Defending Computer Networks
Lecture 4: Exploit Defenses

Stuart Staniford

Adjunct Professor of Computer Science

Logistics

- HW1
  - http://www.cs.cornell.edu/courses/cs5434/2015fa/hw1.pdf
- Due Monday 9/14/15 at midnight via CMS.
- Supplementary Lecture
  - Either Fri or Mon 6-7pm

8 in 10 Internet-connected baby monitors receive 'F' grade for security flaws

CVE-2015-2886	Remote	R7-2015-11.1	Predictable Information Leak	iBaby M6
CVE-2015-2887	Local Net, Device	R7-2015-11.2	Backdoor Credentials	iBaby M3S
CVE-2015-2882	Local Net, Device	R7-2015-12.1	Backdoor Credentials	Philips In.Sight B120/37
CVE-2015-2883	Remote	R7-2015-12.2	Reflective, Stored XSS	Philips In.Sight B120/37
CVE-2015-2884	Remote	R7-2015-12.3	Direct Browsing	Philips In.Sight B120/37
CVE-2015-2888	Remote	R7-2015-13.1	Authentication Bypass	Summer Baby Zoom Wifi Monitor & Internet Viewing System
CVE-2015-2889	Remote	R7-2015-13.2	Privilege Escalation	Summer Baby Zoom Wifi Monitor & Internet Viewing System
CVE-2015-2885	Local Net, Device	R7-2015-14	Backdoor Credentials	Lens Peek-a-View
CVE-2015-2881	Local Net	R7-2015-15	Backdoor Credentials	Gynoii
CVE-2015-2880	Device	R7-2015-16	Backdoor Credentials	TRENDnet WiFi Baby Cam TV-IP743SIC

Newly discovered vulnerabilities in baby monitors

HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities

Mark Stanislav and Tod Beardsley

http://www.networkworld.com/article/2979854/microsoft-subnet/8-in-10-internet-connected-baby-monitors-receive-f-grade-for-security-flaws.html

iBaby M3S, which currently sells for $169.95, ships with hardcoded credentials of admin for username and password. *Mitigations*: Customers are advised to ask the vendor about a firmware update to disable those credentials.

Philips B120/37, which currently sells for between $199.76 or $82.99 for the discontinued model, had several vulnerabilities. The device shipped with “hardcoded and statically generated credentials which can grant access to both the local web server and operating system. The operating system ‘admin’ and ‘mg3500’ account passwords are present due to the stock firmware used by this camera, which is used by other cameras on the market today.”

It’s 1988 all over again…

info | discussion | exploit | solution | references

Berkeley Sendmail DEBUG Vulnerability

Sendmail's debug mode allows the recipient of an email message to be a program that runs with the privileges of the user id which sendmail is running under. This user is normally root.

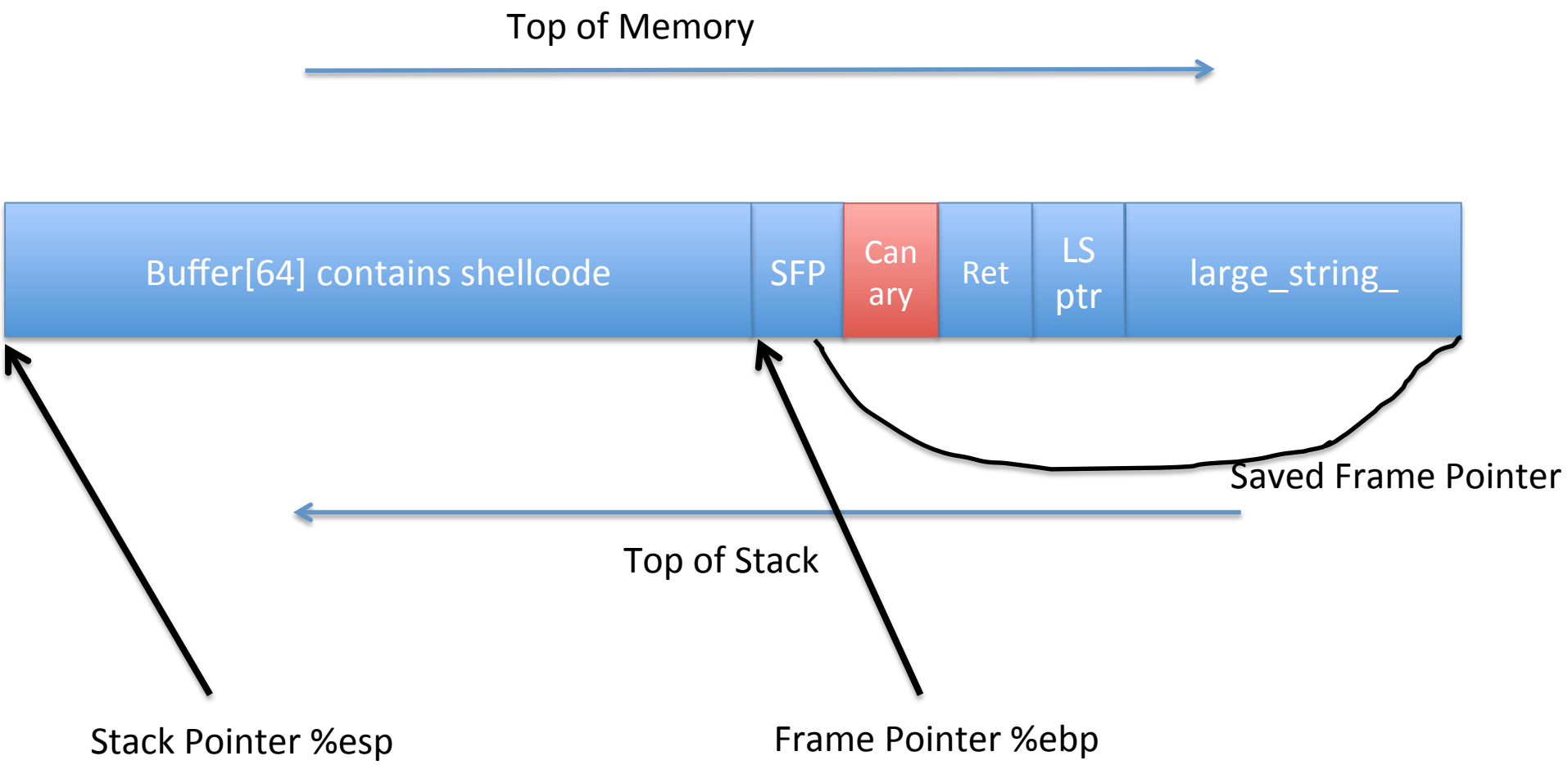
This allows an attacker to set the recipient to the shell and include shell commands in the message body.

This vulnerability was used by the Morris Worm.

Main Goals for Today

- Understand NX defenses against overflows
  - And sketch return oriented programming
- Understand Address Space Randomization
  - And how the dark side can work around it
- Maybe start our networking tutorial

Refresh on Canary Defenses



Not invincible. Eg http://phrack.org/issues.html?issue=56&id=5

Refresh heap vulnerability

```
struct myObject
{
  char name[64];
  int (*foo)(int);
 …
}
```

Note that in C++, virtual functions are stored implicitly in object structure as function pointers

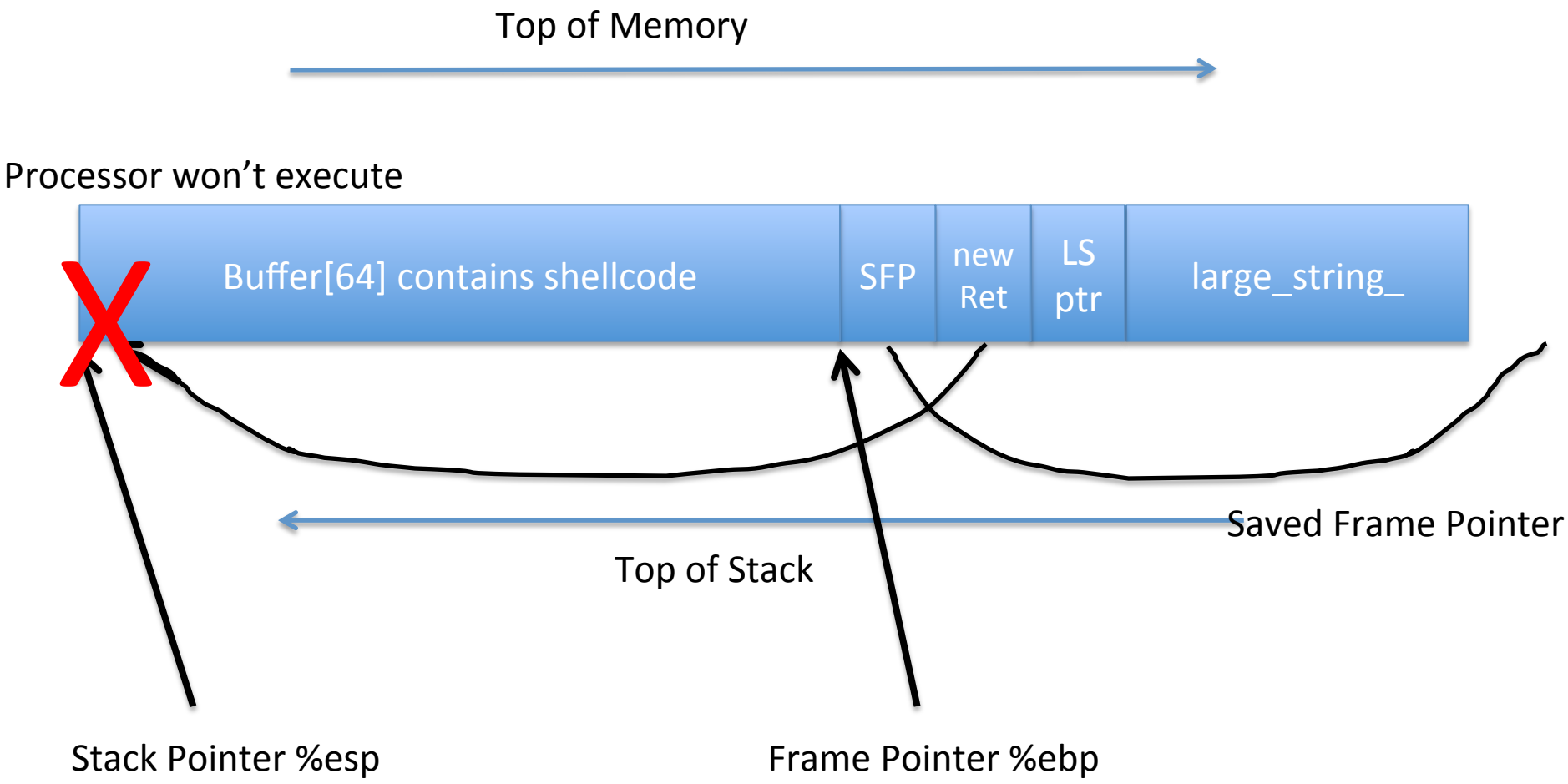
Heap Issues in General

- Often exploitable
- Not nearly as cookie-cutter as stack issues
- Requires more code-analysis from the attacker
  - Each case is different

NX/DEP/W^X

- Related mechanisms with common theme
  - Let's not execute the stack/heap
  - That way, cannot inject shellcode into buffer
  - And then point RIP at buffer contents
  - Ditto in format string attack, cannot put shellcode into buffer
- Requires hardware/OS support
  - But we have that now

NX/DEP/W^X


Top of Memory
Processor won't execute
Buffer[64] contains shellcode
SFP
new Ret
LS ptr
large_string_
Saved Frame Pointer
Top of Stack
Stack Pointer %esp
Frame Pointer %ebp

NX Bit

- General term for hardware feature
  - Originally AMD term
  - XD Bit (Intel)
  - XN Bit (ARM)
- Implemented in the page table
  - Bit 63 says this page cannot be executed
  - Hardware will enforce when doing memory lookup on instruction pointers in virtual address space
  - OS needs to manage the bits on the pages
    - Make sure stack and heap pages cannot execute

DEP: Data Execution Prevention

- Term for the OS Level feature
- Particularly on MS Windows
  - Controllable on a process-by-process basis
  - First optionally available on XP SP2 (circa 2004)
  - Default is still only to be available on core OS stuff
  - Optionally turn on for everything
    - Breaks some applications

W^X

- Write XOR Execute
- Extended version of idea
- All virtual pages can be
  - either writeable or executable
  - But not both
- Prevents self-modifying code
- OpenBSD, OS X, some Linux have full W^X

Defeating NX: Return to LIBC

- We can't make RIP point to our code
- But we can make it point to any pre-existing code on system
  - In text segment, so executable
- And we can set up stack beforehand
- Eg call system();
- So NX not invincible by itself

ROP: Return Oriented Programming

- Generalization of return-to-libc idea
- Shacham, 2007
- Idea is to crawl libc (etc), and find a series of "gadgets"
- A gadget is a useful bit of code right before the ret instruction of some function
- Might just be one or two instructions

Libc from a ROP POV

Etc, etc…

More ROP

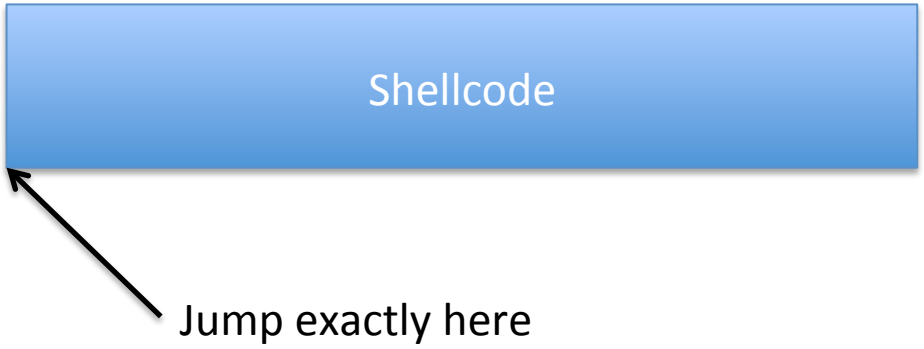
- Then call a bunch of these in sequence
  - from the (scribbled on) stack
- Shachem showed that libc ROP gadgets form a general purpose computation framework that can do anything.
- Very hard to fix this by surgery on libc

NOP Sleds

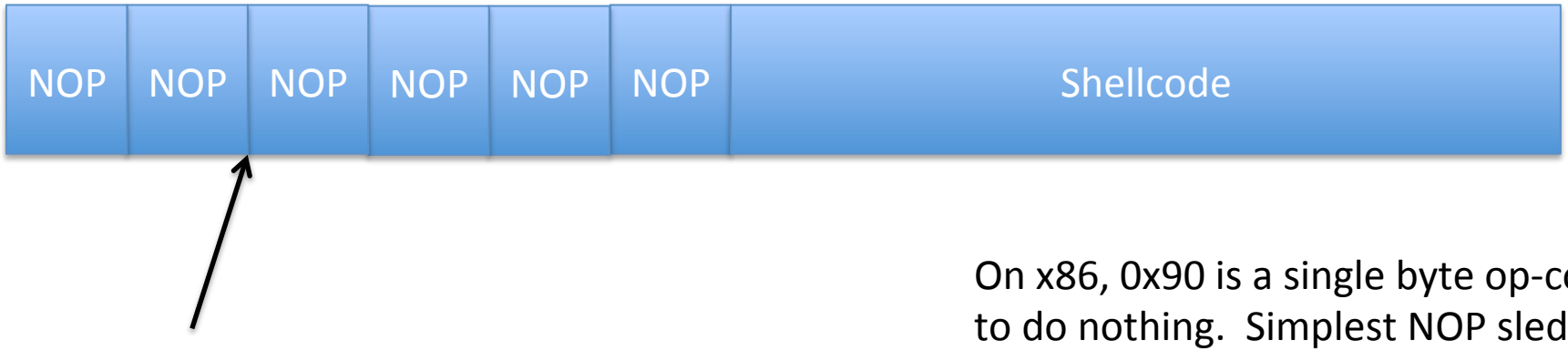
- When we overwrite an address in memory
  - Say a RIP
- We need to know what value to put.
- Simple case:
  - Beginning of shellcode in buffer
- But this is fragile.
  - Any slight difference in code version,
  - Even if it didn't affect the vulnerability
  - Could change the jump address

So allow for some imprecision

Instead of



Do



Jump somewhere in here. Now our exploit is less fragile. Assuming we have the space.

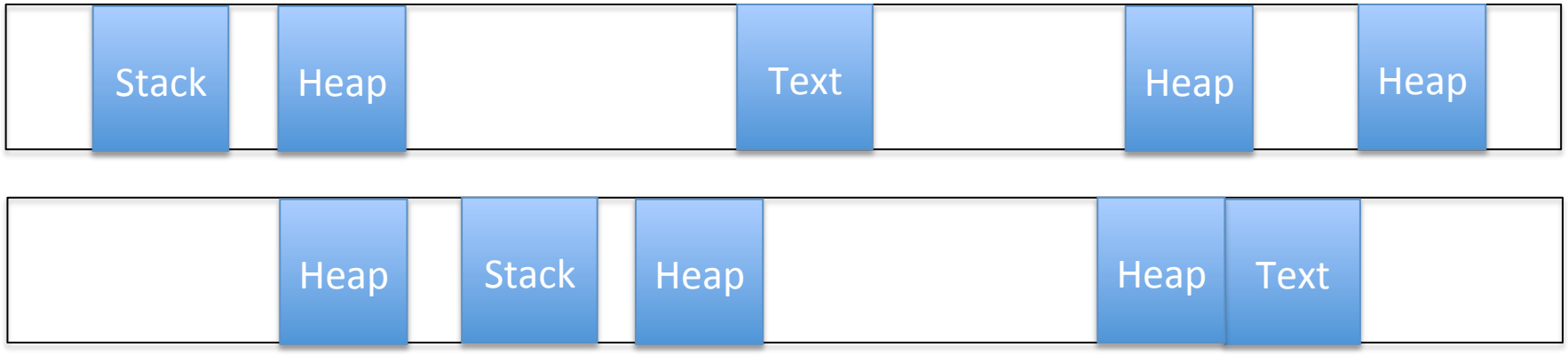
On x86, 0x90 is a single byte op-code to do nothing. Simplest NOP sled.

Address Space Layout Randomization

Basic insight is to make it really hard to figure out what address to jump to. Put key parts of the program in random places in memory

Instead of loading program into memory the same way every time:

Randomize:



So Now

- When we overwrite RIP, we don't know where to point, not even close
  - If we could get close, could use a NOP-sled

ASLR

- Now available on all major OS's
- Not all legacy code is compiled this way
  - May not be position-independent
- But increasingly becoming standard
- So a fully modern exploit must get past
  - Canaries
  - NX/DEP
  - ASLR
- All at the same time…
- But let's look at ASLR in isolation for a moment

Brute Forcing ASLR

- Loading at run time, we can't do fine-grained randomization
  - Code, for example, has all kinds of internal jumps that must be known, so code can't easily be jumbled up at the micro scale.
- Limited to moving around big chunks (stack, text, etc)
  - Consider an 8MB stack in 4GB (32 bit machine)
  - 512 possible positions. Not outrageous to guess
  - Much more difficult on 64 bit machines

Leaking Addresses

- Anything that allows us to see an address,
  - lets us get a handle on where that kind of thing lives
  - Eg format string vulnerability allows us to inspect the stack before doing our attack
    - We can quickly figure out where everything lives
      - Text pointers in RIPs
      - Stack pointers in stack frame bases
      - Heap pointers in local variable pointers to heap buffers

Defeating ALSR/DEP combined

- Any non-ALSR code can be analyzed for ROP.
  - Still sometimes libraries/code lying around. Eg
    - https://blogs.technet.com/b/srd/archive/2013/08/12/mitigating-the-ldrhotpatchroutine-dep-aslr-bypass-with-ms13-063.aspx

The bypass takes advantage of a predictable memory region known as SharedUserData that exists at a fixed location (0x7ffe0000) in every process on every supported version of Windows. On 64-bit versions of Windows prior to Windows 8, this region contains pointers to multiple functions in the 32-bit version of NTDLL that is used by WOW64 processes as shown below:

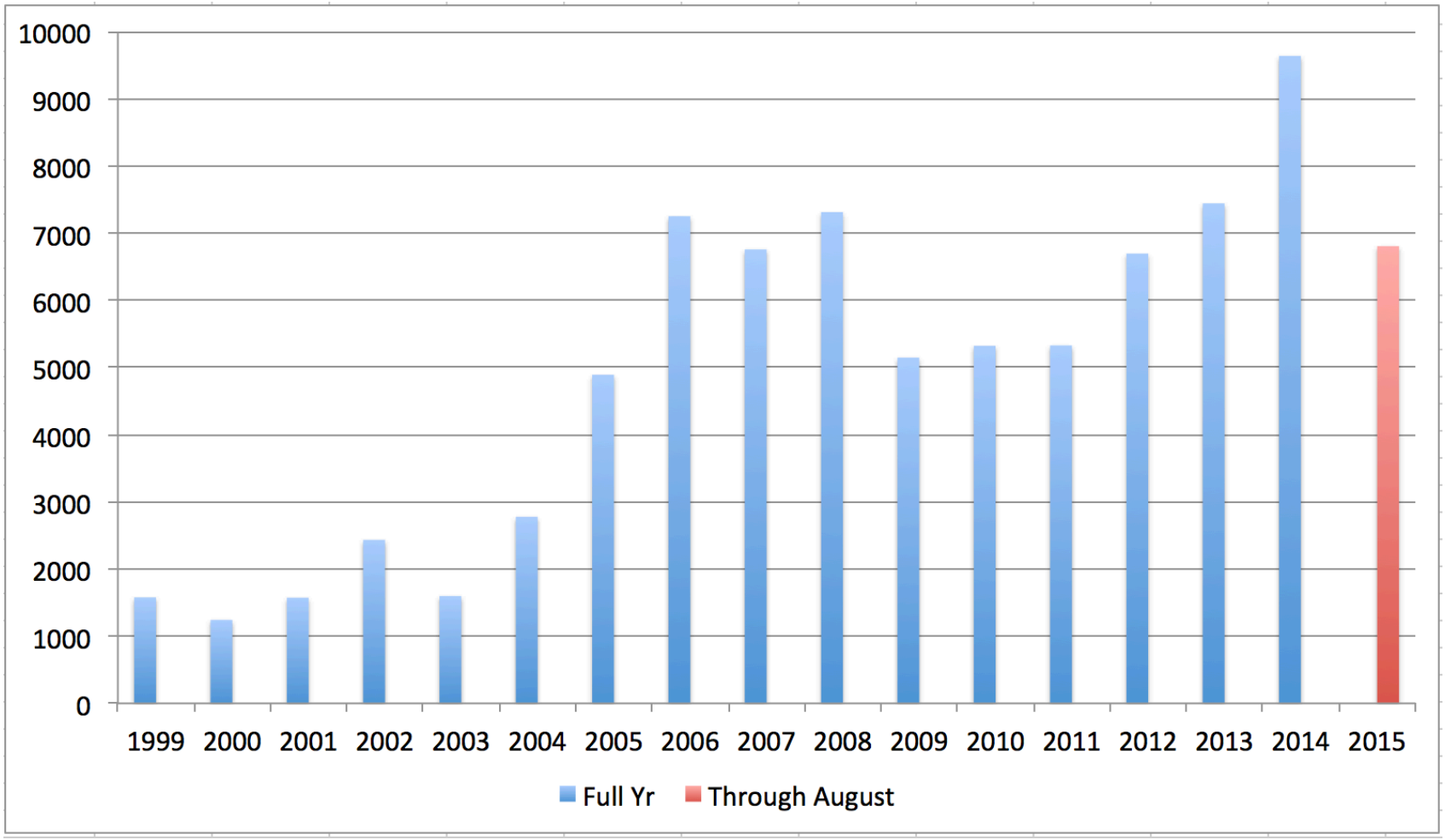
```
0:000> dds 7ffe0340 Lc
00000000`7ffe0340  77829ce9 ntdll32!LdrInitializeThunk
00000000`7ffe0344  77800100 ntdll32!KiUserExceptionDispatcher
00000000`7ffe0348  77800028 ntdll32!KiUserApcDispatcher
00000000`7ffe034c  778000b8 ntdll32!KiUserCallbackDispatcher
00000000`7ffe0350  7788f8d4 ntdll32!LdrHotPatchRoutine
00000000`7ffe0354  77822551 ntdll32!ExpInterlockedPopEntrySListFault
00000000`7ffe0358  7782251b ntdll32!ExpInterlockedPopEntrySListResume
00000000`7ffe035c  77822553 ntdll32!ExpInterlockedPopEntrySListEnd
00000000`7ffe0360  77800190 ntdll32!RtlUserThreadStart
00000000`7ffe0364  77892dfd ntdll32!RtlpQueryProcessDebugInformationRemote
00000000`7ffe0368  778517d9 ntdll32!EtwpNotificationThread
00000000`7ffe036c  777f0000 ntdll32!CsrServerApiRoutine
```

Defeating ALSR/DEP

- Getting harder – Microsoft will pay $100k for any new methods of doing it on Windows
  - http://www.microsoft.com/security/msrc/report/bypass_bounty.aspx

Bottom Line: Defenses Help

But not a panacea yet:



Problem Still Not Fully Solved

Microsoft » Windows 10 : Vulnerability Statistics

Vulnerabilities (**10**) CVSS Scores Report Browse all versions Possible matches for this product Related Metasploit Modules

Related OVAL Definitions : Vulnerabilities (0) Patches (0) Inventory Definitions (1) Compliance Definitions (0)

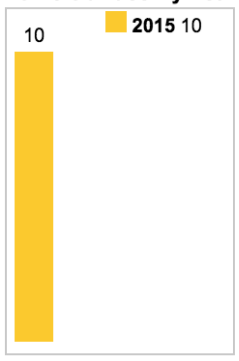
Vulnerability Feeds & Widgets

Vulnerability Trends Over Time

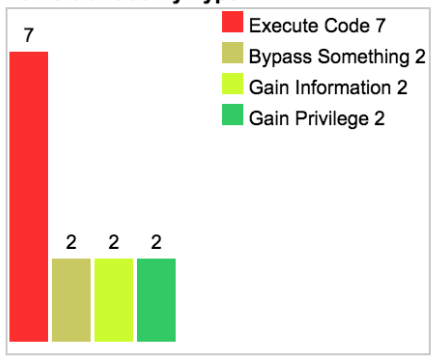
Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2015	10		7							2	2	2			
Total	10		7							2	2	2			
% Of All		0.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0	20.0	20.0	20.0	0.0	0.0	

Warning : Vulnerabilities with publish dates before 1999 are not included in this table and chart. (Because there are not many of them and they make the page look bad; and they may not be actually published in those years.)

Vulnerabilities By Year



Vulnerabilities By Type



http://www.cvedetails.com/product/32238/Microsoft-Windows-10.html?vendor_id=26

Where We Are in Syllabus

Rough Lecture Syllabus:

✔ 1. The technical nature of software vulnerabilities and techniques used for exploiting them.

✔ 2. The pressures of commercial software development, and why firms very rarely produce secure software, even though they should.

☞ 3. Basics of monitoring a network, intro/refresher on TCP/IP. Switches, wireless access devices, routers.

4. Network reconnaissance techniques – ping sweeps, port scans, etc.
5. Algorithms for detecting port scans on the network.
6. Firewalls and network segmentation as a defense against inbound attacks.
7. Detecting exploits with string matching approaches (Snort and similar).
8. Network layer approaches to evading detection.
9. Large scale attacks – worms and distributed denial of service.
10. HTTP attacks as a way around the firewall. Drive-by downloads and social engineering.
11. Defending against HTTP attacks. Web-proxies, in-browser defenses, anti-virus systems.
12. SMTP attacks – spear-phishing, and defenses against it.
13. HTTPS: Encryption and virtual private networks as a means to maintain confidentiality.
14. The modern enterprise network: what a large-scale network looks like, and emerging trends affecting it (BYOD, cloud).
15. Legal and ethical issues in defending networks.

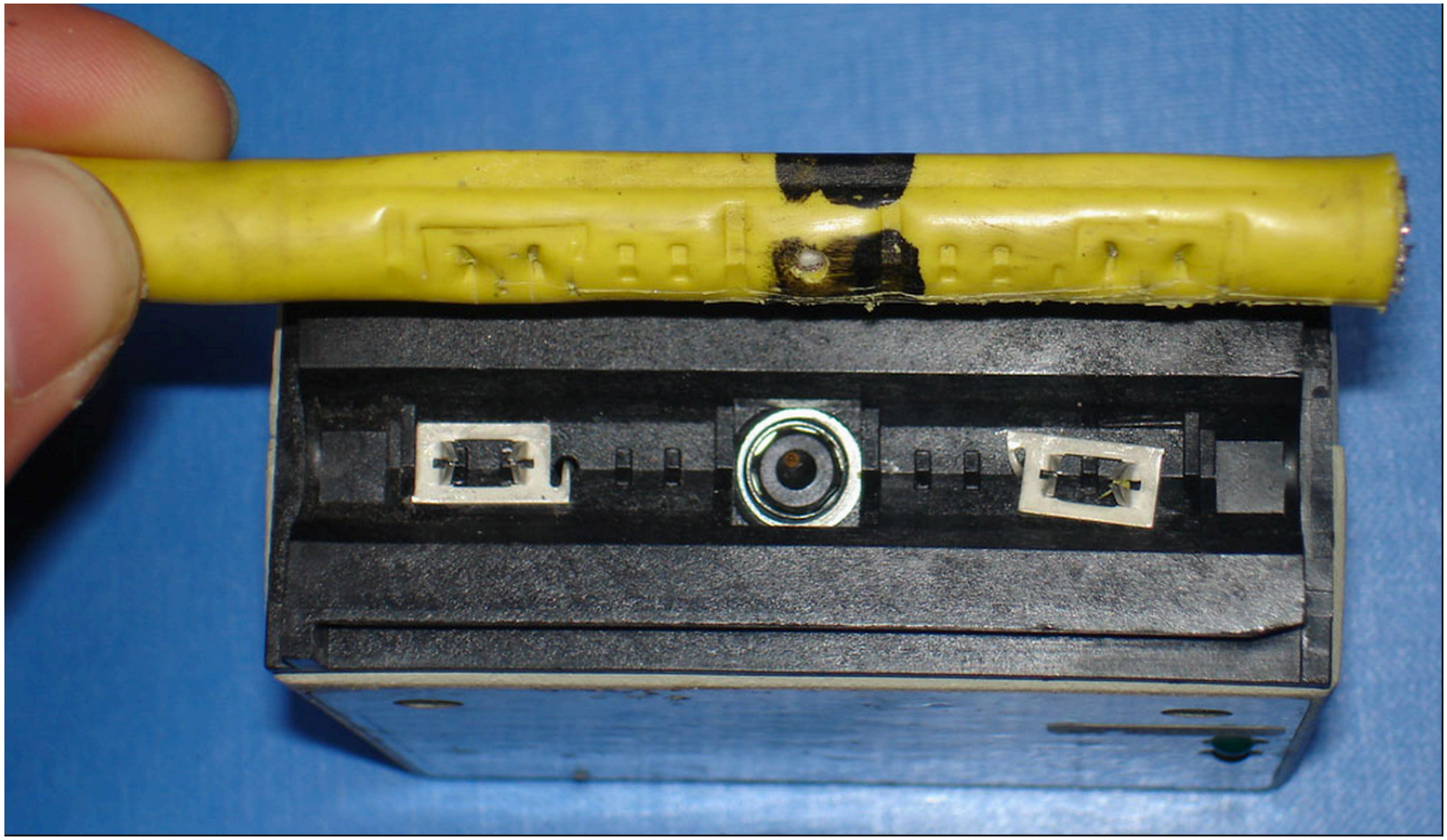
Main Goals for Today

- Basics of Ethernet networks
- Basics of IP packets
- Arp translation and arp spoofing

Ethernet Basics

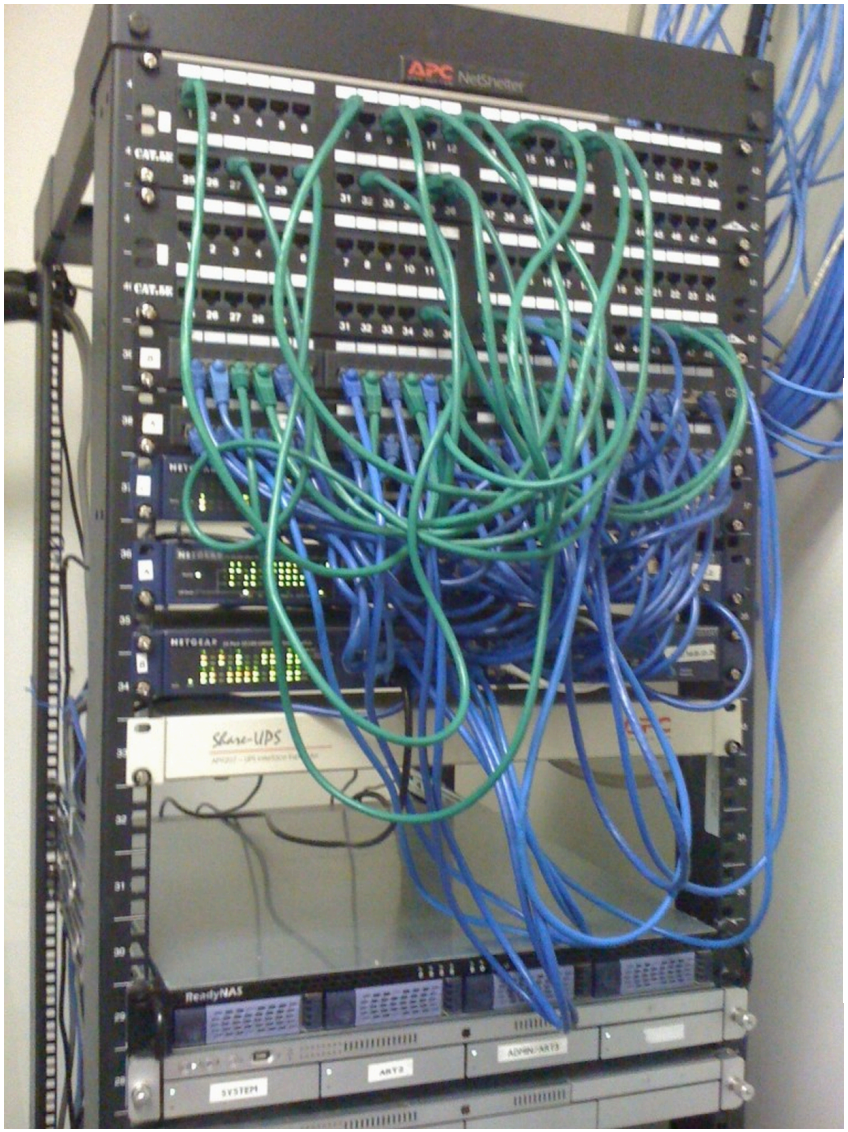
- Ethernet is a physical layer protocol/technology
  - One of many competing physical layers
  - Most popular, but others still important
    - Eg for long-haul cables
- For delivering packets of data
  - Called “frames” in ethernet lingo
  - From one machine to another
- Originally a LAN technology
  - Now sometimes used for sizeable networks

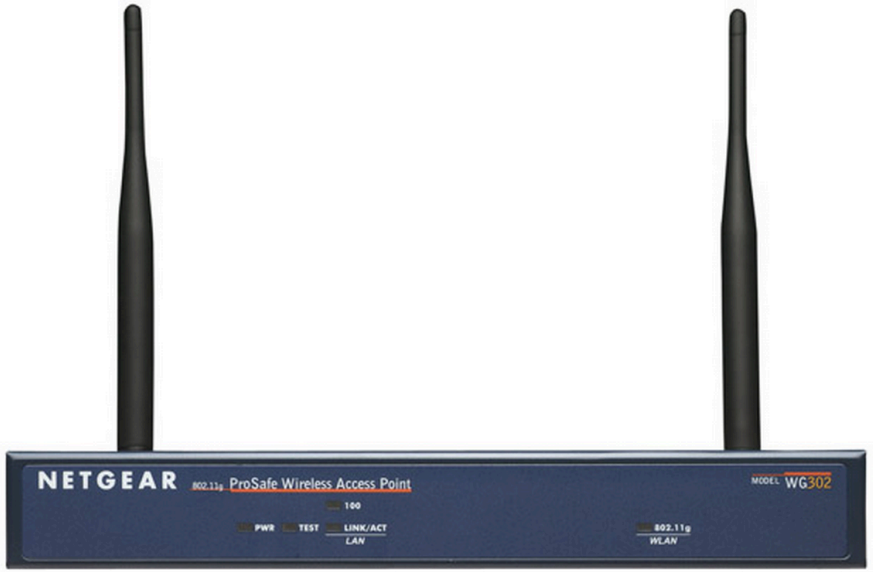
Ethernet Then



10Base5

Ethernet Now

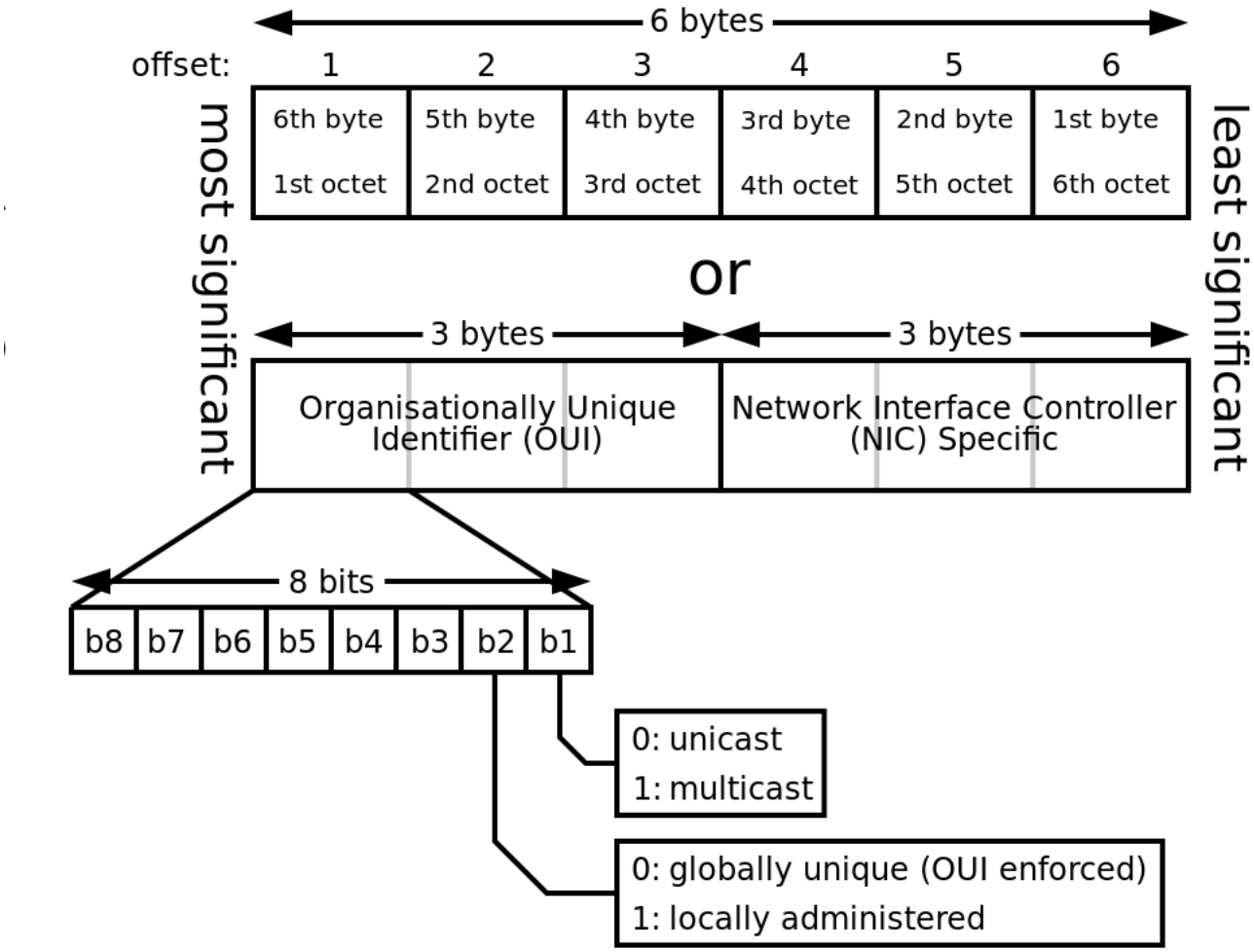




Ethernet Addresses

- 6 byte address
- ifconfig –a (alt: tcpdump –D)
- Every network interface has a hard-coded address
  - (But it's possible to forge in software…)
- Globally unique
  - Achieved by assigning vendor preambles

Ethernet address



Broadcast

- Originally broadcast – all computers hooked to the same wire
- Each interface listens to all traffic
  - Only pays attentions to packets with its address
  - Except for…

Promiscuous Mode

- Possible to put interface/OS into special mode
- Where it looks at every packet, whether or not it's addressed.
- This is the basis of network monitoring.
- Let's do it:
  - sudo tcpdump -i en0 -c 5 –e
  - sudo tcpdump -i en0 -c 5 –e not ether host 14:10:9f:e3:7d:a3

Ethernet Frame

802.3 Ethernet frame structure								
Preamble	**Start of frame delimiter**	**MAC destination**	**MAC source**	**802.1Q tag (optional)**	**Ethertype (Ethernet II) or length (IEEE 802.3)**	**Payload**	**Frame check sequence (32-bit CRC)**	**Interframe gap**
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46-1500 octets	4 octets	12 octets
		← 64–1518 octets (68-1522 octets for 802.1Q tagged frames) →						
← 84–1538 octets (88-1542 octets for 802.1Q tagged frames) →								

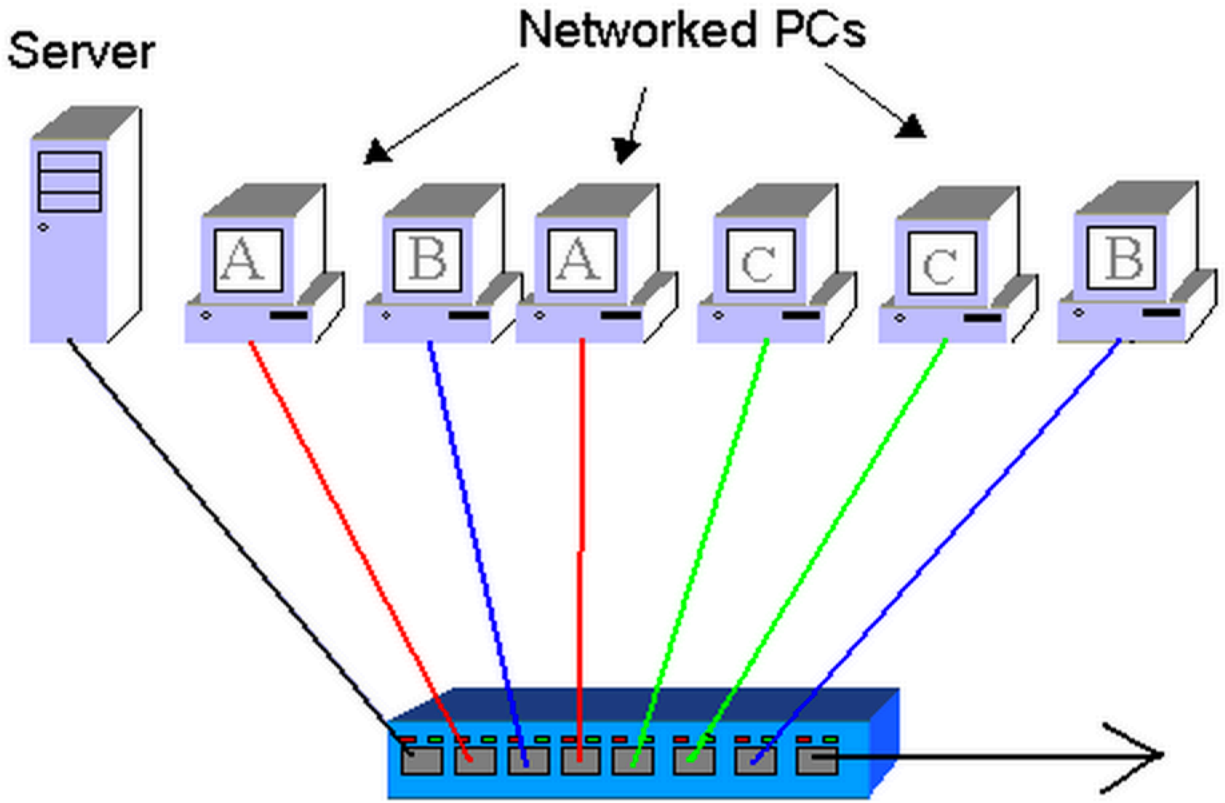
1500 is typical MTU for ethernet

Ethernet Type Codes

Note	Hex	
@	0000-05DC	IEEE802.3 Length Field (0.:1500.)
+	0101-01FF	Experimental
	0200	Xerox PUP (conflicts with 802.3 Length Field range) (see 0A00)
	0201	Xerox PUP Address Translation (conflicts ...) (see 0A01)
	0400	Nixdorf (conflicts with 802.3 Length Field)
+*	0600	Xerox NS IDP
	0601	XNS Address Translation (3Mb only)
+*	0800	DOD Internet Protocol (IP)
+	0801	X.75 Internet
+	0802	NBS Internet
+	0803	ECMA Internet
+	0804	CHAOSnet
+	0805	X.25 Level 3
+*	0806	Address Resolution Protocol (ARP) (for IP and for CHAOS)
	0807	XNS Compatibility
	081C	Symbolics Private
+	0888-088A	Xyplex
	0900	Ungermann-Bass network debugger
	0A00	Xerox IEEE802.3 PUP
	0A01	Xerox IEEE802.3 PUP Address Translation
	0BAD	Banyan Systems
	0BAF	Banyon VINES Echo
	1000	Berkeley Trailer negotiation
	1001-100F	Berkeley Trailer encapsulation for IP
	1234	DCA - Multicast
*	1600	VALID system protocol

http://www.cavebear.com/archive/cavebear/Ethernet/type.html

Switched Ethernet



Source: http://engweb.info/courses/various/gnotes/ethernet_overview.html