to ipv4
  ▪ ip6tables applies to ipv6
  ▪ arptables applies to arp
  ▪ ebtables to ethernet frames

# iptables -L | grep policy
Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)

https://help.ubuntu.com/community/IptablesHowTo
iptables

Three different chains for rules:

**Input:** controls the behavior of incoming connections

**Forward:** used for incoming connections that aren’t being delivered locally (routing / NAT)

**Output:** controls the behavior of outgoing connections
iptables: blocking connections

Block all connections from 10.10.10.10:

```
# iptables -A INPUT -s 10.10.10.10 -j DROP
```

Block all connections from a class C network:

```
# iptables -A INPUT -s 10.10.10.0/24 -j DROP
```
iptables: blocking connections

How can we stop an ssh brute force attack as it’s happening?

sshd[22731] Failed password for invalid user admin from 10.10.10.10 port 32981
sshd[22732] Failed password for invalid user admin from 10.10.10.10 port 32989
...

fail2ban-style rule:

```
# iptables -A INPUT -p tcp --dport ssh -s 10.10.10.10 -j DROP
```
iptables: connection states

Allow existing session to receive traffic:

```bash
# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Allow ssh traffic:

```bash
# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```

Check the current state of the rules:

```bash
# iptables -L
Chain INPUT (policy ACCEPT)
  target     prot opt source           destination
  ACCEPT     all  --  anywhere         anywhere
          all  --  anywhere         anywhere
RELATED,ESTABLISHED
  ACCEPT     tcp  --  anywhere         anywhere
          tcp  --  anywhere         anywhere
          tcp  dpt:ssh
```
iptables: connection states

Allow www traffic:

    # iptables -A INPUT -p tcp --dport www -j ACCEPT

By default, block all traffic:

    # iptables -A INPUT -j DROP

Check the current state of the rules:

    # iptables -L
Chain INPUT (policy ACCEPT)
   target     prot opt source           destination
   ACCEPT     all  --  anywhere         anywhere          state RELATED,ESTABLISHED tcp dpt:ssh
   ACCEPT     tcp  --  anywhere         anywhere          tcp dpt:www
   ACCEPT     tcp  --  anywhere         anywhere
   DROP       all  --  anywhere         anywhere
iptables: editing chains

We forgot to leave the loopback device open — let’s fix that:

```
# iptables -I INPUT 1 -i lo -j ACCEPT
```

Check the current state of the rules:

```
# iptables -L -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out     source       destination
 0     0 ACCEPT     all  --  lo     any     anywhere     anywhere
 0     0 ACCEPT     all  --  any    any     anywhere     anywhere
 0     0 ACCEPT     tcp  --  any    any     anywhere     anywhere
tcp dpt:ssh
 0     0 ACCEPT     tcp  --  any    any     anywhere     anywhere
tcp dpt:www
 0     0 DROP       all  --  any    any     anywhere     anywhere
```
Network Intrusion Detection
IDS is a backup security mechanism

• Assumes other defenses (firewalls, hardened hosts) have failed

• Task is to notice an attack as soon as possible
  - Minimize damage
    ▶ Automated systems or human response

Signature- or Anomaly-based
NIDS placement on network

The firewall is one common location to install a network IDS

By definition, all outside traffic must pass through this chokepoint

All Traffic + NIDS
NIDS placement on network
Signature-based IDS

• Look for patterns in the packet headers and payloads

• Rely on pre-defined signatures for known attacks
  ‣ Professionally developed
  ‣ Community developed

• Examples:
  ‣ Bro (Paxson 1998; https://www.bro.org/)
  ‣ Snort (Roesch 1999)
Snort

https://www.snort.org/

Free and Open Source (commercial offerings via Sourcefire)
Snort Architecture

Internet

Packet Decoder

Pre-processors

Detection Engine

Logging and Alerting System

Output Modules

Packet is Dropped

Output Alert or Log to a File
Usage (Unix)

Sniffer Mode

$ snort -vde ← IP/TCP/UDP/ICMP headers + data

Packet Logger Mode

$ snort -dev -l ./packet.log ← write
$ snort -dv -r packet.log ← read

Network Intrusion Detection Mode

$ snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf ← apply signatures to traffic
Snort Signature

alert tcp any any -> 192.168.1.0/24 111 \n(content:"|00 01 86 a5|"; msg:"mountd access");
Snort Rules Collections

Community Rules:
https://www.snort.org/downloads/#rule-downloads

EmergingThreats.net Open rulesets:
https://rules.emergingthreats.net/open/snort-2.9.0/emerging-all.rules
Let’s examine some real-world threats and corresponding rules...
Remote file inclusion flaw in the WordPress blogging software plugin known as Advanced Custom Fields

The vulnerability allows for remote file inclusion and remote code execution via the `export.php` script.

The Advanced Custom Fields plug-in versions 3.5.1 and below are vulnerable.

This exploit only works when the php option `allow_url_include` is set to On (Default Off).

(exploit available in kali linux 2.0)
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET WEB_SPECIFIC_APPS WordPress Plugin Advanced Custom Fields Remote File Inclusion"; flow:established,to_server; content:"/wp-content/plugins/advanced-custom-fields/core/actions/export.php"; nocase; http_uri; fast_pattern:20,20; content:"abspath="; nocase; http_client_body; pcre:"/abspath=\s*([^\:\?\s]+)\s*\//Pi"; classtype:attempted-user; sid:2016148; rev:1;)
Wuerzburg Shellcode

Origin: Nepenthes modular low-interaction honeypot

Image source: http://books.gigatux.nl/mirror/honeypot/final/ch06lev1sec2.html
Wuerzburg Shellcode

0040200c    eb 27    jmp short wuerzbur.00402035
0040200e    90      nop
0040200f    90      nop
00402010    90      nop
00402011    90      nop
00402012    90      nop
00402013    90      nop
00402014    5d      pop ebp
00402015    33c9    xor ecx,ecx
00402017    66:b9 2502 mov cx,225
0040201b    8d75 05 lea esi,dword ptr ss:[ebp+5]
0040201e    8bfe    mov edi,esi
00402020    8a06    mov al,byte ptr ds:[esi]
00402022    3c 99   cmp al,99
00402024    75 05   jnz short wuerzbur.0040202b
00402026    46      inc esi
00402027    8a06    mov al,byte ptr ds:[esi]
00402029    2c 30   sub al,30
0040202b    46      inc esi
0040202c    34 99   xor al,99
0040202e    8807    mov byte ptr ds:[edi],al
00402030    47      inc edi
00402031    ^e2 ed loopd short wuerzbur.00402020
00402033    eb 0a   jmp short wuerzbur.0040203f
00402035    e8 dafffffff call wuerzbur.00402014
Wuerzburg Shellcode

alert tcp any any -> any any (msg:"ET SHELLCODE Wuerzburg Shellcode"; flow:established; content:"| eb 27|"; content:"| 5d 33 c9 66 b9|"; distance:0; content:"| 8d 75 05 8b fe 8a 06 3c|"; distance:0; content:"| 75 05 46 8a 06|"; distance:0; content:"| 88 07 47 e2 ed eb 0a e8 da ff ff ff|"; distance:0; reference:url,doc.emergingthreats.net/2009251; classtype:shellcode-detect; sid:2009251; rev:3;)

NOP detection (ssh exploit)

#alert tcp $EXTERNAL_NET any -> $HOME_NET 22
(msg:"GPL SHELLCODE ssh CRC32 overflow NOOP"); flow:to_server,established;
content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 |"; reference:bugtraq,2347;

164
Metasploit Meterpreter Kill Process

**Meterpreter**: advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime

Provides a basic shell and allows new features to be added as necessary

Image credit: http://hardsec.net/wp-content/uploads/2014/01/sniffer2.png
Metasploit Meterpreter Kill Process

Ruby code:

```ruby
def Process.kill(*args)
  request =
    Packet.create_request('stdapi_sys_process_kill')

  args.each { |id|
    request.add_tlv(TLV_TYPE_PID, id)
  }

  client.send_request(request)

  return true
end
```
Metasploit Meterpreter Kill Process


167
nmap XMAS scan

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL SCAN nmap XMAS"; flow:stateless; flags:FPU,12; reference:arachnids,30; classtype:attempted-recon; sid:2101228; rev:8;)

nmap SYN scan

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET SCAN NMAP -sS window 1024"; fragbits:!D; dsize:0; flags:S,12; ack:0; window:1024; threshold: type both, track by_dst, count 1, seconds 60; reference:url,doc.emergingthreats.net/2009582; classtype:attempted-recon; sid:2009582; rev:3;)