Ethical Hacking / Penetration testing Network Security

Boldizsár Bencsáth PhD

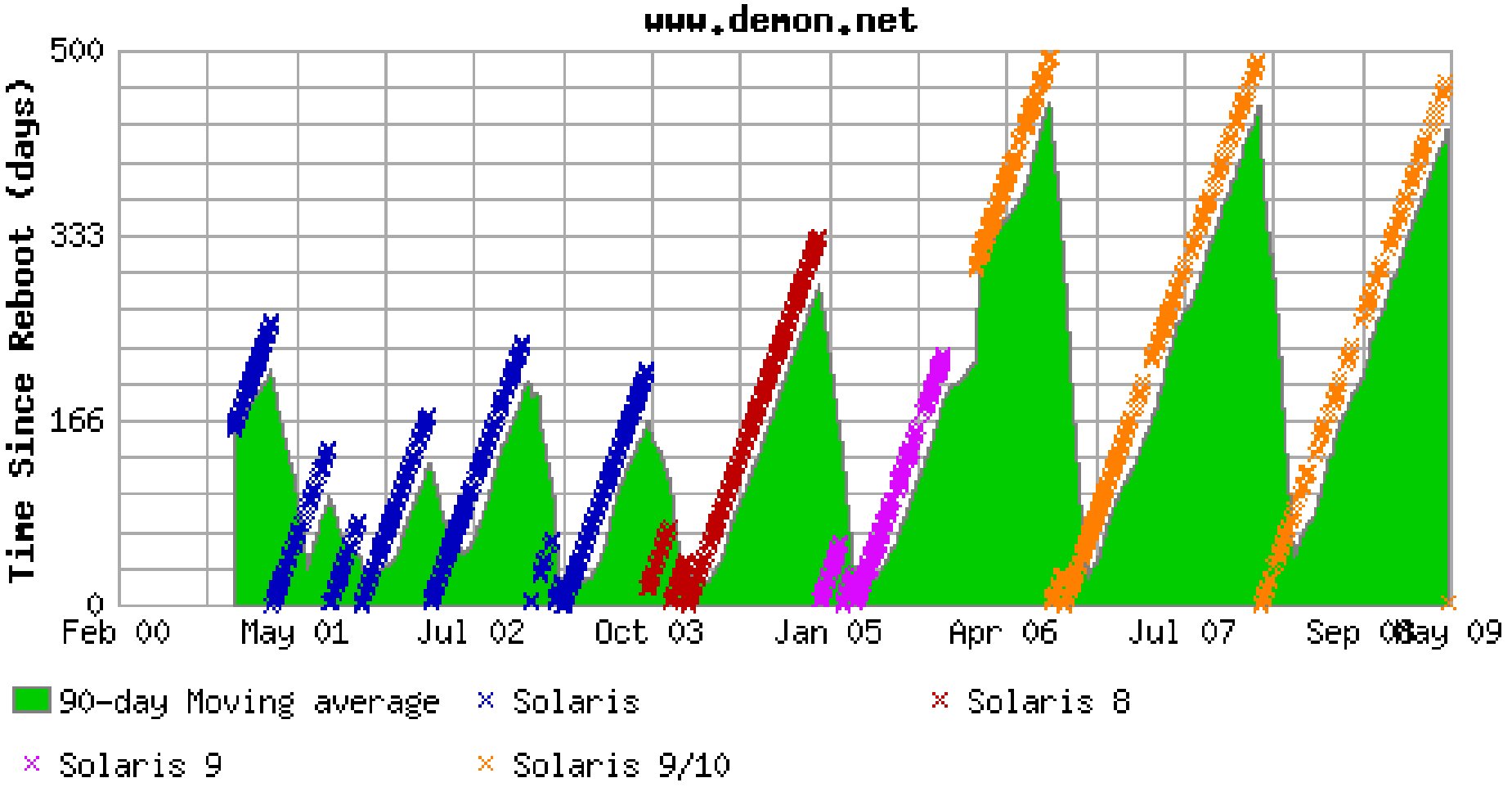
Web info leak

- Apache gives out important module and version informations in error messages and protocol header
- ServerSignature and ServerTokens config parameter can be used to disable such info leaks
- Server version is automatically collected by internet sites such as netcraft.com – historical information might be retrieved
- Also, packed based uptime information is recorded

Tcp timestamping: check http://www.securiteam.com/securitynews/5NP0C153PI.html

- Search engines know lot about our servers
- You should be aware what others know about You

Uptime, remotely



Uptime. Remotely.

- **What is Timestamping? How can it be used to gain information about a running system? Timestamping is a TCP option, which may be set, and if set takes 12 bytes in the header (for each packet) in addition to the 20 bytes a TCP header normally takes. This is exclusive of any other options.**
- **Linux**
  Sends TS on first packet replied to - default always get TS
  Note:
  To disable do:
  echo 0 >/proc/sys/net/ipv4/tcp_timestamps
  To enable do:
  echo 1 >/proc/sys/net/ipv4/tcp_timestamps

  Increments 100 ticks/sec
  2.0.x does not support TCP Timestamps
  2.1.90+ Supports Timestamps
  2.2.x Supports Timestamps
  2.4.x Supports Timestamps
- OS Ticks/sec Rollover time

4.4BSD	2	34 years, 8 days, 17:27:27
Solaris 2	10	6 years, 293 days, 22:53:00
Linux 2.2+	100	248 days, 13:13:56
Cisco IOS	1000	24 days, 20:31:23

- **Windows**
  Win2k sends the timestamp after the syn/ack handshake is complete (sends 0 TS during the 3-way handshake) and increment every 100ms initial random number.
  95/98 does not support TS
  NT 3.5/4 does not support TS

Port scanning

- Port scanning: Information gathering tool to find out
  - What are the available services on a target
  - Considered services: mainly TCP and UDP
  - Finding out services that might be available but filtered by a firewall
  - Scanning tools have additional features such as passive, active fingeprinting, scanning multiple hosts, fast scanning etc.

  A Port scanner is generally an **Active** tool (it sends out packets and analyzes the results)

  Passive port scanning also possible (needs sniffing capabilities)

Simple TCP connect() scan

- -sT: TCP connect scan

listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

18:58:57.725838 00:18:f3:43:d9:e7 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 60: arp who-has 10.105.1.54 (ff:ff:ff:ff:ff:ff) tell 10.105.1.254

18:58:57.725868 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype ARP (0x0806), length 42: arp reply 10.105.1.54 is-at 00:0c:29:85:af:a2

18:58:57.743297 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 66: 10.105.1.254.49746 > 10.105.1.54.22: S 3002070029:3002070029(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 7>

18:58:57.743336 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 66: 10.105.1.54.22 > 10.105.1.254.49746: S 1601689082:1601689082(0) ack 3002070030 win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 5>

18:58:57.743768 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.49746 > 10.105.1.54.22: . ack 1 win 46

18:58:57.743859 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.49746 > 10.105.1.54.22: R 1:1(0) ack 1 win 46

Port scanning

- Initiate a TCP connection
- No answer? – Maybe our packet was filtered out (“discarded”), firewall?
- The answer is an RST packet? – Most likely a closed port
- The answer is a SYN packet? – Open port

- Details on port scanning:

http://nmap.org/book/man-port-scanning-techniques.html

TCP connect scan

- TCP connect() scanning : This is the most basic form of TCP scanning. The connect() system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, connect() will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges.
- Any user on most UNIX boxes is free to use this call.
- Another advantage is speed. While making a separate connect() call for every targeted port in a linear fashion would take ages over a slow connection, you can hasten the scan by using many sockets in parallel. Using non-blocking I/O allows you to set a low time-out period and watch all the sockets at once. (The big downside is that this sort of scan is easily detectable and filterable. The target hosts logs will show a bunch of connection and error messages for the services which take the connection and then have it immediately shutdown.)- This is not valid anymore

History of port scanning

- First there was nothing
- Then port scanners appeared
- Then portscan-detectors appeared
- And the sysadmins were happy that they knew that on the second Monday in February somebody tried to break in to their system
- And more and more attackers came
- And the logs were full with port scans, blacklisted IPs, and the sysadmin's phone was full with alert SMSs.
- And then the alerts were turned off and the portscanning activity is considered as 'background radiation' on the internet

An intermezzo - port knocking

- Port knocking is an authentication scheme “before” real connections happen
- E.g. We do not allow SSH connections (port 22), only with port knocking.
- The client should first send packets to other ports for authentication purposes (special port number, order, timing and contents –crypto might help against simply session replays etc.)
- E.g. client first sends a SYN to port 1000,2000,3000,1000 within 2 seconds and then the port 22 will be opened (for a short period of time, and only for that client) (in real life, packet order is not static, e.g. jumping code might be used based on hash functions)
- Basically a good idea but it’s not that easy to decide: Time synchronization? Software problems (remote update through SSH hangs and cannot authenticate again)? Overhead? DoS possibilities?
- http://en.wikipedia.org/wiki/Port_knocking

TCP SYN scan

- -sS: TCP SYN Scan – it is also called Half-open scan
- 19:02:06.144781 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.40437 > 10.105.1.54.22: S 1864424766:1864424766(0) win 4096 <mss 1460>
  - 19:02:06.144846 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 58: 10.105.1.54.22 > 10.105.1.254.40437: S 270580685:270580685(0) ack 1864424767 win 5840 <mss 1460>
- 19:02:06.145277 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.40437 > 10.105.1.54.22: R 1864424767:1864424767(0) win 0

- Closed port:
- 19:06:55.603663 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.36008 > 10.105.1.54.299: S 4081495570:4081495570(0) win 1024 <mss 1460>
- 19:06:55.603755 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 54: 10.105.1.54.299 > 10.105.1.254.36008: R 0:0(0) ack 4081495571 win 0

TCP SYN scan

- TCP SYN scanning : This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and wait for a response. A SYN|ACK indicates the port is listening. A RST is indicative of a non-listener.
- If a SYN|ACK is received, you immediately send a RST to tear down the connection (actually the kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets. SYN scanning is the -s option of nmap.
- SYN scan is relatively unobtrusive and stealthy, since it never completes TCP connections. It also works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's FIN/NULL/Xmas, Maimon and idle scans do. It also allows clear, reliable differentiation between the open, closed, and filtered states.

FIN scan

- -sF FIN scan (closing TCP connection that does not exists)
- 19:10:13.385148 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.45939 > 10.105.1.54.22: F 2976776891:2976776891(0) win 4096
- 19:10:13.487156 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.45940 > 10.105.1.54.22: F 2976842426:2976842426(0) win 3072

- For closed ports:
- 19:11:01.269659 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60: 10.105.1.254.52539 > 10.105.1.54.299: F 12278294:12278294(0) win 2048
  - 19:11:01.269705 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 54: 10.105.1.54.299 > 10.105.1.254.52539: R 0:0(0) ack 12278295 win

TCP FIN scan

- TCP FIN scanning : There are times when even SYN scanning isn't clandestine enough. Some firewalls and packet filters watch for SYNs to restricted ports, and programs like synlogger and Courtney are available to detect these scans. FIN packets, on the other hand, may be able to pass through unmolested. This scanning technique was featured in detail by Uriel Maimon in Phrack 49, article 15.
- The idea is that closed ports tend to reply to your FIN packet with the proper RST.
- Open ports, on the other hand, tend to ignore the packet in question.
- As Alan Cox has pointed out, this is required TCP behavior. However, some systems (notably Micro$oft boxes), are broken in this regard. They send RST's regardless of the port state, and thus they aren't vulnerable to this type of scan. It works well on most other systems I've tried. Actually, it is often useful to discriminate between a *NIX and NT box, and this can be used to do that. FIN scanning is the -U (Uriel) option of nmap.

Null, Xmas scan

- Similar scans to FIN scan:
- NULL scan:
  - Does not set any bits (TCP flag header is 0)
- Xmas scan:
  - Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree.

Firewall detection: ACK scan

- This scan is different than the others discussed so far in that it never determines open (or even open|filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.
- The ACK scan probe packet has only the ACK flag set (unless you use --scanflags).
- When scanning unfiltered systems, open and closed ports will both return a RST packet. Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is undetermined.
- Ports that don't respond, or send certain ICMP error messages back (type 3, code 1, 2, 3, 9, 10, or 13), are labeled filtered.

ACK scan

- -sA ACK scan
- No firewall open port:

19:48:43.364020 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60:
10.105.1.254.39971 > 10.105.1.54.22: . ack 1810896703 win 3072

19:48:43.364063 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 54:
10.105.1.54.22 > 10.105.1.254.39971: R 1810896703:1810896703(0) win 0

- Closed port:

19:49:14.735544 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60:
10.105.1.254.53509 > 10.105.1.54.299: . ack 197384061 win 3072

19:49:14.735605 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 54:
10.105.1.54.299 > 10.105.1.254.53509: R 197384061:197384061(0) win 0

- iptables -I INPUT -j REJECT -p tcp --dport 299

19:50:02.273079 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60:
10.105.1.254.37866 > 10.105.1.54.299: . ack 150092826 win 3072

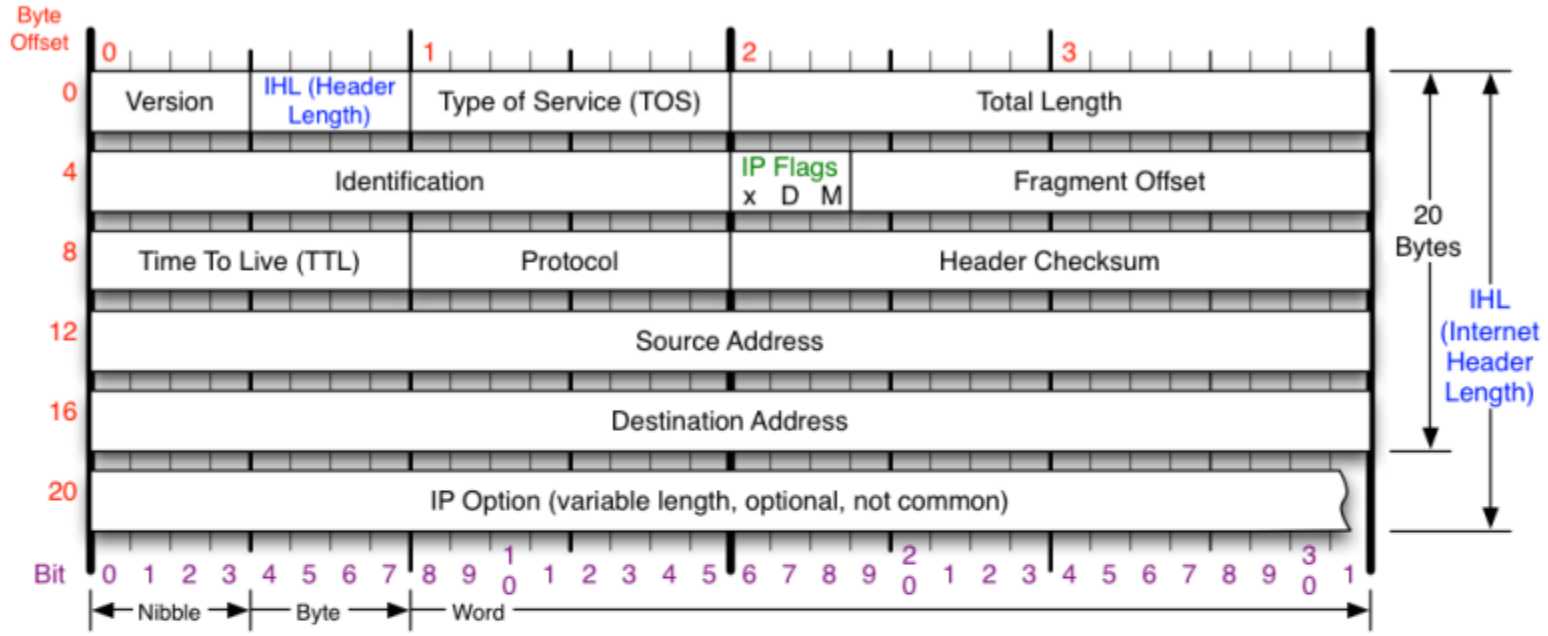
19:50:02.273123 00:0c:29:85:af:a2 > 00:18:f3:43:d9:e7, ethertype IPv4 (0x0800), length 82:
10.105.1.54 > 10.105.1.254: **ICMP** 10.105.1.54 tcp port 299 unreachable, length 48

- iptables -I INPUT -j DROP -p tcp --dport 299
- 19:51:05.153944 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60:
10.105.1.254.60120 > 10.105.1.54.299: . ack 2129899220 win 4096
- 19:51:05.257879 00:18:f3:43:d9:e7 > 00:0c:29:85:af:a2, ethertype IPv4 (0x0800), length 60:
10.105.1.254.60121 > 10.105.1.54.299: . ack 2129833685 win 1024

Zombie host scan, Idle scan

- -sI <*zombie host*>[:<*probeport*>] (idle scan)
  - This advanced scan method allows for a truly blind TCP port scan of the target (meaning no packets are sent to the target from your real IP address). Instead, a unique side-channel attack exploits predictable IP fragmentation ID sequence generation on the zombie host to glean information about the open ports on the target. IDS systems will display the scan as coming from the zombie machine you specify (which must be up and meet certain criteria). Full details of this fascinating scan type are in the section called "TCP Idle Scan (-sI)".
  - Besides being extraordinarily stealthy (due to its blind nature), this scan type permits mapping out IP-based trust relationships between machines. The port listing shows open ports *from the perspective of the zombie host.* So you can try scanning a target using various zombies that you think might be trusted (via router/packet filter rules).
  - You can add a colon followed by a port number to the zombie host if you wish to probe a particular port on the zombie for IP ID changes. Otherwise Nmap will use the port it uses by default for TCP pings (80).

IP header



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

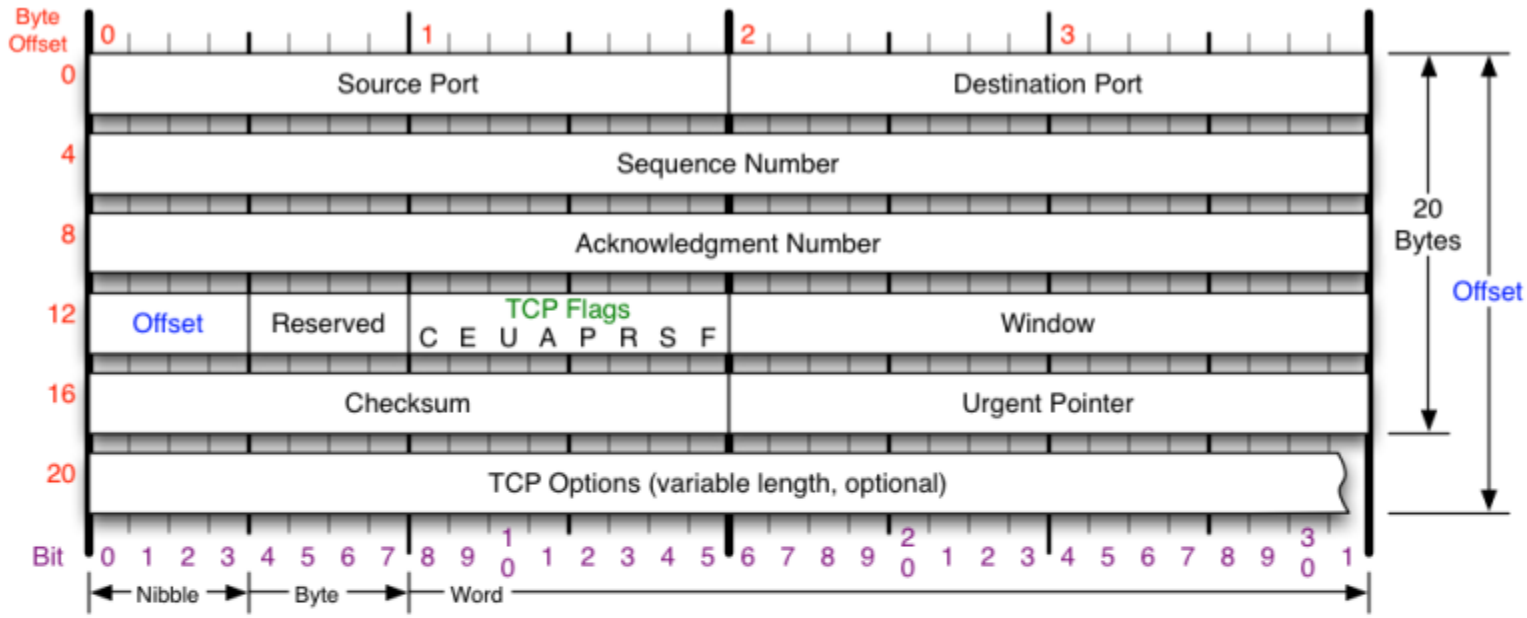
x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

TCP header



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

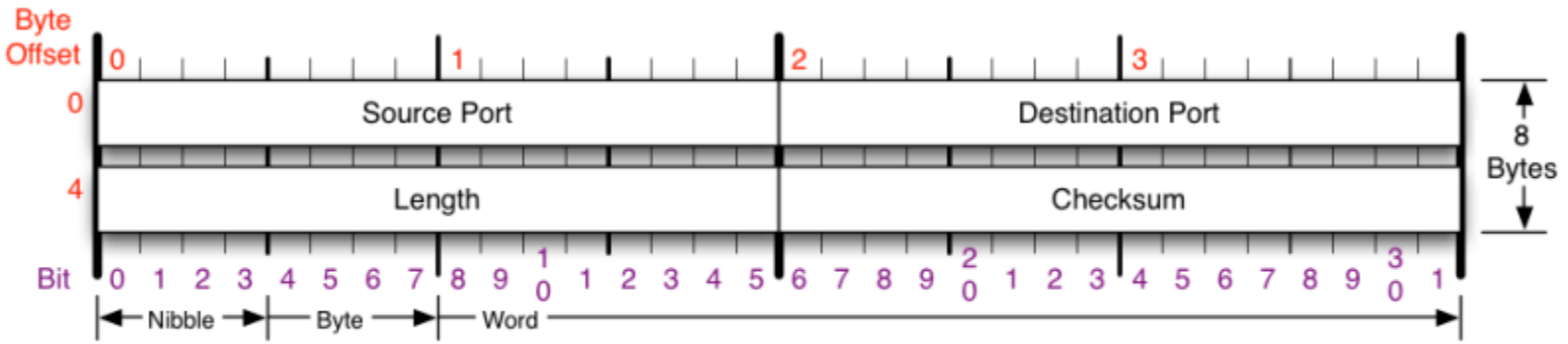
Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

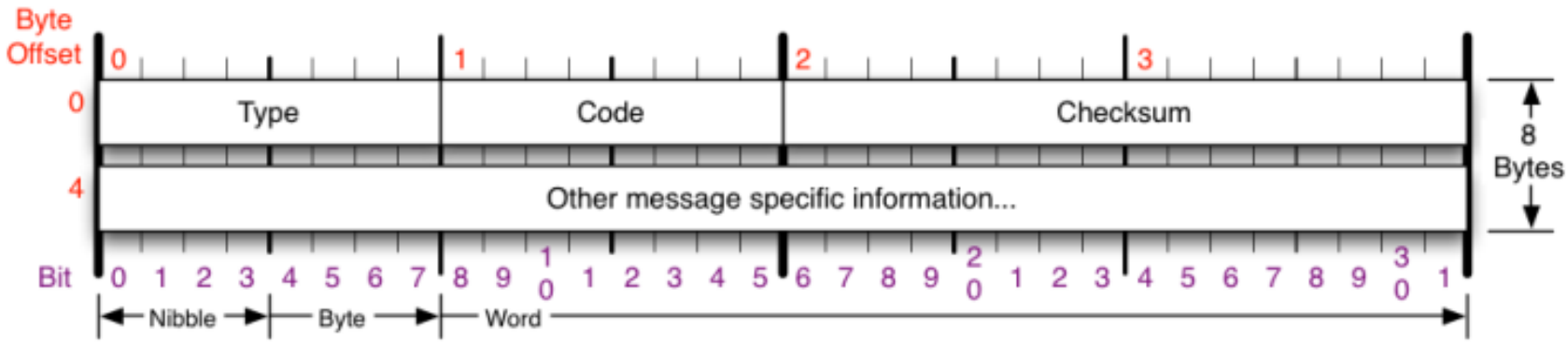
UDP header



Checksum of entire UDP segment and pseudo header (parts of IP header)

Please refer to RFC 768 for the complete User Datagram Protocol (UDP) Specification.

ICMP header



ICMP Message Types

Type	Code/Name
0	Echo Reply
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation required, and DF set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Network Administratively Prohibited
	10 Host Administratively Prohibited
	11 Network Unreachable for TOS
3	Destination Unreachable (continued)
	12 Host Unreachable for TOS
	13 Communication Administratively Prohibited
4	Source Quench
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the TOS & Network
	3 Redirect Datagram for the TOS & Host
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceded
	0 TTL Exceeded
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer Problem
	1 Missing a Required Operand
	2 Bad Length
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute

Checksum

Checksum of ICMP header

RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.

Idle scan

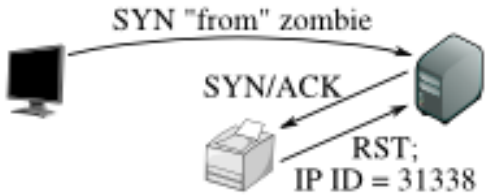
- Probe the zombie's IP ID and record it.
- Forge a SYN packet from the zombie and send it to the desired port on the target. Depending on the port state, the target's reaction may or may not cause the zombie's IP ID to be incremented.
- Probe the zombie's IP ID again. The target port state is then determined by comparing this new IP ID with the one recorded in step 1.
- The latest versions of Linux, Solaris and OpenBSD, Windows Vista are not suitable as zombie, since the IPID has been implemented with patches that randomized the IP ID.

Idle scan fig – from nmap

Step 1: Probe the zombie's IP ID.

The attacker sends a SYN/ACK to the zombie. The zombie, not expecting the SYN/ACK, sends back a RST, disclosing its IP ID.

Step 2: Forge a SYN packet from the zombie.



The target sends a SYN/ACK in response to the SYN that appears to come from the zombie. The zombie, not expecting it, sends back a RST, incrementing its IP ID in the process.

Step 3: Probe the zombie's IP ID again.

The zombie's IP ID has increased by 2 since step 1, so the port is open!

Version scanning

- -sV

Service scan Timing: About 55.56% done; ETC: 22:36 (0:00:05 remaining)

Nmap scan report for 10.105.1.54

Host is up (0.00064s latency).

Not shown: 986 closed ports

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	ProFTPD 1.3.1
22/tcp	open	ssh	(protocol 2.0)
23/tcp	open	telnet	Linux telnetd
25/tcp	open	smtp	Exim smtpd 4.69
53/tcp	open	domain	ISC BIND 9.5.1-P3
79/tcp	open	finger	Linux fingerd
80/tcp	open	http	Apache httpd
111/tcp	open	rpcbind	2 (rpc #100000)
143/tcp	open	imap	Courier Imapd (released 2008)

Network vulnerability scanners

- Target: A system, IP range, etc.
- Goal: To find vulnerable software components of the target in a fast and efficient way: Test against ten thousands of vulnerabilities in seconds
- Working method:
- Scans target for services
- identifies software version
- Performs basic tests to find out vulnerable services (e.g. is anonymous ftp login enabled?)
- Generally uses a number of “active plugins” to test target service against known vulnerabilities
- Uses a database that contains vulnerable software version numbers -> only matching this to the identified software version might result in large number of false positives
- Generally a lot more is incorporated (login support, password trial for weak pw., fuzzing tests, etc.)

Problems and advantages of vulnerability scanners

Problems

- Limited availability of free tools (Nessus: free, open source to closed source, limited free version)
- Vulnerability databases have to be kept updated
- Knowledge might be needed on the OS, Services to have accurate results (to avoid false positives)
- Attacks against custom settings, tools, software components is generally missing
- Generally, no “new attack” or system-wide vulnerability can be found
- The human knowledge is still needed

Advantages

- Automatic running, fast scanning of multiple hosts against thousands of vulnerabilities
- Good looking automatic reports as audit material
- Most of the internet-wide scanning attacks can be prevented (those attackers also use standard attacks, databases)

Software titles

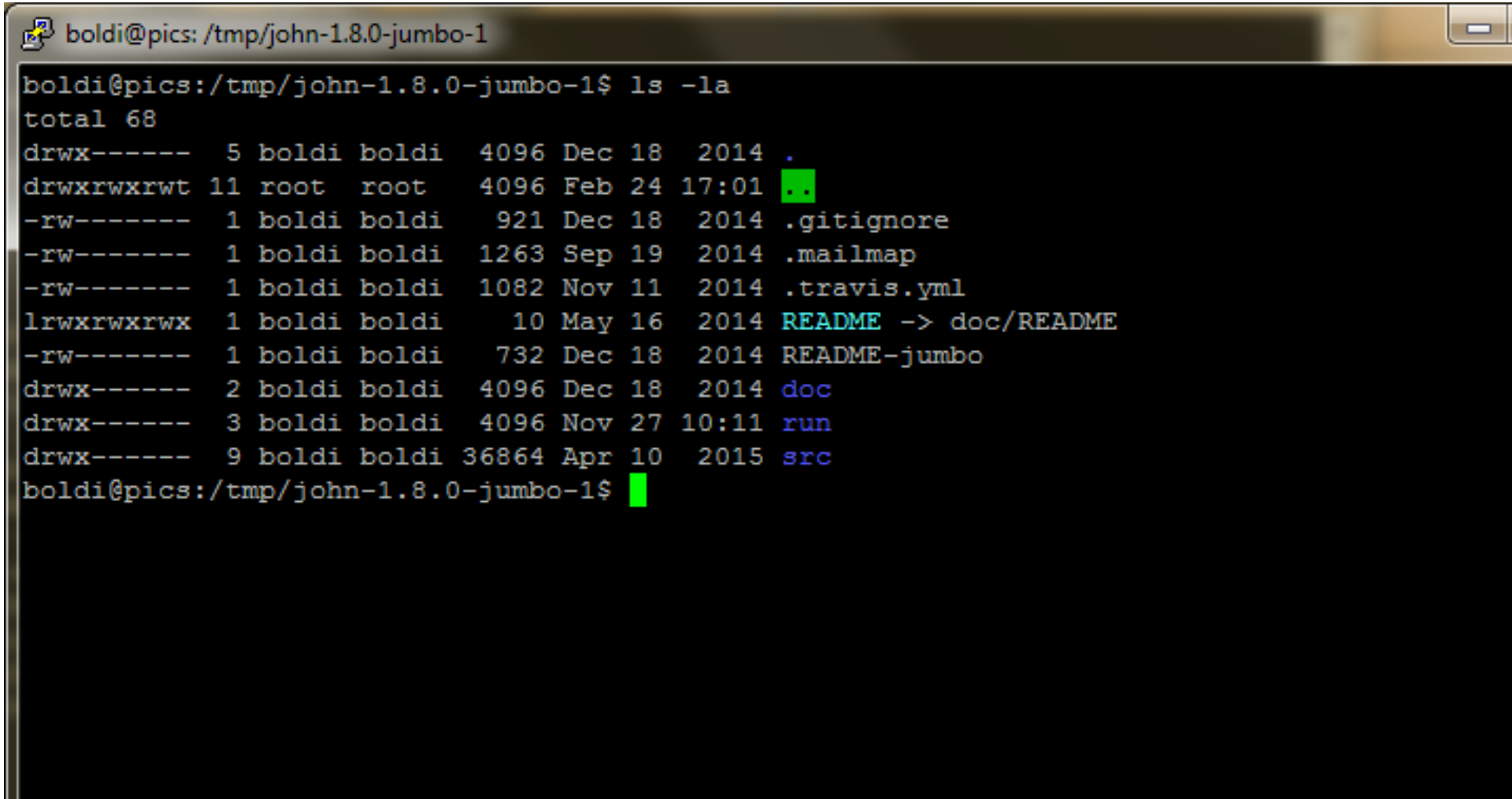
- Nmap: port scanner with many script plugins for vulnerability detection / free
- Nessus, Nexpose: automatic vulnerability scanners (mainly based on database of version-vuln infos
- Acunetix Web Vulnerability Scanner: In addition to find known problems good to find sql injection and XSS type of problems on unknown web applications
- Defensecode web scanner: community edition is free
- WPScan: wordpress vulnerability scanner
- SSLLabs: helps you to fix SSL related problems on web site
- OpenVAS: free nessus/nexpose alternative. Expect SLOOOOOOW operation

Acunetix scanner XSS scan

- Cross site scripting can affect many things on a web page
- Any parameter, value coming from user interaction is important
- Hidden tags, cookies, header parameters can also be affected
- Proper handling of user input is essential
- Automated tools exist to find out vulnerable variables
- It’s impossible to check every possibility: combinations, special values, handling of special characters, etc.
- Also: handling of input depends on the session – after login, during search, cart checkout, etc. – hard to check all the functionality
- Scanners might run for a long time and find out only few problems
- Source code and human intelligence can help to resolve issues
- Preventive steps are crucial: Learn secure programming

Password cracking

- OCLhashcat (GPU tricks)
- John the ripper



- Run directory contains actual programs
- Many utilities given just to convert encrypted passwords (e.g. pdf2john.py, 7z2john.py

```
boldi@pics:/tmp/john-1.8.0-jumbo-1/run$ ls
1password2john.py   dumb32.conf                  keychain2john       mkvcalcproba              rexgen2rules.pl
7z2john.py          dynamic.conf                 keychain2john.py    ml2john.py                sap2john.pl
SIPdump             dynamic_flat_sse_formats.conf keyring2john       mozilla2john.py           sha-dump.pl
aix2john.pl         ecryptfs2john.py             keystore2john       netntlm.pl                sha-test.pl
aix2john.py         efs2john.py                  keystore2john.py    netscreen.py              sipdump2john.py
alnum.chr           encfs2john.py                known_hosts2john.py odf2john.py               ssh2john
alnumspace.chr      fw.txt                       korelogic.conf      office2john.py            ssh2sshng.py
alpha.chr           genincstats.rb               kwallet2john        openbsd_softraid2john.py  sshng2john.py
androidfde2john.py  genmkvpwd                    kwallet2john.py     openssl2john.py           stats
apex2john.py        gpg2john                     lanman.chr          pass_gen.pl               strip2john.py
ascii.chr           hccap2john                   latin1.chr          password.lst              sxc2john.py
base64conv          hextoraw.pl                  ldif2john.pl        pcap2john.py              tgtsnarf
benchmark-unify     htdigest2john.py             leet.pl             pdf2john.py               truecrypt_volume2john
bitcoin2john.py     ikescan2john.py              lion2john-alt.pl    pfx2john                  uaf2john
blockchain2john.py  ios7tojohn.pl                lion2john.pl        putty2john                unafs
calc_stat           john                         lm_ascii.chr        pwsafe2john               undrop
cisco2john.pl       john.bash_completion         lotus2john.py       racf2john                 unique
cprepair            john.conf                    lower.chr           radius2john.pl            unshadow
cracf2john.py       john.local.conf              lowernum.chr        rar2john                  upper.chr
dictionary.rfc2865  john.old                     lowerspace.chr      raw2dyna                  uppernum.chr
digits.chr          john.zsh_completion          luks2john           regex_alphabets.conf      utf8.chr
dmg2john            kdcdump2john.py              mailer              relbench                  vncpcap2john
dmg2john.py         keepass2john                 makechr             repeats16.conf            wpapcap2john
dumb16.conf         kernels                      mcafee_epo2john.py  repeats32.conf            zip2john
boldi@pics:/tmp/john-1.8.0-jumbo-1/run$
```

- Let’s create an example password file:

boldi@pics:/tmp/john-1.8.0-jumbo-1/run$ htpasswd -c trial.pw boldi

New password:

Re-type new password:

Adding password for user boldi

boldi@pics:/tmp/john-1.8.0-jumbo-1/run$ cat trial.pw

boldi:$apr1$oOXAJIjJ$q8R2QkM2KP1cBMJN8d034.

(note: htpasswd and operating systems support multiple password formats, this is just one of the many)

Let’s crack the password

```
boldi@pics:/tmp/john-1.8.0-jumbo-1/run$ ./john trial.pw
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-opencl"
Use the "--format=md5crypt-opencl" option to force loading these as that type instead
Loaded 1 password hash (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 12x])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:00 21.74% 2/3 (ETA: 17:07:31) 0g/s 41648p/s 41648c/s 41648C/s harold3..college3
0g 0:00:00:02  3/3 0g/s 54885p/s 54885c/s 54885C/s morena..shance
0g 0:00:00:04  3/3 0g/s 53335p/s 53335c/s 53335C/s barch21..briesta
0g 0:00:00:08  3/3 0g/s 60903p/s 60903c/s 60903C/s myalove..sancin1
0g 0:00:00:16  3/3 0g/s 67653p/s 67653c/s 67653C/s jrmrys..cccaso
0g 0:00:00:18  3/3 0g/s 68607p/s 68607c/s 68607C/s biall15..127730
0g 0:00:00:36  3/3 0g/s 73232p/s 73232c/s 73232C/s lindall..linch09
pass123          (boldi)
1g 0:00:01:12 DONE 3/3 (2016-02-24 17:08) 0.01379g/s 75621p/s 75621c/s 75621C/s
passe08..pasho13
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

(g/s is successful guesses per second (so it'll stay at 0 until at least one password is cracked), p/s is candidate passwords tested per second, c/s is "crypts" (password hash or cipher computations) per second, and C/s is combinations of candidate password and target hash per second)

Many types of password cracking

- Based on dictionary
  - Trick: how to collect good dictionaries?
  - Which dictionaries to be used?
- Brute force
  - What characters to be used
  - Define a mask (e.g. capital first char, several [a-z], then 1 or 2 numbers)
- Special models (e.g. Markov chains)
- Based on user data (e.g. login ID, name, phone number)
- Combination of all described above
- Check john.conf for john tricks, and tools in hashcat-utils
- In some cases other tools are more efficient: e.g. old .zip password protection was prone to known cleartext-ciphertext attack