University of Rome Tor Vergata
ICT and Internet Engineering

Network and System Defense

Alessandro Pellegrini, Angelo Tulumello

A.A. 2023/2024

Denial of Service Attacks

from ***“Computer Security: Principles and Practice” (Chapter 7)***
William Stallings and Lawrie Brown

Denial of Service (DoS) Attack

The NIST Computer Security Incident Handling Guide defines a DoS attack as:

“An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.”

Denial of Service (DoS) Attack

- A form of attack on the availability of some service
- Categories of resources that could be attacked are:

Network bandwidth

Relates to the capacity of the network links connecting a server to the Internet

For most organizations this is their connection to their Internet Service Provider (ISP)

System resources

Aims to overload or crash the network handling software

Application resources

Typically involves a number of valid requests, each of which consumes significant resources, thus limiting the ability of the server to respond to requests from other users

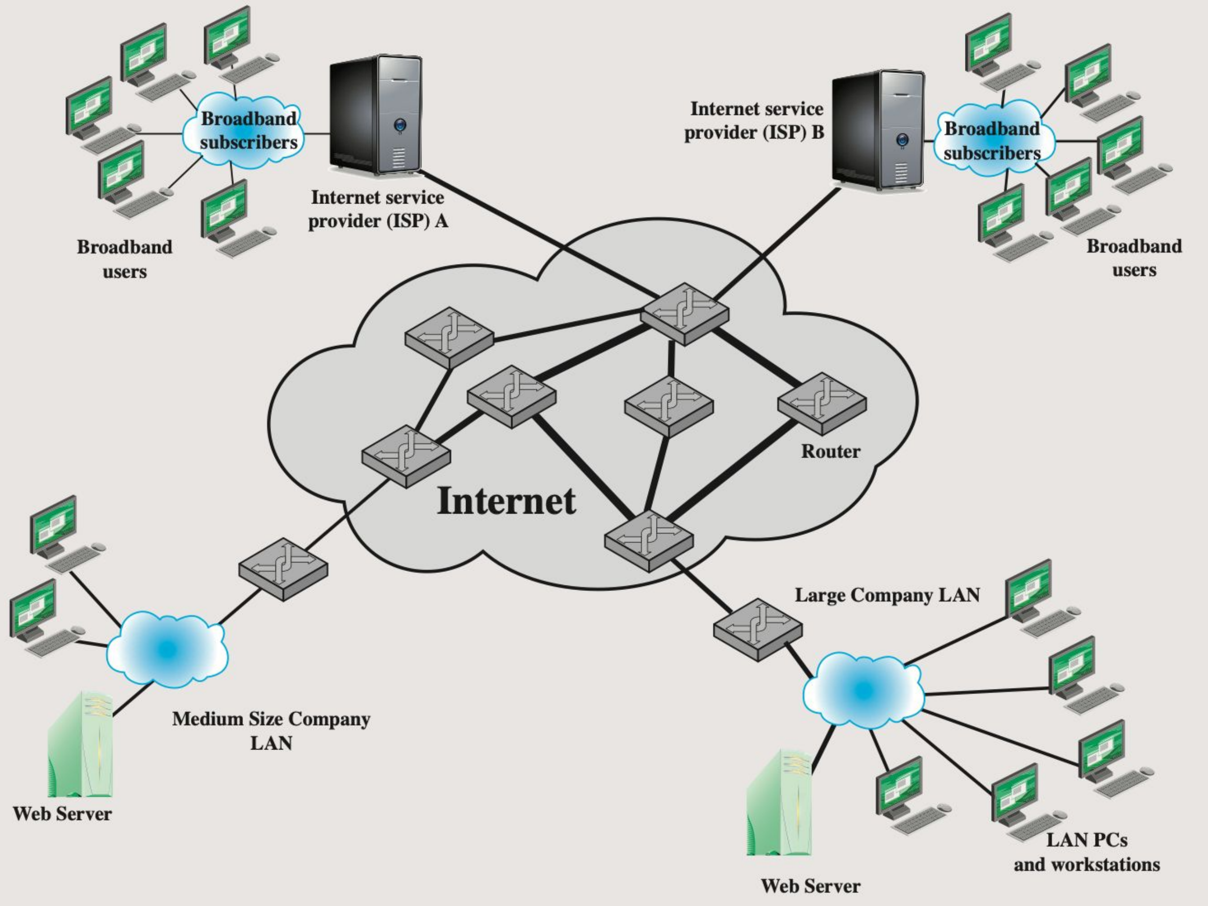


Figure 7.1 Example Network to Illustrate DoS Attacks

Classic DoS Attacks

- Classic DoS attack on network resources
  - Aim of this attack is to overwhelm the capacity of the network connection to the target organization
  - Traffic can be handled by higher capacity links on the path, but packets are discarded as capacity decreases
  - Source of the attack is clearly identified unless a spoofed address is used
  - Network performance is noticeably affected

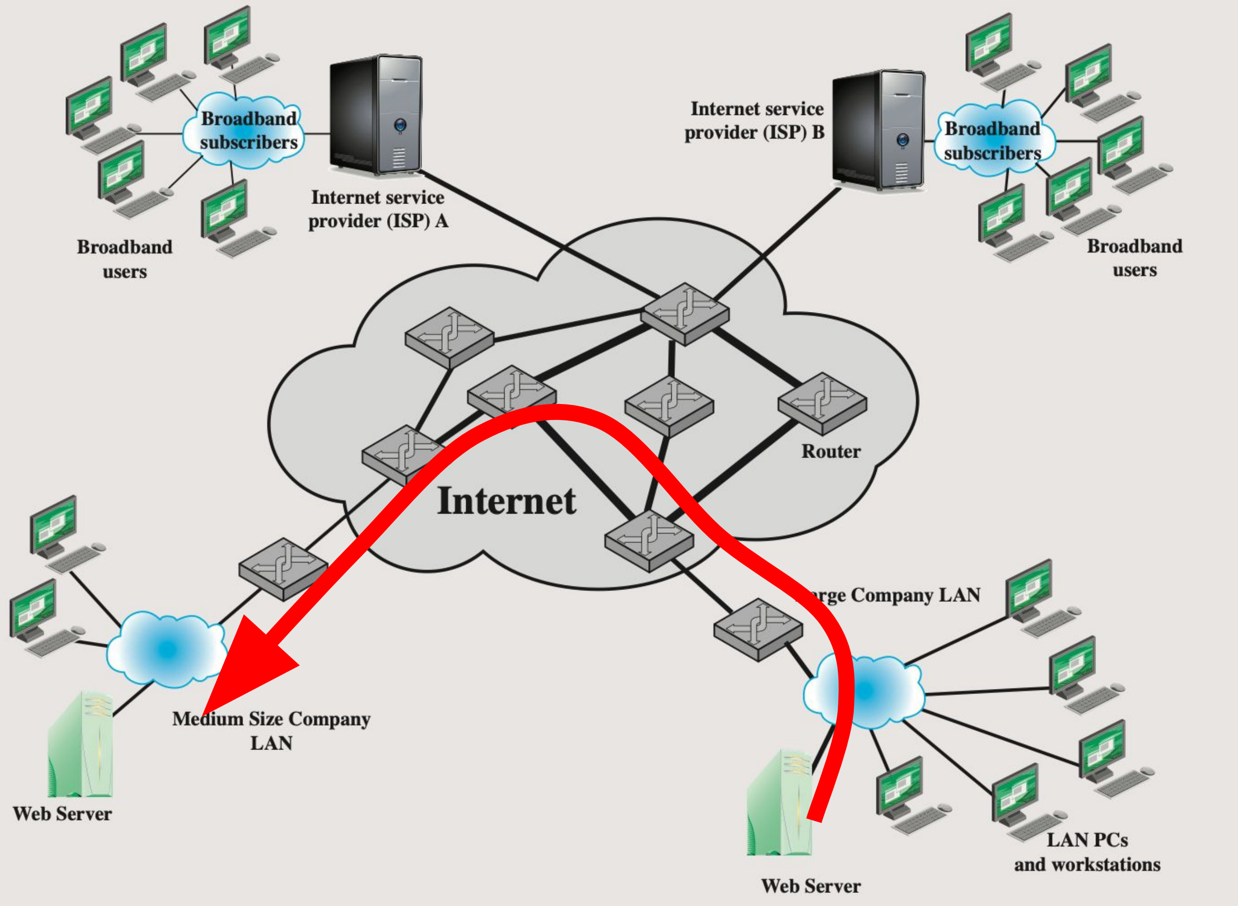


Figure 7.1 Example Network to Illustrate DoS Attacks

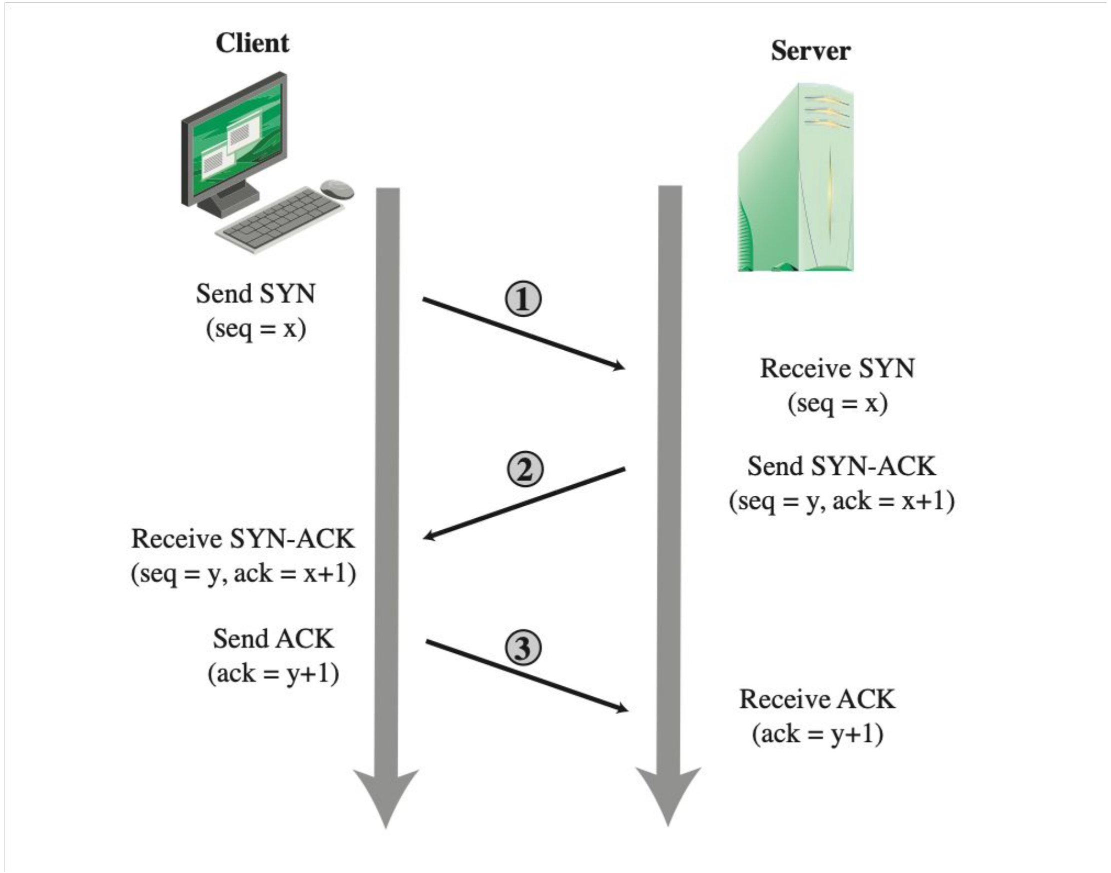
Source Address Spoofing

- ❑ Use forged source addresses
  - ❑ Usually via the raw socket interface on operating systems
  - ❑ Makes attacking systems harder to identify
- ❑ Attacker generates large volumes of packets that have the target system as the destination address
- ❑ Congestion would result in the router connected to the final, lower capacity link
- ❑ Requires network engineers to specifically query flow information from their routers
- ❑ ISPs are supposed to filter spoofed IP addresses belonging to other ISPs

SYN Spoofing

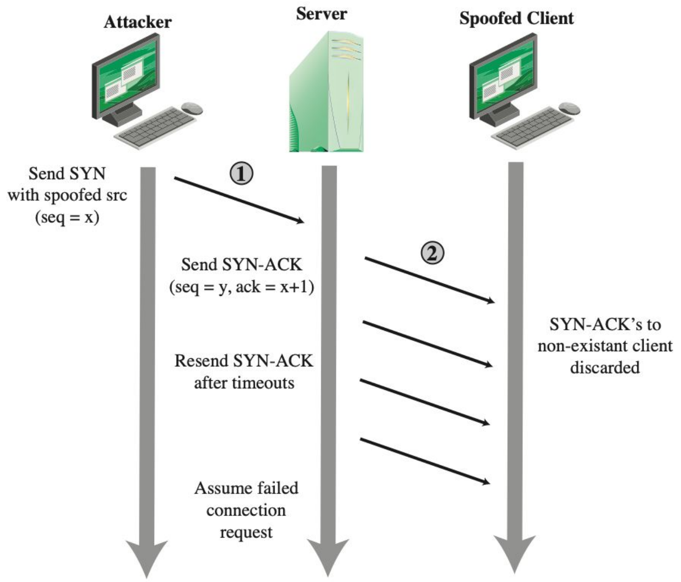
- ❑ Common DoS attack (on system resources)
- ❑ Attacks the ability of a server to respond to future connection requests by overflowing the tables used to manage them
- ❑ Thus legitimate users are denied access to the server
- ❑ Hence an attack on system resources, specifically the network handling code in the operating system

TCP 3-way handshake



SYN Spoofing

- Attacker generates a number of SYN connection request packets with forged source addresses
- For each of these the server records the details of the TCP connection request and sends the SYN-ACK
  - Valid SRC: TCP RST
  - Non-existing SRC: re-TX and wait
- The attacker directs a very large number of forged connection requests at the targeted server.
- These rapidly fill the table of known TCP connections on the server
- Once this table is full, any future requests, including legitimate requests from other users, are rejected
- Significant difference in the volume of network traffic between a SYN attack and ICMP flooding


Attacker
Server
Spoofed Client
Send SYN with spoofed src (seq = x)
①
Send SYN-ACK (seq = y, ack = x+1)
②
Resend SYN-ACK after timeouts
SYN-ACK's to non-existant client discarded
Assume failed connection request

HTTP attacks

- **Flooding**
  - Attack that bombards Web servers with HTTP requests
  - Consumes considerable resources
  - Spidering
    - Bots starting from a given HTTP link and following all links on the provided Web
- **Slowloris**
  - Attempts to monopolize by sending HTTP requests that never complete
  - Eventually consumes Web server’s connection capacity
  - Utilizes legitimate HTTP traffic
  - Existing intrusion detection and prevention solutions that rely on signatures to detect attacks will generally not recognize Slowloris

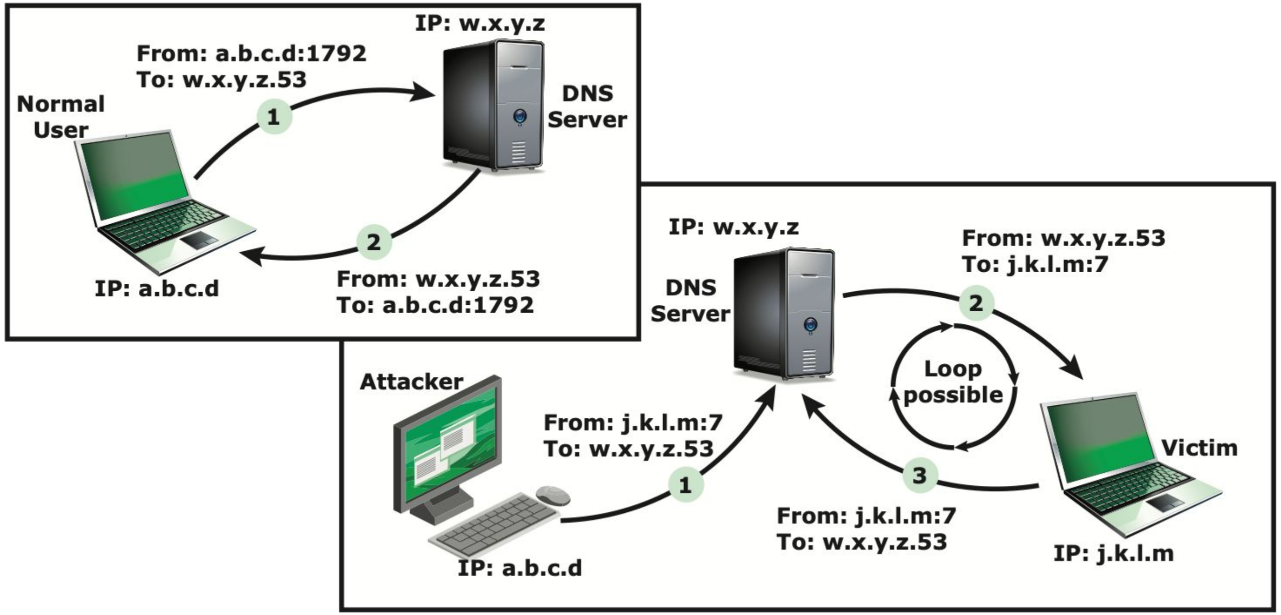
Recap

- Flooding attacks
  - Classified based on network protocol used
  - Intent is to overload the network capacity on some link to a server or its resources
  - Virtually any type of network packet can be used
- Limitations
  - Low traffic volume from a single attacker
  - Esiar to detect and thus mitigate (not so easy, but doable...)
- **Enhancements**
  - *Reflector attacks*
  - *Amplifier attacks*
  - *Distributed denial-of-service attacks*
- Combinations are possible: e.g. DDoS attacks based on Amplification
- Harder to detect (and mitigate) because of distribution and redirection
- No particular resource/bandwidth requirement

Reflection and Amplification Attacks

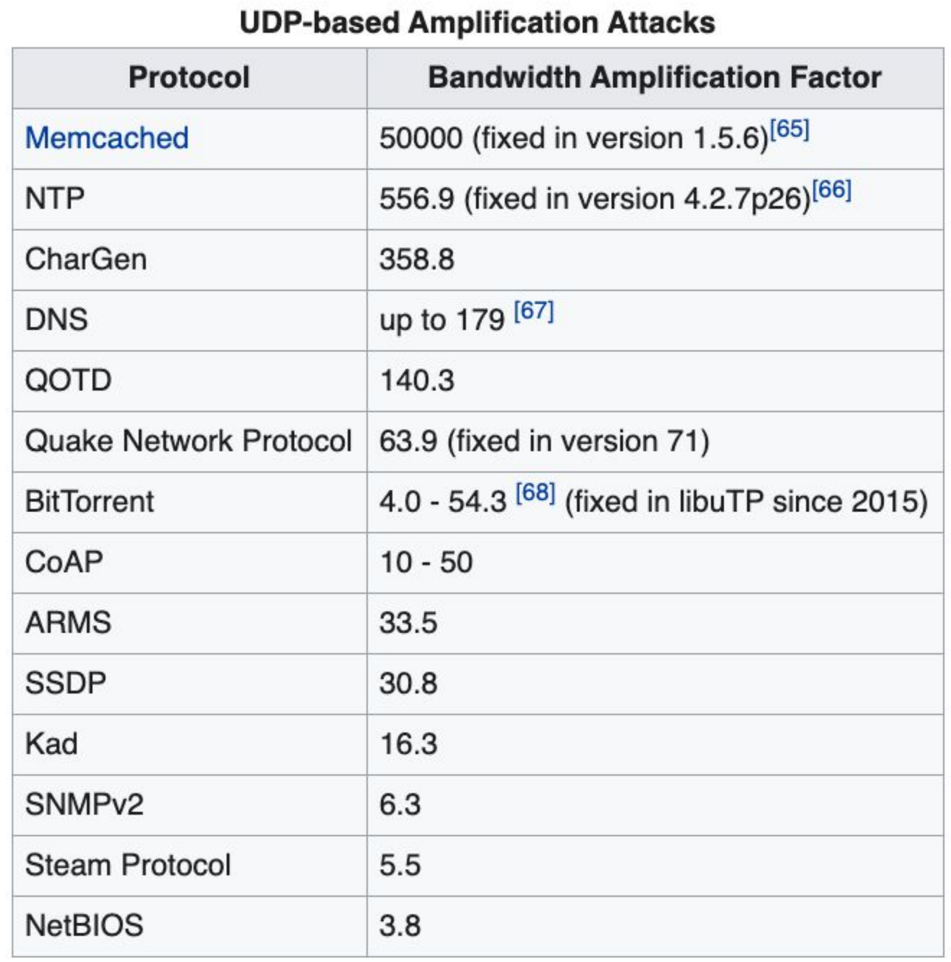
- Attacker sends packets to a known service on the intermediary with a spoofed source address of the actual target system
  - reflector (and amplifier) attacks use network systems functioning *normally*
  - *but need source address spoofing*
- When intermediary responds, the response is sent to the target: **Reflection**
- “Reflects” the attack off the intermediary (reflector)
- Ideally the attacker would like to use a service that created a larger response packet than the original request: **Amplification**
- Goal is to generate enough volumes of packets to flood the link to the target system without alerting the intermediary
- The basic defense against these attacks is blocking spoofed-source packets

DNS Reflection Attack



Amplification Attack Example

- Use packets directed at a legitimate server as the intermediary system
- Attacker creates a series of requests containing the spoofed source address of the target system
- Convert a small request to a much larger response (amplification)
  - the attacker need only generate a moderate flow of packets to cause a larger amplified flow
- Target is flooded with responses
- There are many amplification techniques
- Example: GitHub attack (1.3 Tbps - 130 Mpps, Feb 2018)
  - memcached DDoS attack (so there were no botnets involved)
  - the attackers leveraged the amplification effect of a popular database caching system known as memcached
  - ***50,000x amplification!***

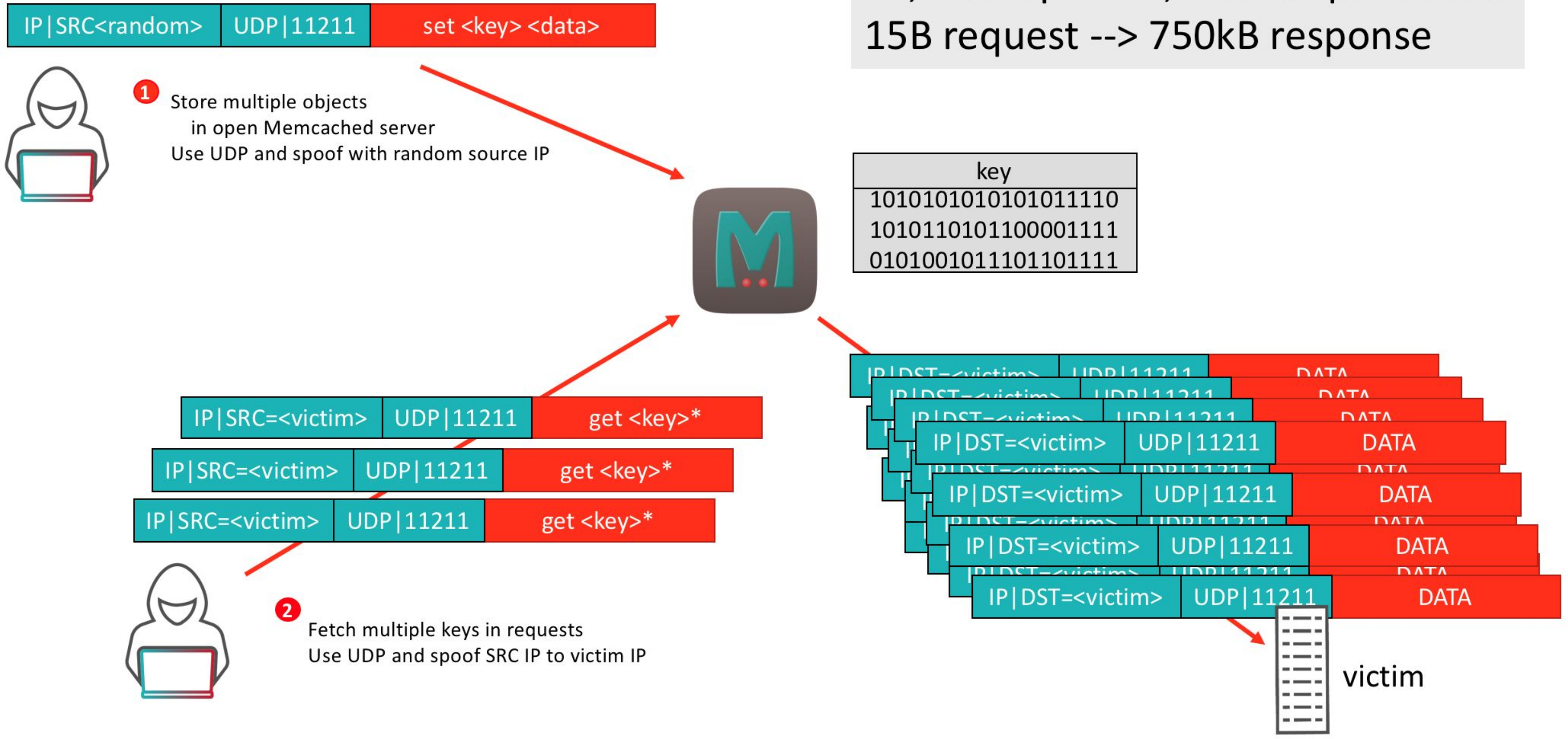


UDP-based Amplification Attacks

Protocol	Bandwidth Amplification Factor
Memcached	50000 (fixed in version 1.5.6)[65]
NTP	556.9 (fixed in version 4.2.7p26)[66]
CharGen	358.8
DNS	up to 179 [67]
QOTD	140.3
Quake Network Protocol	63.9 (fixed in version 71)
BitTorrent	4.0 - 54.3 [68] (fixed in libuTP since 2015)
CoAP	10 - 50
ARMS	33.5
SSDP	30.8
Kad	16.3
SNMPv2	6.3
Steam Protocol	5.5
NetBIOS	3.8

memcached attack

10,000x up to 52,000x amplification
15B request --> 750kB response

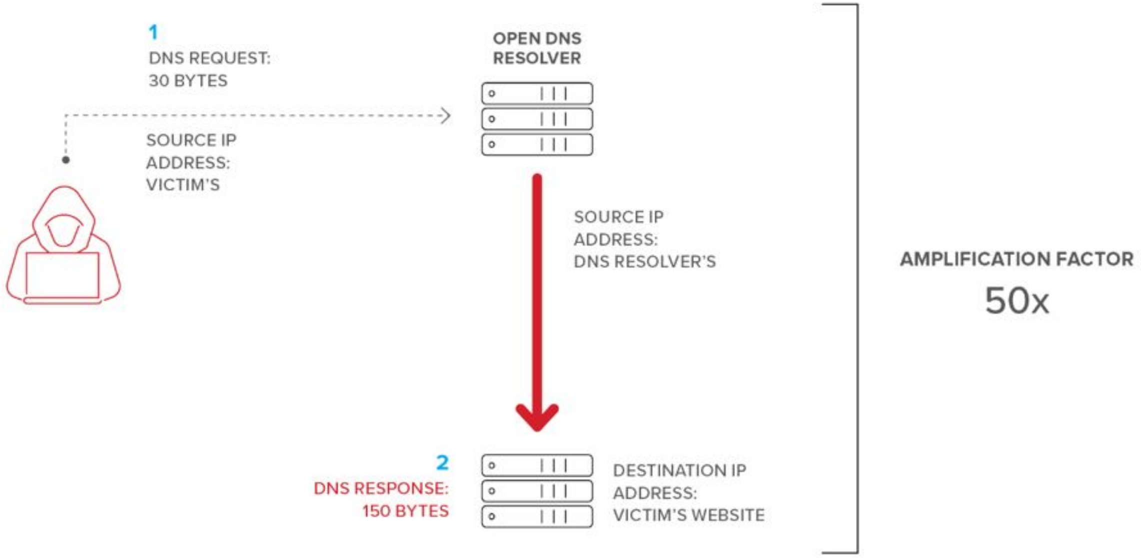


DNS amplification

Attackers craft DNS requests in a way that substantially amplifies the size of the response.

One way to do this is by requesting not just the IP address but the entire domain (for example, using DNS requests for the record type “ANY”), so the response might include details about subdomains, backup servers, mail servers, aliases, and more.

Suddenly, a 10-byte DNS request could generate a response that’s 10, 20, even 50 times larger.



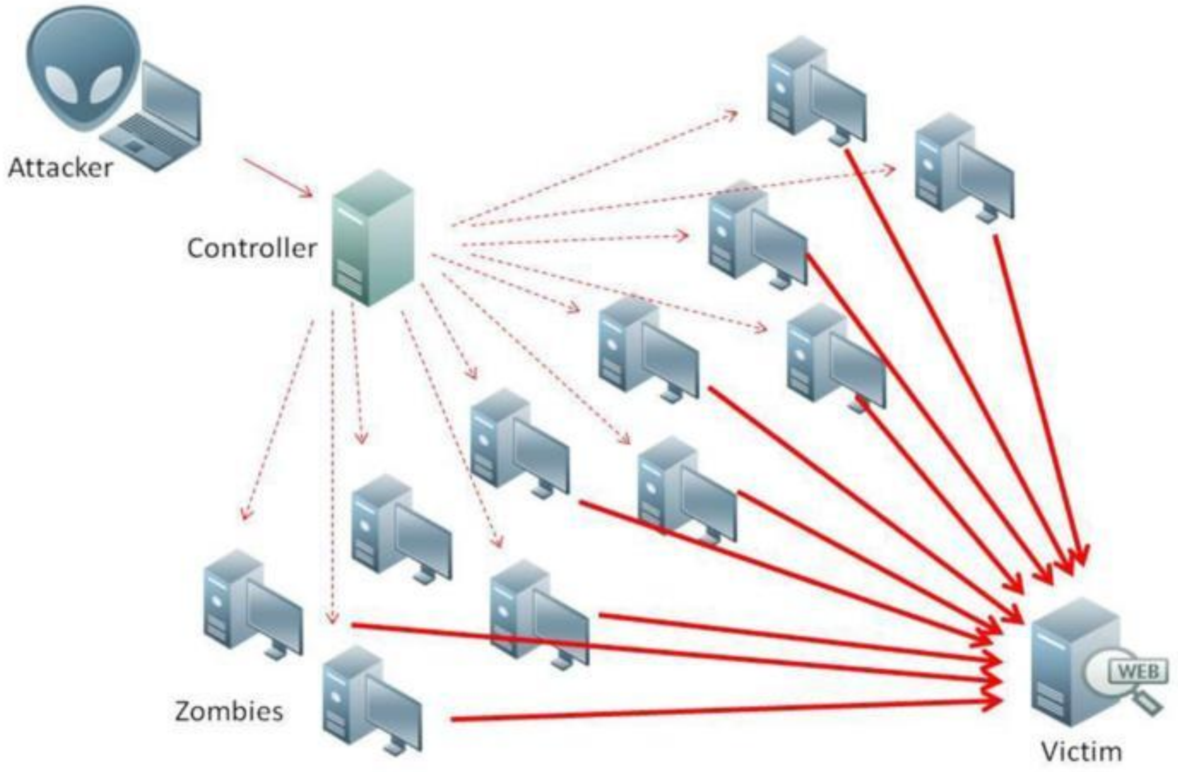
Distributed Denial of Service (DDoS) Attacks

Use of multiple systems to generate attacks

Attacker uses a flaw in operating system or in a common application to gain access and installs their program on it (zombie)

Large collections of such systems under the control of one attacker's control can be created, forming a botnet

DDoS Attack Architecture



DDoS attack mitigation

- These attacks cannot be prevented entirely…
- And in fact also big providers are victim of such attacks
  - 2.3 Tbps attack to Amazon AWS cloud in Feb 2020!!!!
  - High traffic volumes may be legitimate
  - High publicity about a specific site
  - Activity on a very popular site
  - Described as slashdotted, flash crowd, or flash event
  - in general, moder attacks are based on “normal activities” and hard to identify
- Four lines of defense against DDoS attacks
  - ***Attack prevention and preemption*** (before the attack)
  - ***Attack detection and filtering*** (during the attack)
  - ***Attack source traceback and identification*** (during and after the attack)
  - ***Attack reaction*** (after the attack)

DDoS Attack Prevention

- Block spoofed source addresses
  - On routers as **close to source** as possible
- **Filters** may be used to ensure path back to the claimed source address is the one being used by the current packet
  - Filters must be applied to traffic **before** it leaves the ISP’s network or at the point of entry to their network
- Use modified TCP connection handling code (TCP SYN cookies)
  - Cryptographically encode critical information in a cookie that is sent as the server’s initial sequence number
    - Legitimate client responds with an ACK packet containing the incremental sequence number cookie
  - Drop an entry for an incomplete connection from the TCP connections table when it overflows

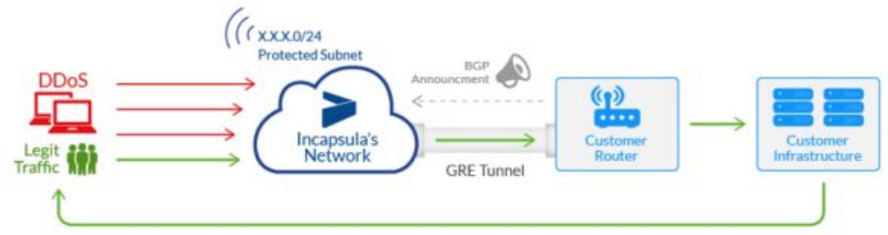
DDoS Attack Prevention

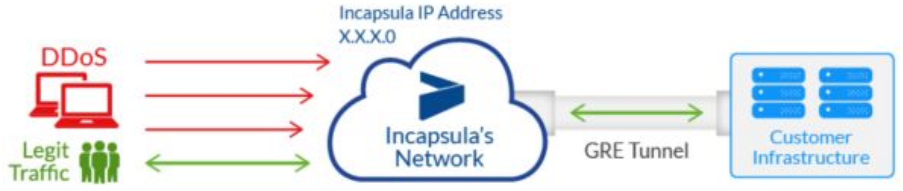
- ❏ Block suspicious services and combinations
- ❏ Manage application attacks with a form of graphical puzzle (captcha) to distinguish legitimate human requests
- ❏ Good general system security practices
- ❏ Use mirrored and replicated servers when high-performance and reliability is required
- ❏ Go to the cloud :)

DDoS Attack Detection and Filtering

- ❑ HW/SW solutions try to detect and mitigate DDoS attack directly in the victim network
- ❑ Even when you don't use third party cloud base hosting, it is better to leave this job to experts :)
  - ❑ ***Cloud Based DDoS mitigation***

- ❑ ***Single IP protection***
  - ❑ Cloud Provider give one IP (from its own pool) to customer
  - ❑ WEB Client traffic is naturally routed to the scrub center and then tunneled to the customer
  - ❑ Responses directly to clients
- ❑ ***Full subnet protection***
  - ❑ customers subnet announced by cloud provider
  - ❑ customer router with BGP/GRE
  - ❑ customer mush have another public IP outside the "protected subnet" for GRE tunneling
  - ❑ customer speaks BGP with cloud provider to dynamically enable cloud based protection





Additional readings

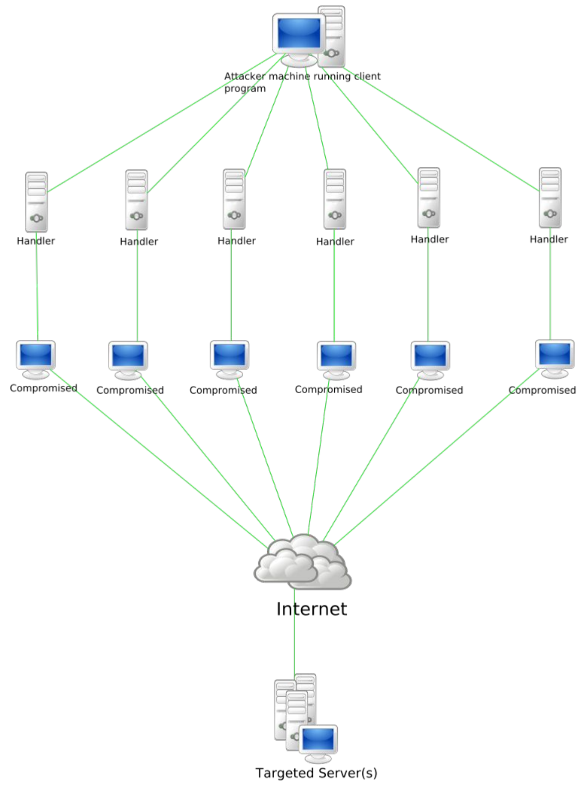
Nice set of shorts (not so hard to understand) articles from Cloudflare

https://www.cloudflare.com/it-it/learning/ddos/what-is-a-ddos-attack/

Botnets

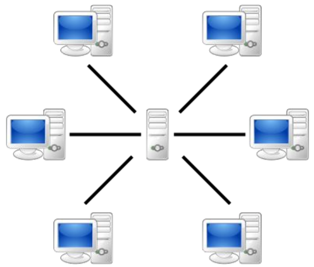
Definition

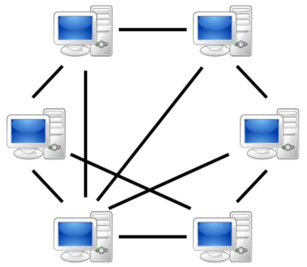
- A **botnet** is a number of Internet-connected devices, each of which is running one or more bots
- Botnets can be used to perform Distributed Denial-of-Service (DDoS) attacks
  - but also steal data, send spam, phishing, and allow the attacker to access the device and its connection
- The owner can control the botnet using command and control (C&C) software



Architecture Overview

- ❑ Botnet architecture has evolved over time in an effort to evade detection and disruption
- ❑ Traditionally, bot programs are constructed as **clients** which communicate via existing **servers**
- ❑ This allows the **bot herder** (the person controlling the botnet) to perform all control from a remote location, which obfuscates the traffic
- ❑ Many recent botnets now rely on existing peer-to-peer networks to communicate
- ❑ These P2P bot programs perform the same actions as the client-server model, but they do not require a central server to communicate


client-server


p2p

Client-Server Model

- First Botnets = client-server model
- Operate through Internet Relay Chat (**IRC**) networks, domains, or websites
  - Infected clients access a predetermined location and await incoming commands from the server.
  - The bot herder sends commands to the server, which relays them to the clients.
  - Clients execute the commands and report their results back to the bot herder.
  - In the case of IRC botnets, infected clients connect to an infected IRC server and join a channel pre-designated for C&C by the bot herder.
  - The bot herder sends commands to the channel via the IRC server. Each client retrieves the commands and executes them.
  - Clients send messages back to the IRC channel with the results of their actions.

P2P Model

- Bot herders have begun deploying malware on peer-to-peer networks.
- These bots may use digital signatures so that only someone with access to the private key can control the botnet
- Newer botnets fully operate over P2P networks.
  - Rather than communicate with a centralized server, P2P bots perform as **both** a command distribution **server** and a **client** which receives commands
  - This avoids having any single point of failure, which is an issue for centralized botnets.
- In order to find other infected machines, the bot discreetly probes random IP addresses until it contacts another infected machine.
  - The contacted bot replies with information such as its software version and list of known bots.
  - If one of the bots' version is lower than the other, they will initiate a file transfer to update
  - This way, each bot grows its list of infected machines and updates itself by periodically communicating to all known bots
- Alternative approach: distributed dictionaries like DHTs

Core components

- A botnet's **originator** (known as a *"bot herder"* or *"bot master"*) controls the botnet remotely
  - This is known as the command-and-control (C&C)
- The program for the operation must communicate via a covert channel to the client on the victim's machine (zombie computer)

Control protocols

- IRC is a historically favored means of C&C because of its communication protocol.
- A bot herder creates an IRC channel for infected clients to join. Messages sent to the channel are broadcast to all channel members.
- The bot herder may set the channel's topic to command the botnet
  - `:herder!herder@example.com TOPIC #channel DDoS www.victim.com`
    - bot herder alerts all infected clients belonging to #channel to begin a DDoS attack on the website www.victim.com
  - `:bot1!bot1@compromised.net PRIVMSG #channel I am DDoSing www.victim.com`
    - client alerts the bot herder that it has begun the attack
- Other example: telnet (or ssh) control protocol
  - remember we can run commands with ssh? :)
  - but many others: web pages, DNS, even twitter !!!
  - more complicated techniques: Fast Flux (more details later...)

Zombie computer

- Computer connected to the Internet that has been **compromised** by a hacker, computer virus or trojan horse and can be used to perform malicious tasks of one sort or another under remote direction
- Most owners of zombie computers are unaware that their system is being used in this way
- A coordinated DDoS attack by multiple botnet machines also resembles a zombie horde attack
- The process of stealing computing resources as a result of a system being joined to a botnet is sometimes referred to as "*scrumping*"

Botnet detection (very very high level)

- Detect the malicious software (*system level*)
    - This is usually done by local antivirus
    - <u>System defense part of this course</u>
- Detect unusual network activities (*network level*)
    - Example:
        - monitor fast DNS record fluctuation
        - monitor the content of packets (not really applicable nowadays)
        - stateful monitoring of traffic pattern (even of encrypted traffic)
    - This can be done everywhere
        - ISP network
        - Data center
        - Edge network
        - Computer
    - Monitoring and Intrusion Detection Systems

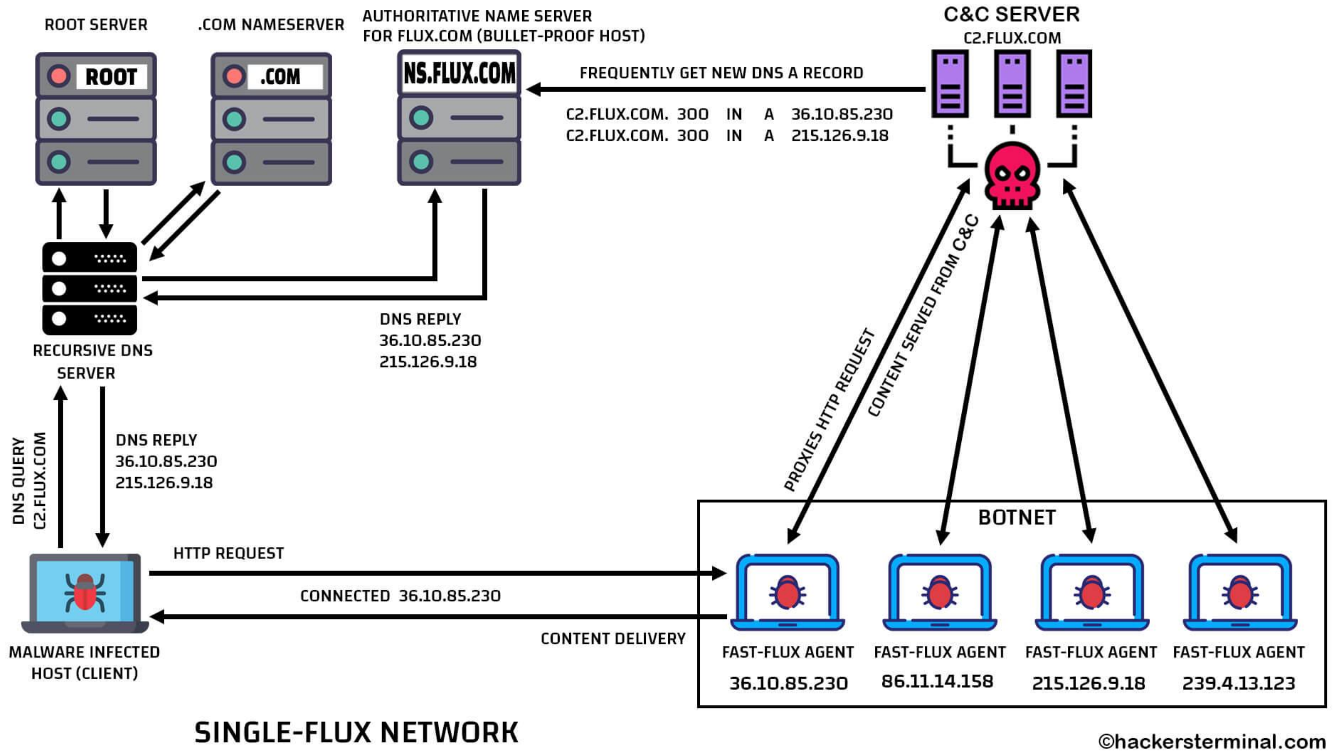
Fast Flux Botnets

Single Fast Flux

- C&C server is hidden behind a set of C&C fast flux (FF) agents that act as a reverse proxy to the actual C&C server
- An infected host contact one of the FF agents by resolving a fixed domain name (let’s say ffbotnet.com)
- FF agents change frequently over time
  - because the C&C rapidly update the DNS A record for ffbotnet.com on a bulletproof server
- The infected host sends a control message (let’s say a HTTP message) to (one of the) the FF agent
- The FF agent relay the request to C&C server and replay the response back to the infected host
- **Problem**: fixed authoritative DNS server for the FF domain

Single Fast Flux

https://hackersterminal.comfast-flux-service-networks-ffsn-technique/

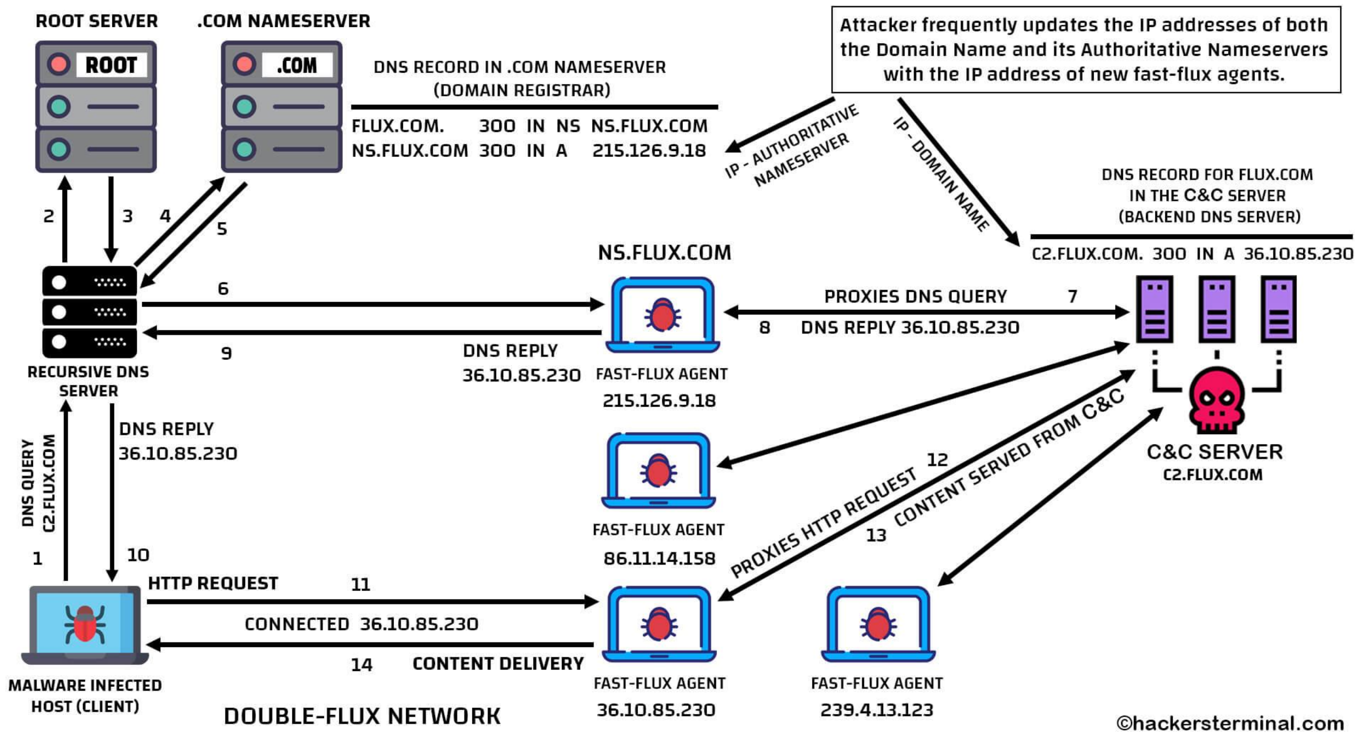

ROOT SERVER
ROOT
.COM NAMESERVER
.COM
AUTHORITATIVE NAME SERVER FOR FLUX.COM (BULLET-PROOF HOST)
NS.FLUX.COM
C&C SERVER
C2.FLUX.COM
FREQUENTLY GET NEW DNS A RECORD
C2.FLUX.COM. 300 IN A 36.10.85.230
C2.FLUX.COM. 300 IN A 215.126.9.18
RECURSIVE DNS SERVER
DNS REPLY 36.10.85.230 215.126.9.18
PROXIES HTTP REQUEST
CONTENT SERVED FROM C&C
DNS QUERY C2.FLUX.COM
DNS REPLY 36.10.85.230 215.126.9.18
BOTNET
HTTP REQUEST
CONNECTED 36.10.85.230
CONTENT DELIVERY
MALWARE INFECTED HOST (CLIENT)
FAST-FLUX AGENT 36.10.85.230
FAST-FLUX AGENT 86.11.14.158
FAST-FLUX AGENT 215.126.9.18
FAST-FLUX AGENT 239.4.13.123
SINGLE-FLUX NETWORK
©hackersterminal.com

Double Fast Flux

- Same as single FF but also the authoritative DNS server change frequently over time
- C&C server points the authoritative server for the FF domain to one of the available FF agents
- The FF agents act as both
  - C&C commands relay
  - DNS query relay
- **Problem**: still fixed domain name for the FF domain

Double Fast Flux

https://hackersterminal.com/fast-flux-service-networks-ffsn-technique/

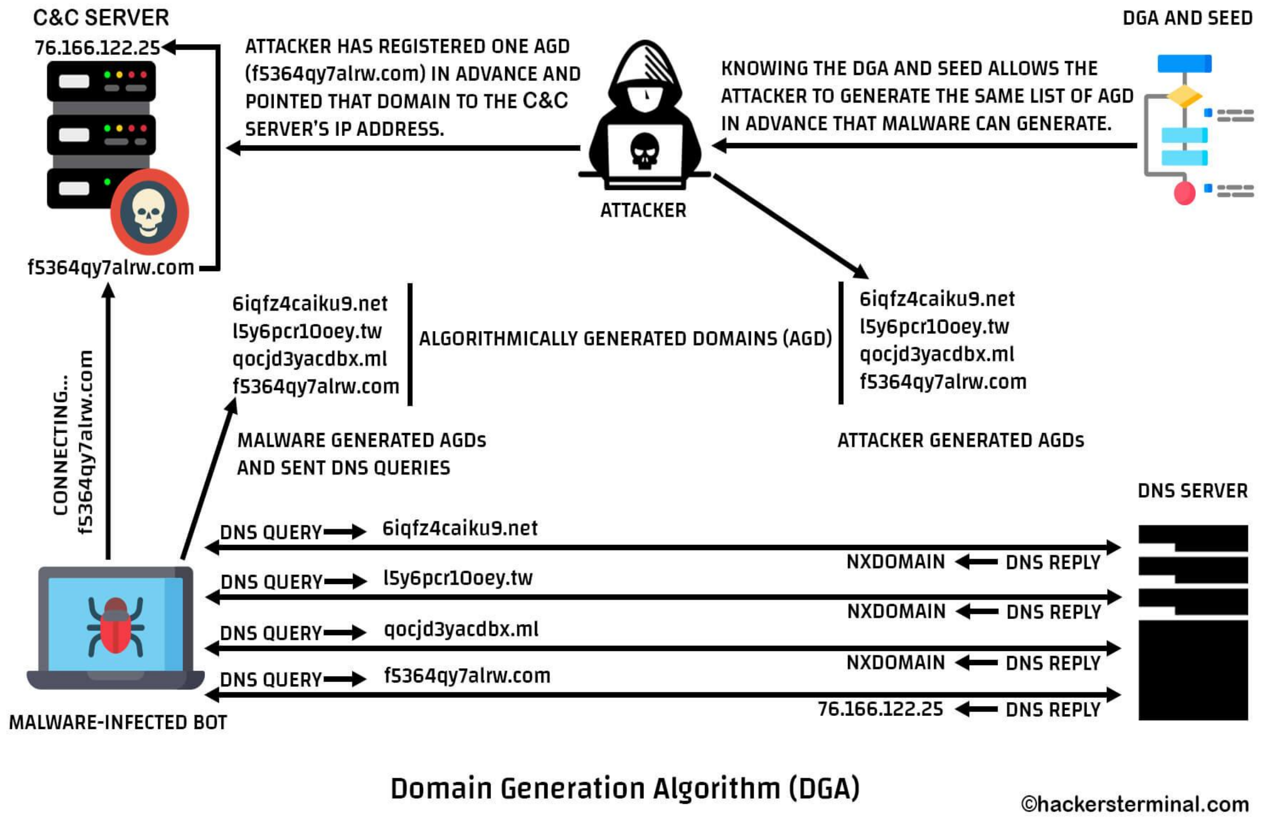

ROOT SERVER
ROOT
.COM NAMESERVER
.COM
DNS RECORD IN .COM NAMESERVER (DOMAIN REGISTRAR)
FLUX.COM. 300 IN NS NS.FLUX.COM
NS.FLUX.COM 300 IN A 215.126.9.18
Attacker frequently updates the IP addresses of both the Domain Name and its Authoritative Nameservers with the IP address of new fast-flux agents.
IP - AUTHORITATIVE NAMESERVER
IP - DOMAIN NAME
DNS RECORD FOR FLUX.COM IN THE C&C SERVER (BACKEND DNS SERVER)
C2.FLUX.COM. 300 IN A 36.10.85.230
2
3
4
5
6
NS.FLUX.COM
PROXIES DNS QUERY
7
8
DNS REPLY 36.10.85.230
9
DNS REPLY 36.10.85.230
FAST-FLUX AGENT 215.126.9.18
RECURSIVE DNS SERVER
DNS QUERY C2.FLUX.COM
1
DNS REPLY 36.10.85.230
10
FAST-FLUX AGENT 86.11.14.158
C&C SERVER C2.FLUX.COM
12
PROXIES HTTP REQUEST
13
CONTENT SERVED FROM C&C
HTTP REQUEST
11
CONNECTED 36.10.85.230
14
CONTENT DELIVERY
MALWARE INFECTED HOST (CLIENT)
FAST-FLUX AGENT 36.10.85.230
FAST-FLUX AGENT 239.4.13.123
DOUBLE-FLUX NETWORK
©hackersterminal.com

DOMAIN GENERATION ALGORITHM (DGA)

- Fixed FF domain names can be blacklisted (if discovered)
- ***Solution: use a malicious domain name that changes over time***
- The infected hosts and the C&C master (pseudo) randomly generate domain names using the same DGA algorithm
- As for any pseudo-random generator, if we start from the same seed we obtain the same pseudo-random sequence
  - ...hence the C&C master knows in advance the domain names that will be used
- The DGA seed is the secret shared among C&C master and infected hosts

Domain Generation Algorithms

https://hackersterminal.com/domain-generation-algorithm-dga-in-malware/


C&C SERVER
76.166.122.25
f5364qy7alrw.com
ATTACKER HAS REGISTERED ONE AGD (f5364qy7alrw.com) IN ADVANCE AND POINTED THAT DOMAIN TO THE C&C SERVER'S IP ADDRESS.
ATTACKER
KNOWING THE DGA AND SEED ALLOWS THE ATTACKER TO GENERATE THE SAME LIST OF AGD IN ADVANCE THAT MALWARE CAN GENERATE.
DGA AND SEED
6iqfz4caiku9.net
l5y6pcr10oey.tw
qocjd3yacdbx.ml
f5364qy7alrw.com
ALGORITHMICALLY GENERATED DOMAINS (AGD)
6iqfz4caiku9.net
l5y6pcr10oey.tw
qocjd3yacdbx.ml
f5364qy7alrw.com
CONNECTING... f5364qy7alrw.com
MALWARE GENERATED AGDs AND SENT DNS QUERIES
ATTACKER GENERATED AGDs
DNS SERVER
DNS QUERY 6iqfz4caiku9.net
NXDOMAIN DNS REPLY
DNS QUERY l5y6pcr10oey.tw
NXDOMAIN DNS REPLY
DNS QUERY qocjd3yacdbx.ml
NXDOMAIN DNS REPLY
DNS QUERY f5364qy7alrw.com
76.166.122.25 DNS REPLY
MALWARE-INFECTED BOT
Domain Generation Algorithm (DGA)
©hackersterminal.com

Intrusion Detection Systems

from ***“Computer Security: Principles and Practice” (Chapter 8)***
William Stallings and Lawrie Brown

Intrusion Detection Systems

- ❏ A hardware or software function that gathers and analyzes information from various areas within a computer or a network to identify possible security intrusions
- ❏ ***Host-based IDS (HIDS):*** Monitors the characteristics of a single host for suspicious activity
- ❏ ***Network-based IDS (NIDS):*** Monitors network traffic and analyzes network, transport, and application protocols to identify suspicious activity
- ❏ ***Distributed or hybrid IDS:*** Combines information from a number of sensors, often both host and network based, in a central analyzer that is able to better identify and respond to intrusion activity

Intrusion Detection Systems' Requirements

- Run continually
- Be fault tolerant
- Resist subversion
- Impose a minimal overhead on system
- Configured according to system security policies
- Adapt to changes in systems and users
- Scale to monitor large numbers of systems
- Provide graceful degradation of service
- Allow dynamic reconfiguration

Two Approaches to Intrusion Detection

Anomaly detection

- Involves the collection of data relating to the behavior of legitimate users over a period of time
- Current observed behavior is analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

- Uses a set of known malicious data patterns or attack rules that are compared with current behavior
- Also known as misuse detection
- Can only identify known attacks for which it has patterns or rules

Anomaly Detection

- **Statistical**
  - Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics
- **Knowledge based**
  - Approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior
- **Machine Learning**
  - Approaches automatically determine a suitable classification model from the training data using data mining techniques

Signature or Heuristic Detection

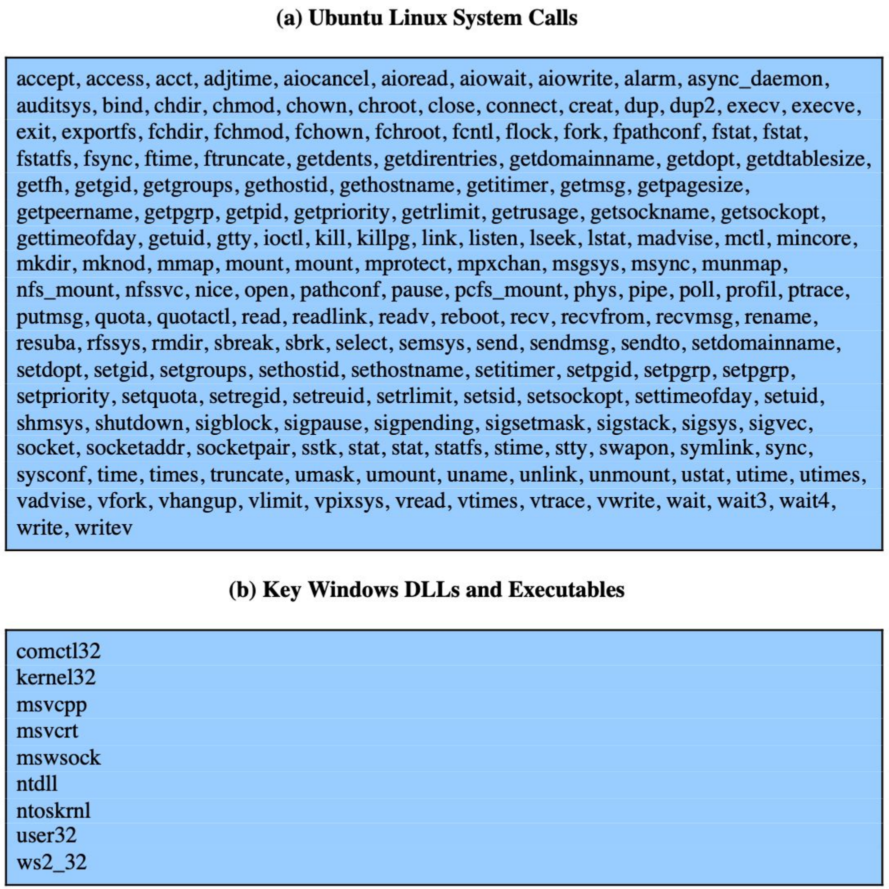
- **Signature approaches**
  - Match a large collection of known patterns of malicious data against data stored on a system or in transit over a network
  - The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data
  - Widely used in anti-virus products, network traffic scanning proxies, and in NIDS
- **Rule-based heuristic identification**
  - Involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses
  - Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage
  - Typically rules used are specific

Host-based IDS (HIDS)

- Adds a specialized layer of security software to vulnerable or sensitive systems
- Can use either anomaly or signature and heuristic approaches
- Monitors activity to detect suspicious behavior
  - Primary purpose is to detect intrusions, log suspicious events, and send alerts
  - Can detect both external and internal intrusions

Data Sources and Sensors

- ❑ Common data sources include:
  - ❑ System call traces
  - ❑ Audit (log file) records
  - ❑ File integrity checksums
  - ❑ Registry access



(a) Ubuntu Linux System Calls

accept, access, acct, adjtime, aiocancel, aioread, aiowait, aiowrite, alarm, async_daemon, auditsys, bind, chdir, chmod, chown, chroot, close, connect, creat, dup, dup2, execv, execve, exit, exportfs, fchdir, fchmod, fchown, fchroot, fcntl, flock, fork, fpathconf, fstat, fstat, fstatfs, fsync, ftime, ftruncate, getdents, getdirentries, getdomainname, getdopt, getdtablesize, getfh, getgid, getgroups, gethostid, gethostname, getitimer, getmsg, getpagesize, getpeername, getpgrp, getpid, getpriority, getrlimit, getrusage, getsockname, getsockopt, gettimeofday, getuid, gtty, ioctl, kill, killpg, link, listen, lseek, lstat, madvise, mctl, mincore, mkdir, mknod, mmap, mount, mount, mprotect, mpxchan, msgsys, msync, munmap, nfs_mount, nfssvc, nice, open, pathconf, pause, pcfs_mount, phys, pipe, poll, profil, ptrace, putmsg, quota, quotactl, read, readlink, readv, reboot, recv, recvfrom, recvmsg, rename, resuba, rfssys, rmdir, sbreak, sbrk, select, semsys, send, sendmsg, sendto, setdomainname, setdopt, setgid, setgroups, sethostid, sethostname, setitimer, setpgid, setpgrp, setpgrp, setpriority, setquota, setregid, setreuid, setrlimit, setsid, setsockopt, settimeofday, setuid, shmsys, shutdown, sigblock, sigpause, sigpending, sigsetmask, sigstack, sigsys, sigvec, socket, socketaddr, socketpair, sstk, stat, stat, statfs, stime, stty, swapon, symlink, sync, sysconf, time, times, truncate, umask, umount, uname, unlink, unmount, ustat, utime, utimes, vadvise, vfork, vhangup, vlimit, vpixsys, vread, vtimes, vtrace, vwrite, wait, wait3, wait4, write, writev

(b) Key Windows DLLs and Executables

comctl32
kernel32
msvcpp
msvcrt
mswsock
ntdll
ntoskrnl
user32
ws2_32

Network-Based IDS (NIDS)

- ❑ Monitors traffic at selected points on a network (which may be an end device network interface)
- ❑ Examines traffic packet by packet in real or close to real time
- ❑ May examine network, transport, and/or application-level protocol activity
- ❑ Comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface
- ❑ Analysis of traffic patterns may be done at the sensor, the management server or a combination of the two

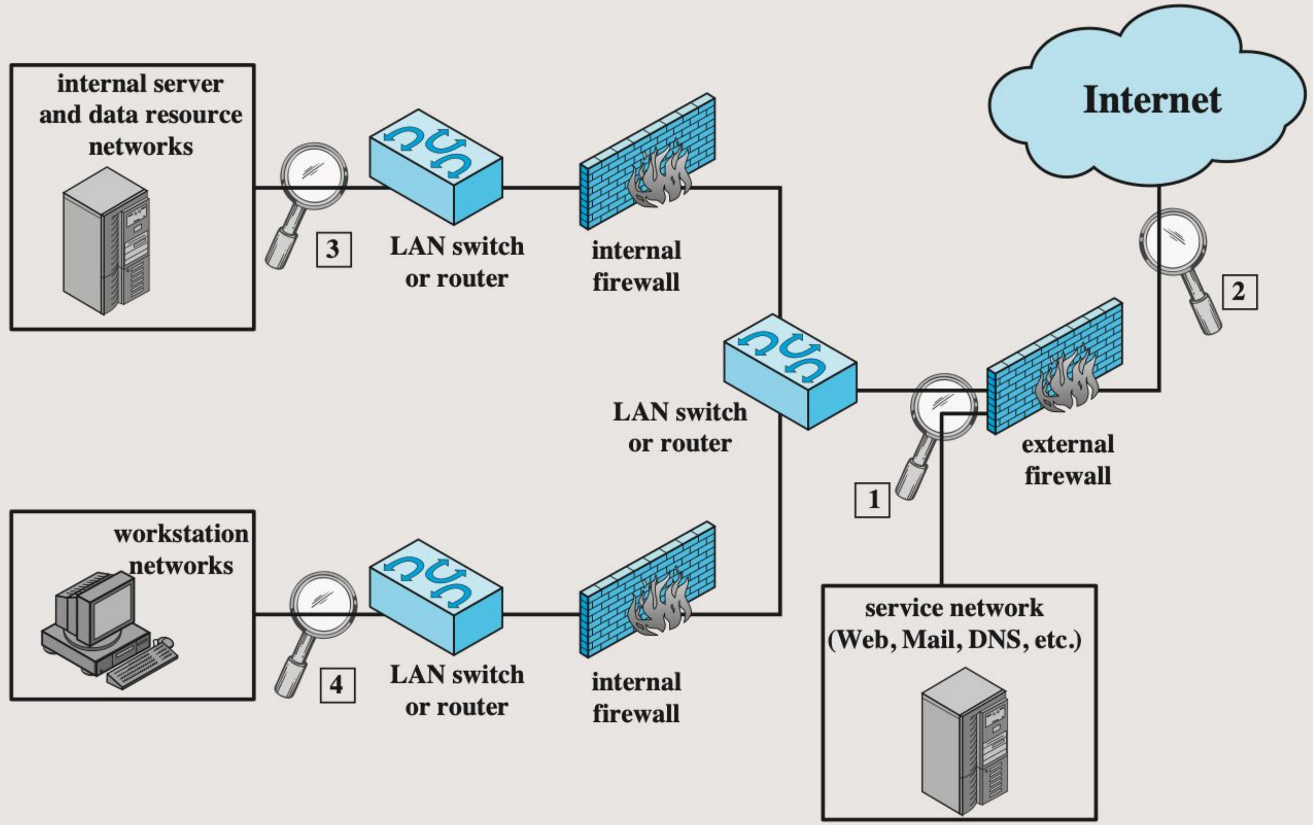


Figure 8.5 Example of NIDS Sensor Deployment

Honeypots

- Decoy systems designed to:
    - Lure a potential attacker away from critical systems
    - Collect information about the attacker’s activity
    - Encourage the attacker to stay on the system long enough for administrators to respond
- Systems are filled with fabricated information that a legitimate user of the system wouldn’t access
- Resources that have no production value
    - Therefore incoming communication is most likely a probe, scan, or attack
    - Initiated outbound communication suggests that the system has probably been compromised

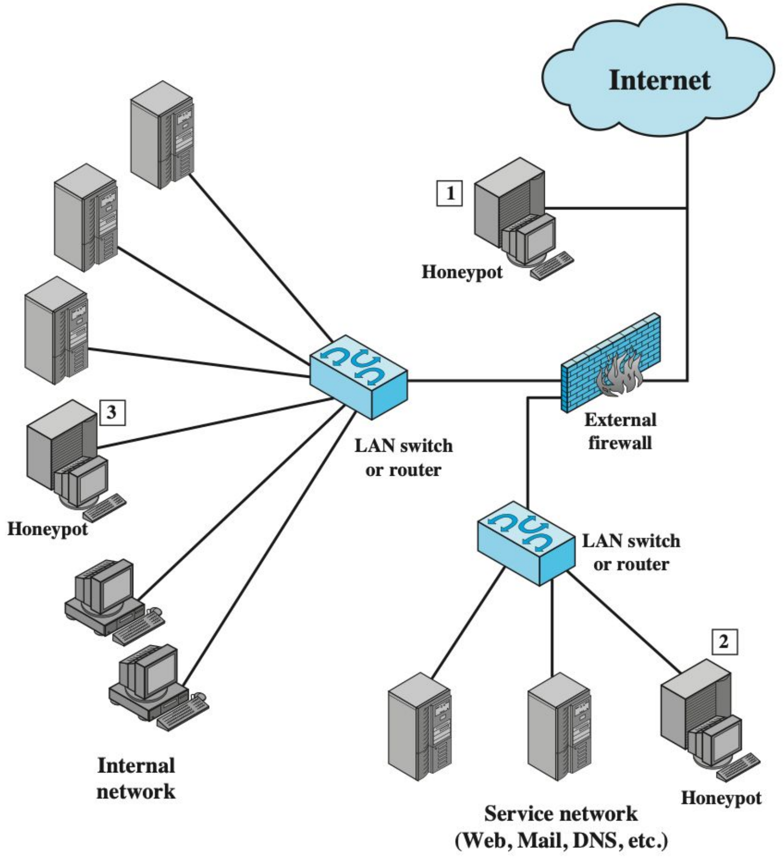


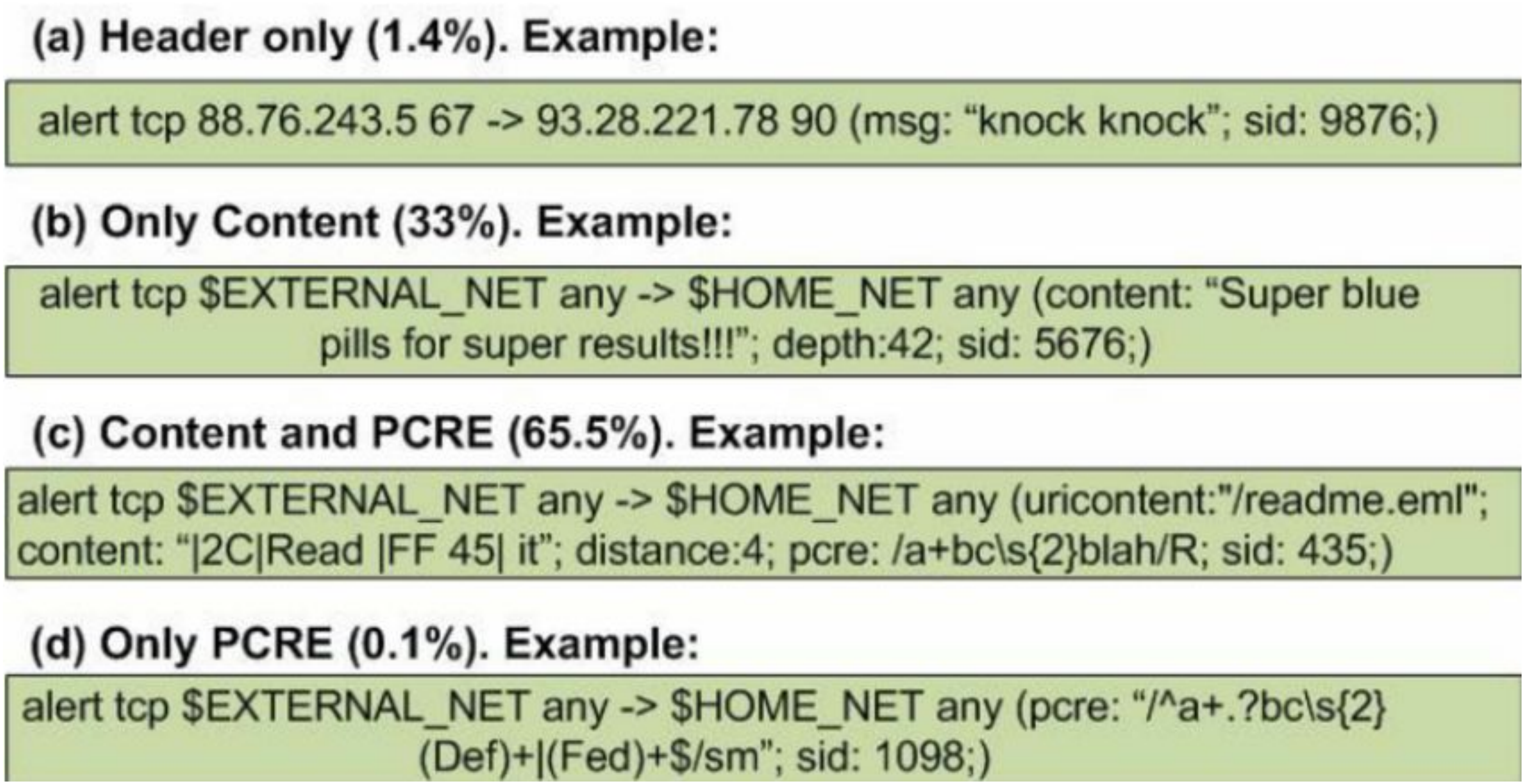
Figure 8.8 Example of Honeypot Deployment

Snort: an open source IPS

IPS = IDS + prevention (i.e. detect and react)

- ❏ Snort is the foremost Open Source Intrusion Prevention System (IPS) in the world
- ❏ Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users
- ❏ Can be deployed inline to stop these packets, as well.
- ❏ Three primary uses:
  - ❏ packet sniffer like tcpdump
  - ❏ packet logger, which is useful for network traffic debugging,
  - ❏ or it can be used as a full-blown network intrusion prevention system
- ❏ Other relevant IPS: *Suricata* (*https://suricata.io/*)

Snort Rule Examples



(a) Header only (1.4%). Example:

```
alert tcp 88.76.243.5 67 -> 93.28.221.78 90 (msg: “knock knock”; sid: 9876;)
```

(b) Only Content (33%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (content: “Super blue
pills for super results!!!”; depth:42; sid: 5676;)
```

(c) Content and PCRE (65.5%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (uricontent:"/readme.eml";
content: “|2C|Read |FF 45| it”; distance:4; pcre: /a+bc\s{2}blah/R; sid: 435;)
```

(d) Only PCRE (0.1%). Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (pcre: “/^a+.?bc\s{2}
(Def)+|(Fed)+$/sm”; sid: 1098;)
```

IPS Rulesets

- With these IPS you can create custom heuristic rules, but generally you download a set of pre-defined rules from external repositories
  - continuously updated based on newly discovered attacks (e.g. CVEs)
  - contain rules for detecting botnet specific traffic behavior (e.g. Mirai botnet)
- They can be public, i.e. licensed with open source licenses (MIT, GPLv2, etc.)
  - Proofpoint Emerging Threats Rules
  - Snort Community Rulesets
  - Suricata Traffic ID Rules
- Or private with commercial licenses (usually handling specific threats)
  - Secureworks - Malware Ruleset
  - Proofpoint - Emerging Threats Pro Rules
  - Stamus Networks - Newly Registered Domains Open