Exploiting and Defense
Technical Intro

Dobin Rutishauser

January 2017

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Content

Kinda Relevant

- Code vs. Data
- Vulnerability type: Memory corruptions
- What is an exploit? What types exist?
- What is vulnerable?
- Code and Data: Who is on top? (+Weird machines)

Philosophy and History

- Is exploit writing hacking?
- Morris worm

Code and Data

Difference between Data and Code (spoiler alert: there is none)

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

What is a picture?

What is a picture? In an text editor

What is a picture? In an hex editor

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01  ÿØÿà..JFIF.....
00000010   00 01 00 00 FF DB 00 84 00 05 03 04 09 09 09 08  ....ÿÛ.„........
00000020   08 08 08 06 08 08 08 06 07 07 07 08 07 07 07 07  ................
00000030   07 07 07 07 07 07 07 07 07 07 07 07 0A 10 0B 07  ................
00000040   08 0E 09 07 07 0C 15 0C 0E 11 11 13 13 13 07 0B  ................
00000050   16 18 16 12 18 10 12 13 12 01 05 05 05 08 07 08  ................
00000060   0C 07 07 0C 12 08 07 08 12 12 12 12 12 12 12 12  ................
00000070   12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  ................
00000080   12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  ................
00000090   12 12 12 1E 12 12 12 1E 12 12 FF C0 00 11 08 04  ..........ÿÀ....
000000A0   B0 06 40 03 01 22 00 02 11 01 03 11 01 FF C4 00  °.@.."........ÿÄ.
000000B0   1D 00 00 02 03 01 01 01 01 01 00 00 00 00 00 00  ................
000000C0   00 00 02 03 00 01 04 05 06 07 08 09 FF C4 00 43  ............ÿÄ.C
000000D0   10 00 02 02 01 03 02 05 03 03 03 03 03 03 02 02  ................
000000E0   0B 01 02 00 03 11 04 12 21 05 31 06 13 22 32 41  ........!.1.."2A
000000F0   42 51 52 07 14 62 23 61 72 15 33 82 43 71 92 16  BQR..b#ar.3,Cq'.
00000100   24 53 08 81 A2 34 B2 C2 44 63 91 17 25 73 83 E2  $S..¢4²ÂDc'.%sƒâ
00000110   54 FF C4 00 1B 01 00 03 01 01 01 01 01 00 00 00  TÿÄ.............
00000120   00 00 00 00 00 00 00 01 02 03 04 05 06 07 FF C4  ..............ÿÄ
00000130   00 29 11 01 01 00 02 02 03 01 00 02 03 01 01 00  .)..............
00000140   02 03 00 00 01 02 11 03 12 13 21 31 04 14 41 05  ..........!1..A.
00000150   22 51 61 32 71 91 06 52 62 FF DA 00 0C 03 01 00  "Qa2q'.RbÿÚ.....
00000160   02 11 03 11 00 3F 00 FC 73 17 19 17 05 24 9F 12  .....?.üs....$Ÿ.
00000170   49 F1 00 A9 72 A5 FC 40 AA 49 24 90 24 92 49 20  Iñ.©r¥ü@ªI$.$'I
00000180   12 49 24 80 4F 89 52 FE 25 40 21 92 43 24 02 47  .I$€O‰Rþ%@!'C$.G
```

JFIF file structure		
Segment	**Code**	**Description**
SOI	`FF D8`	Start of Image
JFIF-APP0	`FF E0 s1 s2 4A 46 49 46 00 ...`	see below
JFXX-APP0	`FF E0 s1 s2 4`	
... additional marker segments (for example SOF, DHT, COM)		
SOS	`FF DA`	
	compressed ima	
EOI	`FF D9`	

JFIF APP0 marker segment [edit]

In the mandatory JFIF APP0 marker segment the parameters of the image are specified. Optionally an uncompresse

JFIF APP0 marker segment		
Field	**Size (bytes)**	**Description**
APP0 marker	2	`FF E0`
Length	2	Length of segment excluding APP0 marker
Identifier	5	`4A 46 49 46 00` = "JFIF" in ASCII, terminated by a null byte
JFIF version	2	First byte for major version, second byte for minor version (`01 02` for 1.02)
Density units	1	Units for the following pixel density fields • `00` : No units; width:height pixel aspect ratio = Xdensity:Ydensity • `01` : Pixels per inch (2.54 cm) • `02` : Pixels per centimeter
Xdensity	2	Horizontal pixel density. Must not be zero.
Ydensity	2	Vertical pixel density. Must not be zero.
Xthumbnail	1	Horizontal pixel count of the following embedded RGB thumbnail. May be zero.
Ythumbnail	1	Vertical pixel count of the following embedded RGB thumbnail. May be zero.

maxresdefault.jpg - JPEGsnoop

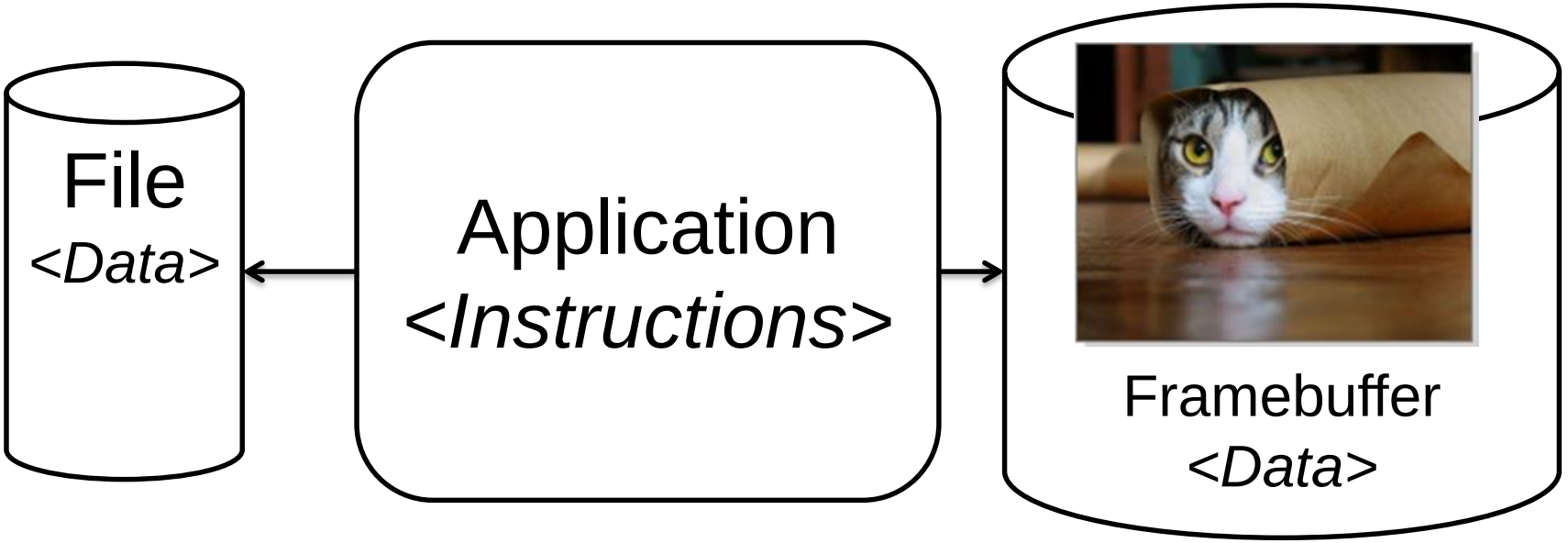
File Edit View Tools Options Help

```
  Filesize: [136356] Bytes

Start Offset: 0x00000000
*** Marker: SOI (xFFD8) ***
  OFFSET: 0x00000000

*** Marker: APP0 (xFFE0) ***
  OFFSET: 0x00000002
  Length     = 16
  Identifier = [JFIF]
  version    = [1.1]
  density    = 1 x 1 (aspect ratio)
  thumbnail  = 0 x 0

*** Marker: DQT (xFFDB) ***
  Define a Quantization Table.
  OFFSET: 0x00000014
  Table length = 132
  ----
  Precision=8 bits
  Destination ID=0 (Luminance)
    DQT, Row #0:   5   3   9   8   6   7   7   7
    DQT, Row #1:   4   9   8   8   7   7   7   7
    DQT, Row #2:   9   8   8   7   7   7   9   7
    DQT, Row #3:   8   8   8   7   7  14  12   7
    DQT, Row #4:   6   7   7   7   8  21  19  11
    DQT, Row #5:   7   7   7   7  12  19  22  16
    DQT, Row #6:   7   7  11  14  19  24  24  18
```

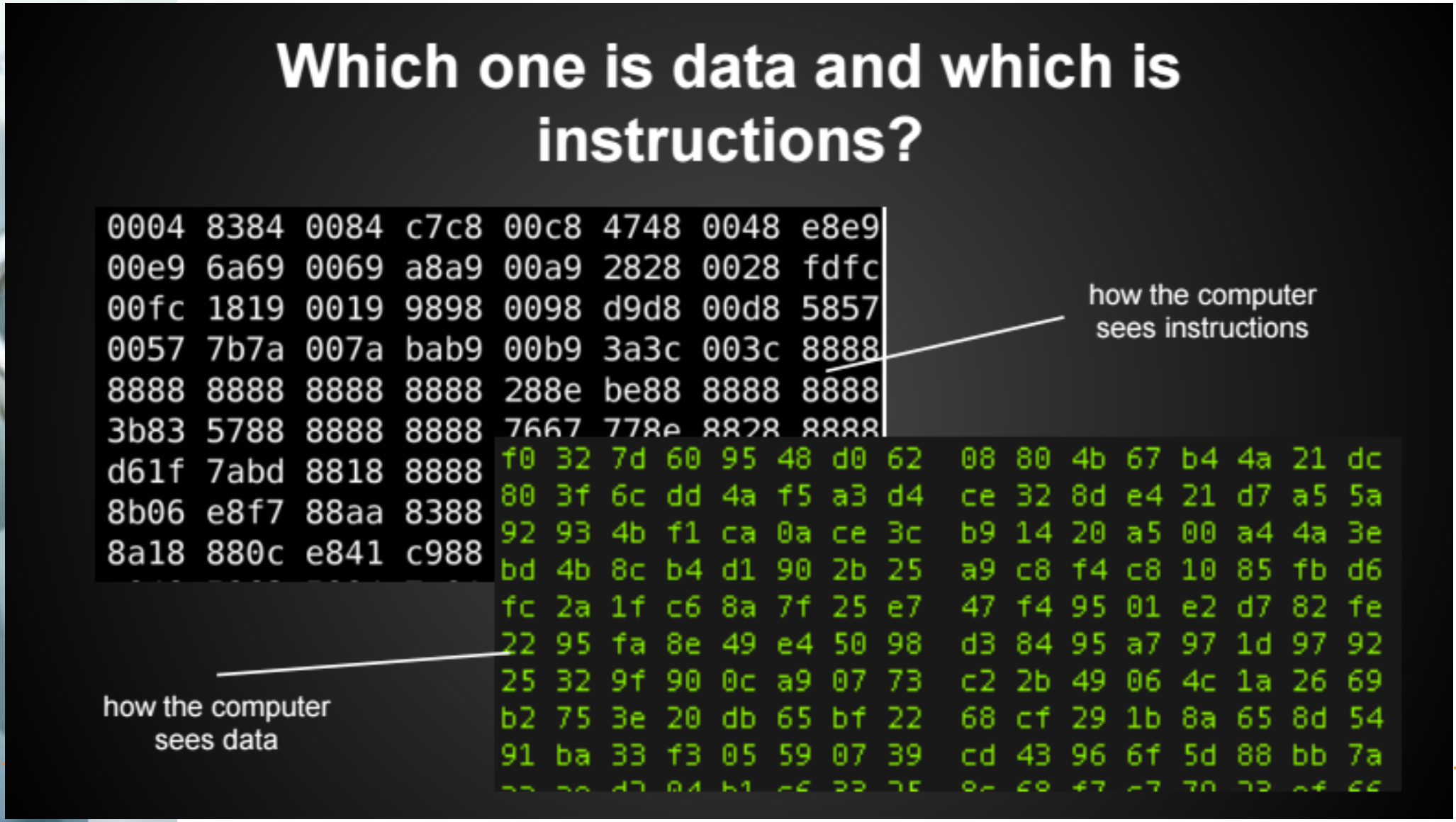
Image (RGB, DC) @ 12.5% (1/8)

What is a picture?

- Data for the computer
- When interpreted correctly, displays a cat
- When interpreted wrongly, displays garbage / crashes
- When interpreted wrongly in the right way, lets us hack a computer



What is a picture?



What is a picture?

Is it possible to create an image which executes code?

- **Yes**
- If this is intentional, it's a feature
- If this is not intentional, the picture is an exploit (exploiting a bug/vulnerability)

How?!

- Make the original program (code) execute our (the attackers) own code (data) by writing into memory locations at runtime which influence where code is being read from

Vulnerability types

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Vulnerability types

Vulnerability types

- **Memory corruption**
- Authentication
- Authorization
- Configuration error
- Input validation
- Logic error
- Sensitive data protection
- Session management
- Encoding Error
- Cryptographic Errors
- Permission Problems
- …

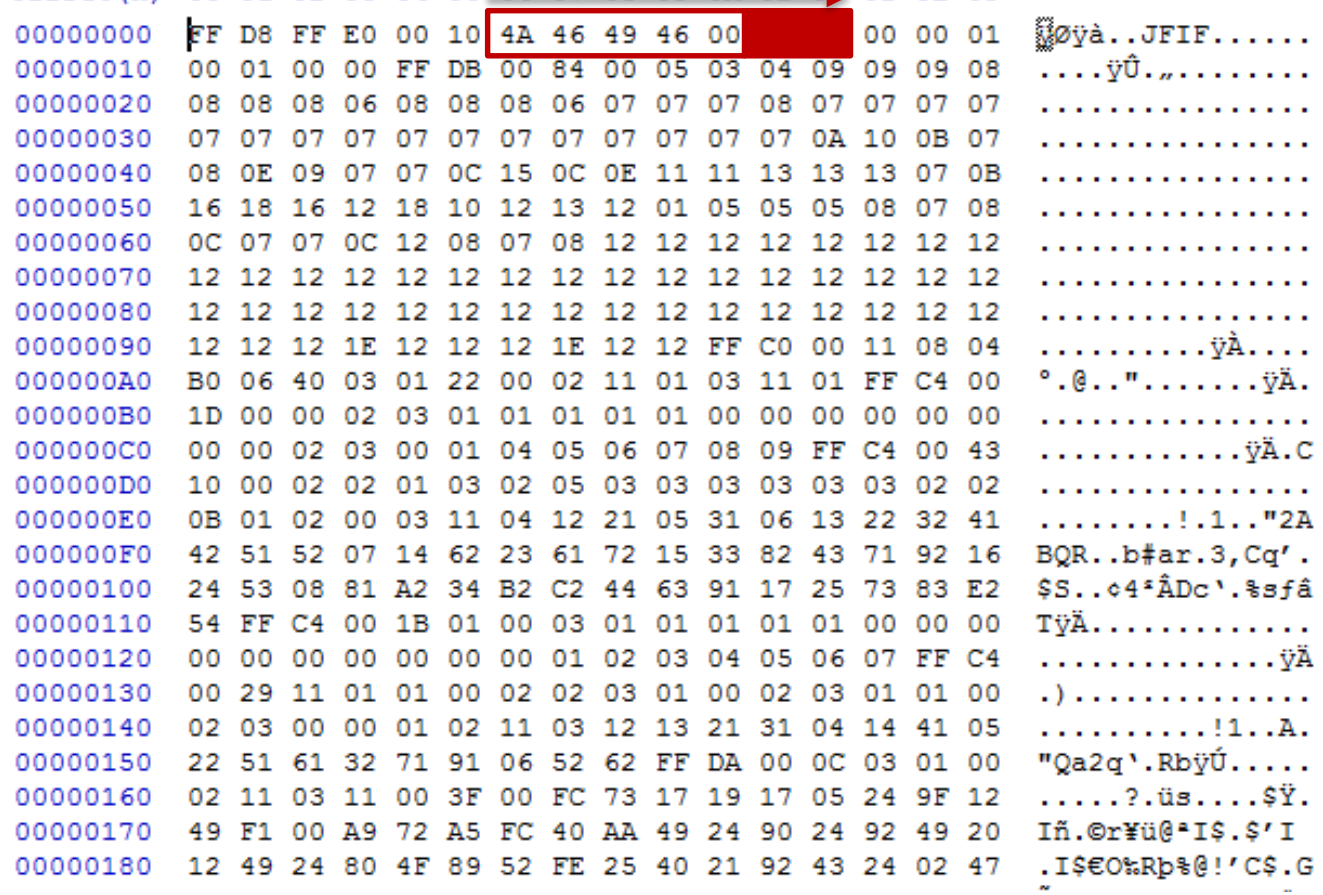
Vulnerability Types

Memory corruption occurs in a computer program when the **contents of a memory location are unintentionally modified** due to programming errors; this is termed violating memory safety. When the corrupted memory contents are used later in that program, **it leads either to program crash or to strange and bizarre program behavior**

Modern programming languages like C and C++ have powerful features of explicit memory management and pointer arithmetic. These features are designed for developing “efficient” applications and system software.

https://en.wikipedia.org/wiki/Memory_corruption

Vulnerability Types – Memory Corruption



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   FF D8 FF E0 00 10 4A 46 49 46 00 [redacted] 00 00 01   ÿØÿà..JFIF.....
00000010   00 01 00 00 FF DB 00 84 00 05 03 04 09 09 09 08   ....ÿÛ.„........
00000020   08 08 08 06 08 08 08 06 07 07 07 08 07 07 07 07   ................
00000030   07 07 07 07 07 07 07 07 07 07 07 07 0A 10 0B 07   ................
00000040   08 0E 09 07 07 0C 15 0C 0E 11 11 13 13 13 07 0B   ................
00000050   16 18 16 12 18 10 12 13 12 01 05 05 05 08 07 08   ................
00000060   0C 07 07 0C 12 08 07 08 12 12 12 12 12 12 12 12   ................
00000070   12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12   ................
00000080   12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12   ................
00000090   12 12 12 1E 12 12 12 1E 12 12 FF C0 00 11 08 04   ..........ÿÀ....
000000A0   B0 06 40 03 01 22 00 02 11 01 03 11 01 FF C4 00   °.@.."........ÿÄ.
000000B0   1D 00 00 02 03 01 01 01 01 01 00 00 00 00 00 00   ................
000000C0   00 00 02 03 00 01 04 05 06 07 08 09 FF C4 00 43   ............ÿÄ.C
000000D0   10 00 02 02 01 03 02 05 03 03 03 03 03 03 02 02   ................
000000E0   0B 01 02 00 03 11 04 12 21 05 31 06 13 22 32 41   ........!.1.."2A
000000F0   42 51 52 07 14 62 23 61 72 15 33 82 43 71 92 16   BQR..b#ar.3,Cq'.
00000100   24 53 08 81 A2 34 B2 C2 44 63 91 17 25 73 83 E2   $S..¢4²ÂDc'.%sƒâ
00000110   54 FF C4 00 1B 01 00 03 01 01 01 01 01 00 00 00   TÿÄ.............
00000120   00 00 00 00 00 00 00 01 02 03 04 05 06 07 FF C4   ..............ÿÄ
00000130   00 29 11 01 01 00 02 02 03 01 00 02 03 01 01 00   .)..............
00000140   02 03 00 00 01 02 11 03 12 13 21 31 04 14 41 05   ..........!1..A.
00000150   22 51 61 32 71 91 06 52 62 FF DA 00 0C 03 01 00   "Qa2q'.RbÿÚ.....
00000160   02 11 03 11 00 3F 00 FC 73 17 19 17 05 24 9F 12   .....?.üs....$Ÿ.
00000170   49 F1 00 A9 72 A5 FC 40 AA 49 24 90 24 92 49 20   Iñ.©r¥ü@ªI$.$'I
00000180   12 49 24 80 4F 89 52 FE 25 40 21 92 43 24 02 47   .I$€O‰Rþ%@!'C$.G
```

What is an exploit?

Formal definition

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Simple Definition of EXPLOIT

- to get value or use from (something)
- **to use (someone or something) in a way that helps you unfairly**

Full Definition of EXPLOIT

- to make productive use of : UTILIZE
  <*exploiting* your talents> <*exploit* your opponent's weakness>
- to make use of meanly or unfairly for one's own advantage
  <*exploiting* migrant farm workers>

http://www.merriam-webster.com/dictionary/exploit

What is an exploit? Hacking related

to exploit (v): To take advantage of a vulnerability so that the target system reacts in a manner other than which the designer intended.

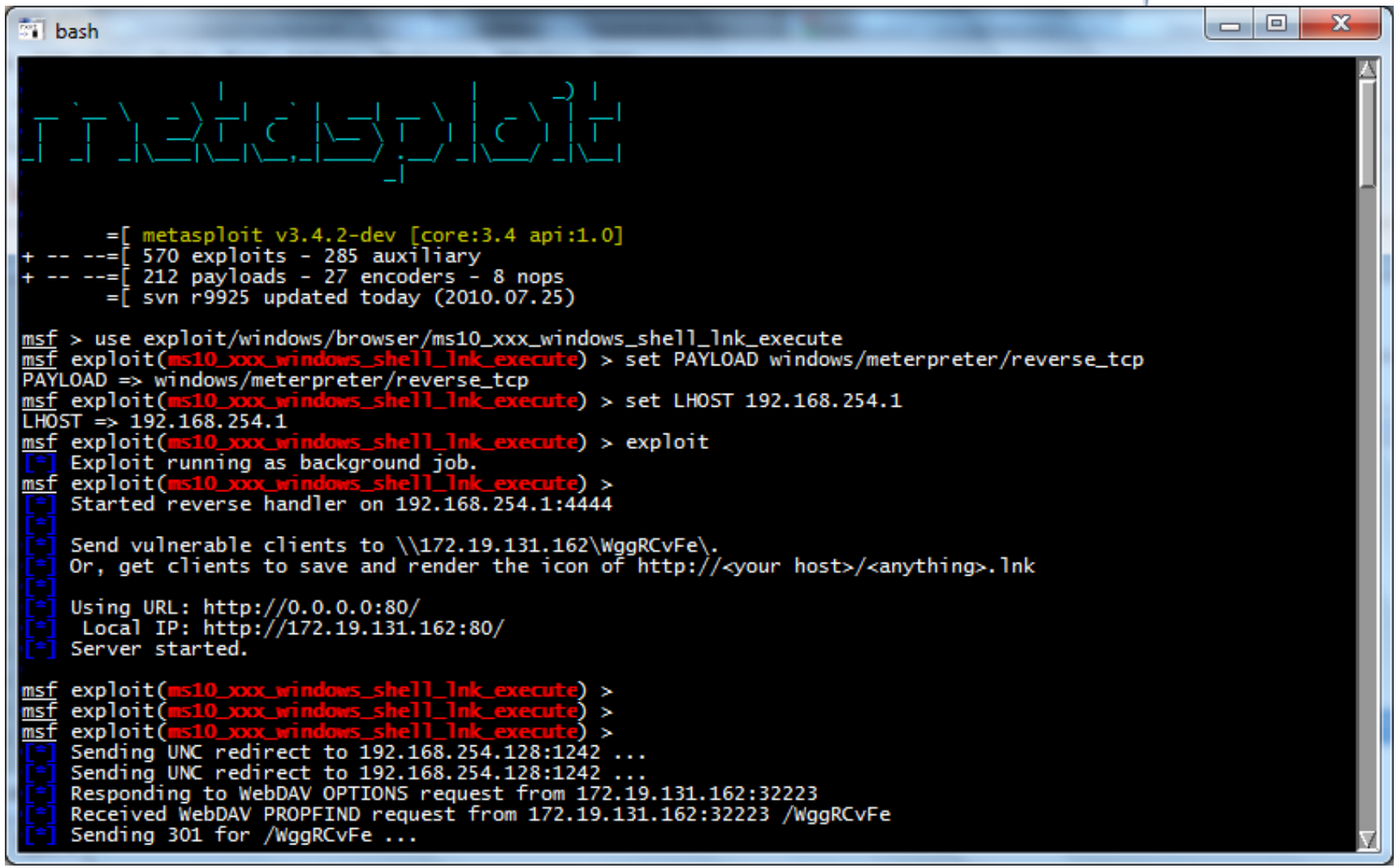
the Exploit (n): The tool, set of instructions, or code that is used to take advantage of a vulnerability.

(The Shellcoders Handbook, 2nd Edition, p4)

What is an exploit?

```
/*
**
** wu-ftpd v2.6.2 off-by-one remote 0day exploit.
**
** exploit by "you dong-hun"(Xpl017Elz), <szoahc@hotmail.com>.
**
** Update:
**         [v0.0.2] August 2, I added wu-ftpd-2.6.2, 2.6.0, 2.6.1 finally.
**         [v0.0.3] August 3, Brute-Force function addition.
**         [v0.0.4] August 4, Added FreeBSD, OpenBSD version wu-ftpd-2.6.x exploit.
**                                    It will be applied well to most XxxxBSD.
**         [v0.0.5] August 4, Remote scan & exploit test function addition.
**                     August 6, Cleaning.
**
*/

#define VERSION "v0.0.5"
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define DEBUG_NG
#undef DEBUG_NG
#define NRL 0
#define SCS 1
#define FAD (-1)
#define MAX_BF (16)
#define BF_LSZ (0x100) /* 256 */
#define DEF_VA 255
#define DEF_PORT 21
#define DEF_ANSH_LINUX 15
#define DEF_ANSH_FRBSD 55
#define GET_HOST_NM_ERR (NULL)
#define SIN_ZR_SIZE 8
#define DEF_ALIGN 4
```

What is an exploit?



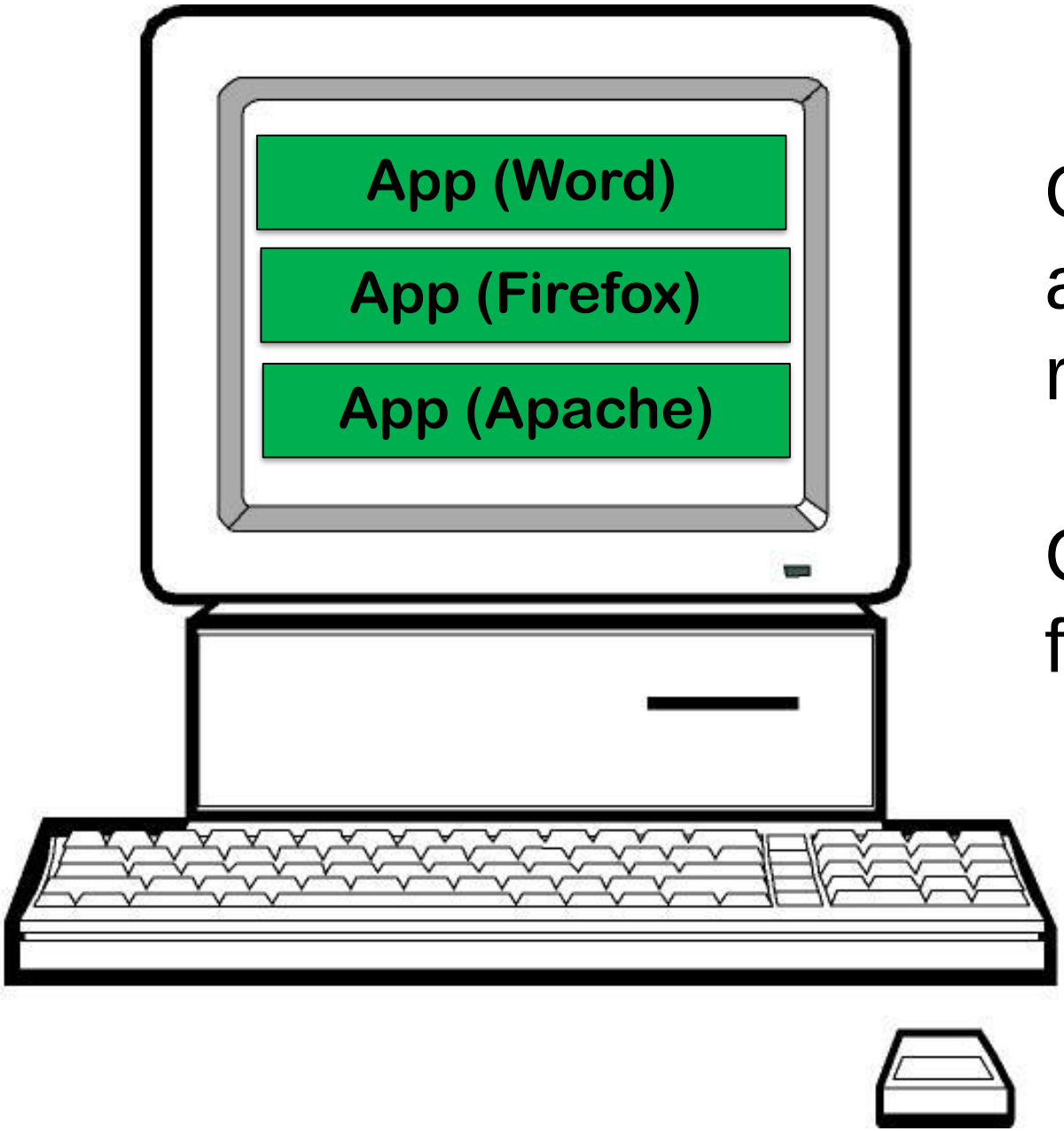
Types of exploits?

Local, remote, client-side exploits

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

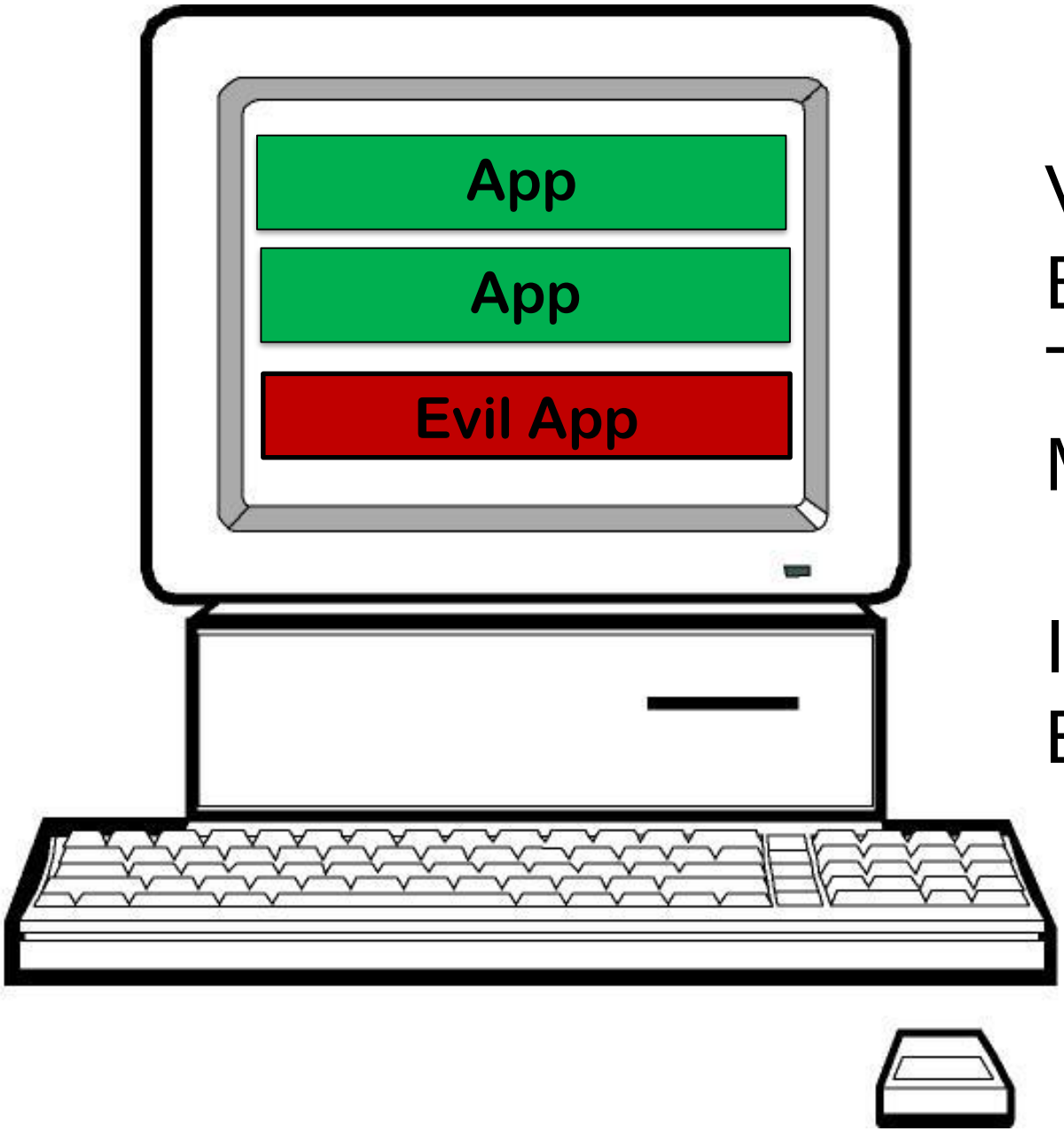
What is an exploit?



Only trusted apps (code) running

Computer fully secure

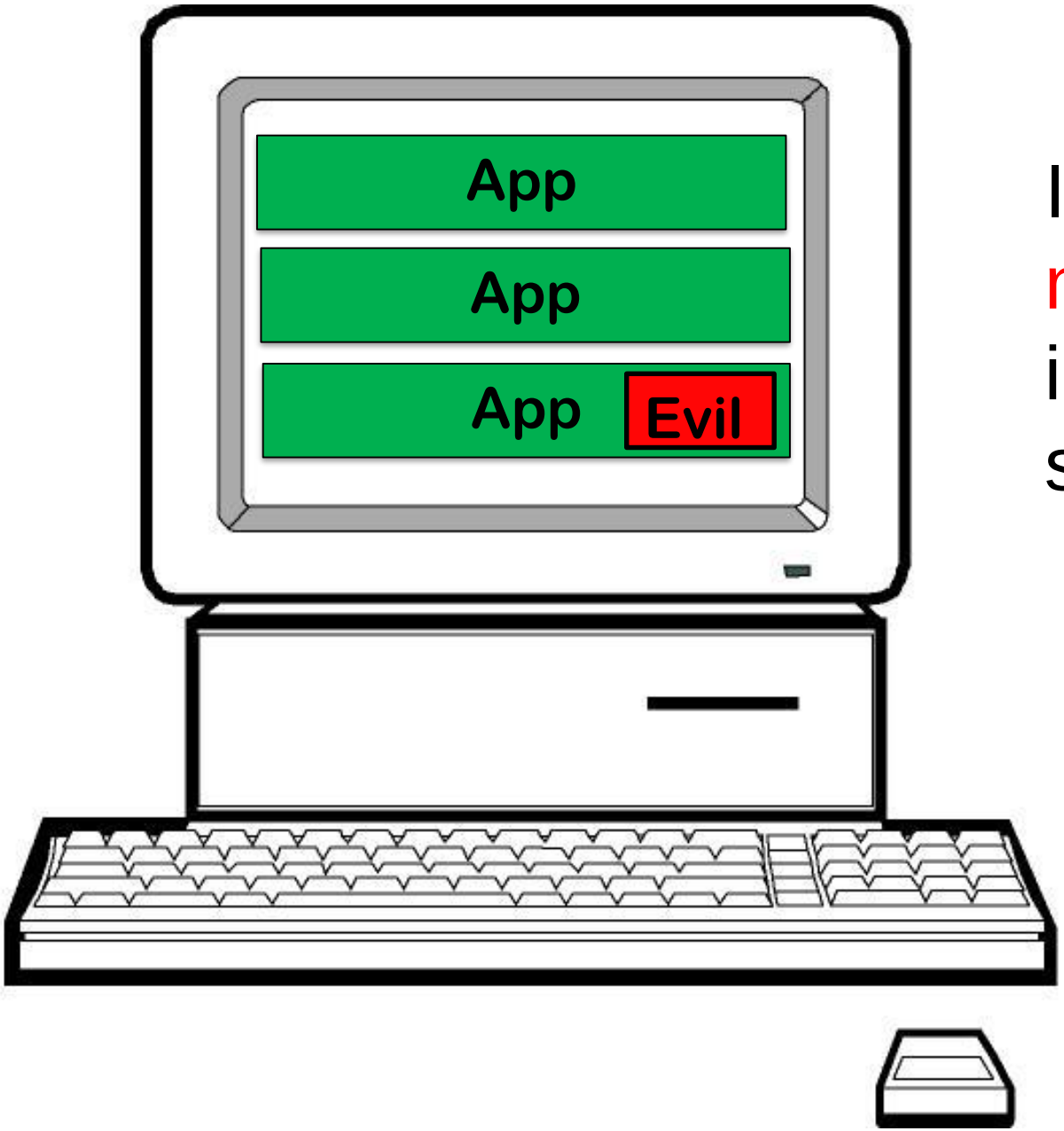
What is an exploit?



Virus /
Backdoor /
Trojan /
Malware

Is **NOT**
Exploit

What is an exploit?



Introduce
new code
into running
software

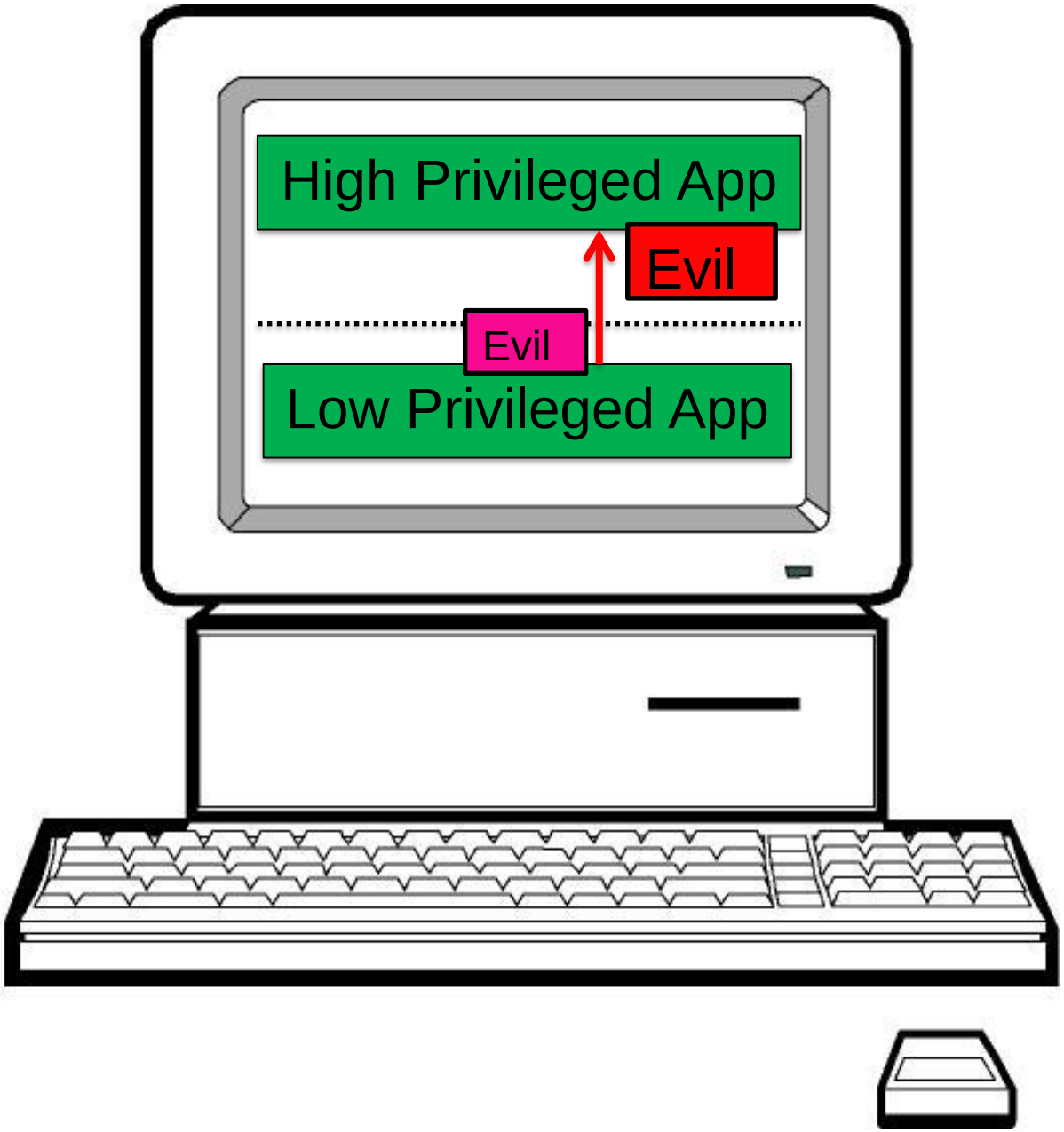
Types of exploits

Local

Server-side

Client-side

Types of exploits - **Local** exploit



Types of exploits - local

Local Exploit:

- Attacker is already on a host
- Wants to execute his code with higher privileges
- “Privilege Escalation”

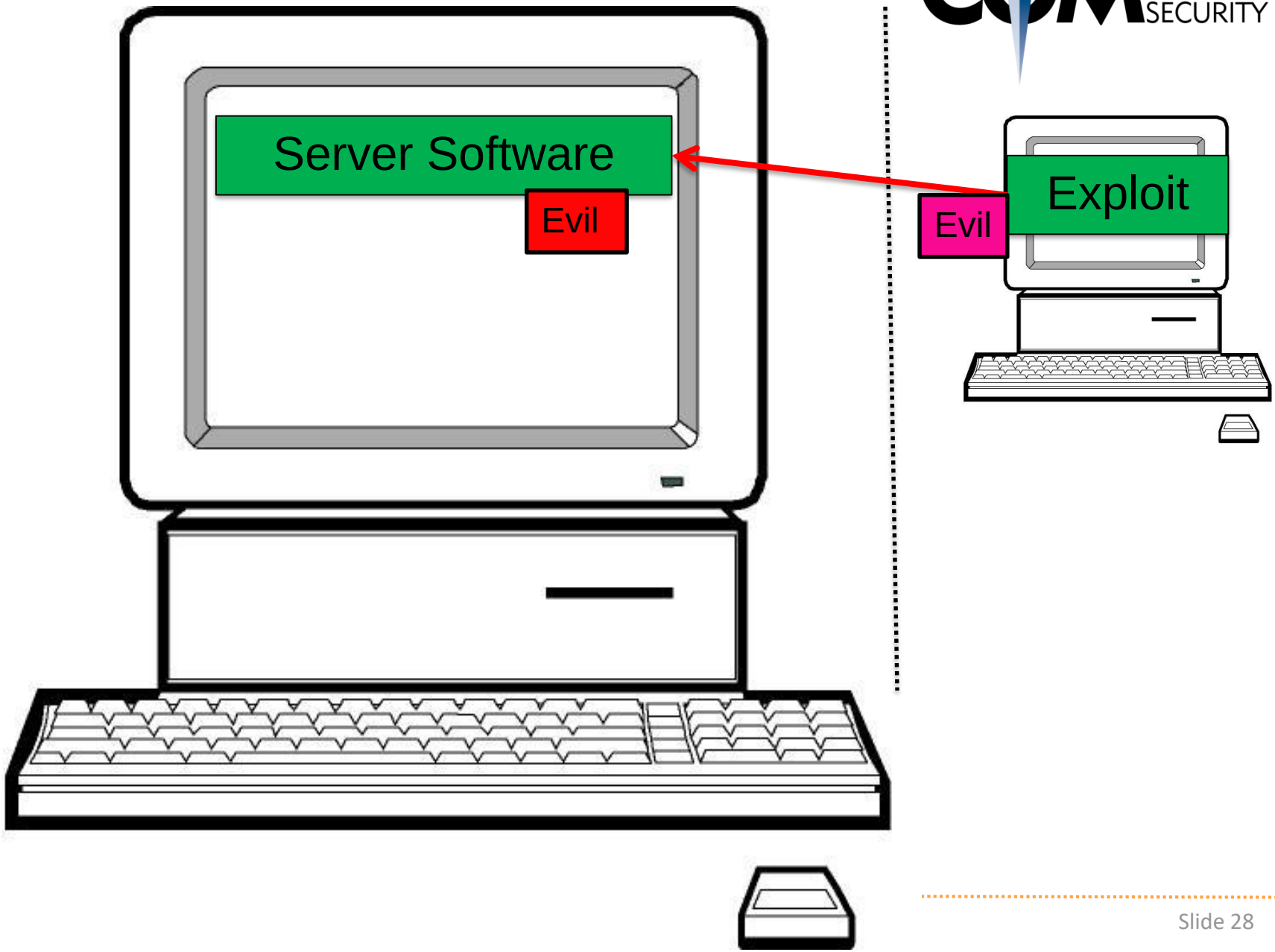
Linux:

- User -> root
- E.g.: www-data -> root

Windows:

- User -> Local Admin (-> System)

Types of exploits - **Server-side** exploit



Types of exploits - Remote

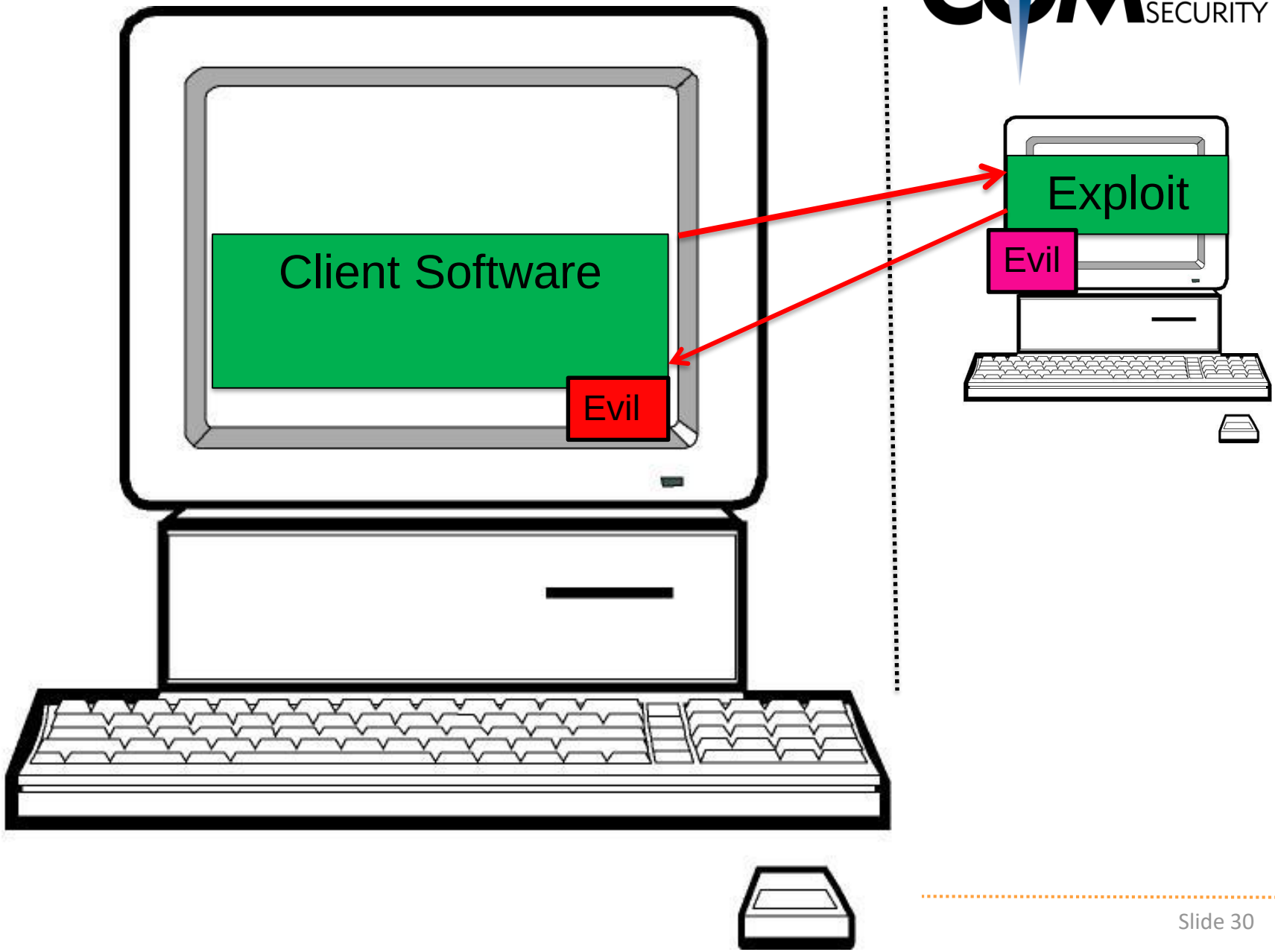
Remote Exploit:

- Attacker can directly talk with a server software on a host
- Wants to execute his code on the remote host

Server Examples

- FTP Server (proftp, wuftp)
- DNS Server (bind)
- Web Server (IIS, Apache)

Types of exploits - **Client-side** exploit



Types of exploits - Client

Client Exploit:

- Attacker can influence data which a client receives
- Wants to execute his code on the client host

Examples:

- Browser
  - Flash
  - Java
  - Image Viewer
- Word
- Putty
- Git
- VLC

What is vulnerable against memory corruption?

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

What is vulnerable?

What software is affected?

Software developed in unsafe programming languages

- (ASM)
- C
- C++
- Fortran (lol)

What is vulnerable?

What software is affected?

Software developed in unsafe programming languages

- (ASM)
- C
- C++
- Fortran (lol)

Who writes software in C/C++, anyway?

- IE, Chrome, Firefox
- Apache / IIS
- Postfix, Sendmail
- BIND
- MS Office / LibreOffice
- Antivirus
- Other “Security” Software

What is vulnerable?

Not affected:

Software written in interpreted languages

- PHP
- Perl
- Ruby
- Bash
- Python
- JavaScript

Languages with strict bound checking

- Rust
- C#
- Java
- Go

Exception: Native calls

What is vulnerable?

Special case: Interpreter itself

What language is PHP, Java, JavaScript, … written-in?

- C / C++ !

Some memory corruption vuln's

From 2015

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

What is vulnerable?

Vulnerability Details : CVE-2015-8617

Format string vulnerability in the zend_throw_or_error function in Zend/zend_execute_API.c in PHP 7.x before 7.0.1 allows remote attackers to execute arbitrary code via format string specifiers in a string that is misused as a class name, leading to incorrect error handling.

Publish Date : 2016-01-19 Last Update Date : 2016-01-21

Collapse All Expand All Select Select&Copy Scroll To Comments External Links

Search Twitter Search YouTube Search Google

– CVSS Scores & Vulnerability Types

CVSS Score	10.0
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code
CWE ID	134

Some memory corruption bugs

CVE-2015-6094

- Microsoft Office is prone to a remote memory-corruption vulnerability because it fails to properly handle objects in memory.
- **Excel 2010, 2013, 2016**

CVE-2015-6068

- Microsoft Internet Explorer is prone to a remote memory-corruption vulnerability. Attackers can exploit this issue by enticing an unsuspecting user to view a specially crafted webpage.
- **IE11**

CVE-2015-5122

- Use-after-free vulnerability in the DisplayObject class in the ActionScript 3 (AS3) implementation in Adobe Flash Player 13.x through 13.0.0.302 on Windows and OS X, 14.x through 18.0.0.203 on Windows and OS X, 11.x through 11.2.202.481 on Linux, and 12.x through 18.0.0.204 on Linux Chrome installations allows remote attackers to execute arbitrary code
- **Flash 11, 12, 13, 14**

Some memory corruption bugs

CVE-2015-0287

- ASN.1 structure reuse memory corruption. Reusing a structure in ASN.1 parsing may allow an attacker to cause memory corruption via an invalid write.
- **OpenSSL 0.9.8-1.0.2**

CVE-2015-7852

- A potential off by one vulnerability exists in the cookedprint functionality of ntpq. A specially crafted buffer could cause a buffer overflow potentially resulting in null byte being written out of bounds.
- **NTP 4.2.8p2**

CVE-2015-1538 (Stagefright)

- Integer overflow in the SampleTable::setSampleToChunkParams function in SampleTable.cpp in libstagefright in Android before 5.1.1 LMY48I allows remote attackers to execute arbitrary code via crafted atoms in MP4 data that trigger an unchecked multiplication
- **Android 1.5 - 5.1**

Some memory corruption bugs

Android:

- Overview
- Bulletins
  - Advisories
  - April 2016
  - March 2016
  - February 2016
  - January 2016
  - December 2015
  - November 2015
  - October 2015
  - September 2015
  - August 2015
- Authentication
- Keystore

Elevation of Privilege Vulnerability in Telecom Component	CVE-2016-0847	High
Elevation of Privilege Vulnerability in Download Manager	CVE-2016-0848	High
Elevation of Privilege Vulnerability in Recovery Procedure	CVE-2016-0849	High
Elevation of Privilege Vulnerability in Bluetooth	CVE-2016-0850	High
Elevation of Privilege Vulnerability in Texas Instruments Haptic Driver	CVE-2016-2409	High
Elevation of Privilege Vulnerability in a Video Kernel Driver	CVE-2016-2410	High
Elevation of Privilege Vulnerability in Qualcomm Power Management Component	CVE-2016-2411	High
Elevation of Privilege Vulnerability in System_server	CVE-2016-2412	High
Elevation of Privilege Vulnerability in Mediaserver	CVE-2016-2413	High
Denial of Service Vulnerability in Minikin	CVE-2016-2414	High
Information Disclosure Vulnerability in Exchange ActiveSync	CVE-2016-2415	High
Information Disclosure Vulnerability in Mediaserver	CVE-2016-2416 CVE-2016-2417 CVE-2016-2418 CVE-2016-2419	High
Elevation of Privilege Vulnerability in Debuggerd Component	CVE-2016-2420	Moderate
Elevation of Privilege Vulnerability in Setup Wizard	CVE-2016-2421	Moderate
Elevation of Privilege Vulnerability in Wi-Fi	CVE-2016-2422	Moderate
Elevation of Privilege Vulnerability in Telephony	CVE-2016-2423	Moderate
Denial of Service Vulnerability in SyncStorageEngine	CVE-2016-2424	Moderate
Information Disclosure Vulnerability in AOSP Mail	CVE-2016-2425	Moderate
Information Disclosure Vulnerability in Framework	CVE-2016-2426	Moderate
Information Disclosure Vulnerability in BouncyCastle	CVE-2016-2427	Moderate

Some memory corruption bugs

Firefox:

Fixed in Firefox 45

- 2016-38 Out-of-bounds write with malicious font in Graphite 2
- 2016-27 Use-after-free during XML transformations
- 2016-26 Memory corruption when modifying a file being read by FileReader
- 2016-19 Linux video memory DOS with Intel drivers
- 2016-18 CSP reports fail to strip location information for embedded iframe pages
- 2016-17 Local file overwriting and potential privilege escalation through CSP reports
- 2016-16 Miscellaneous memory safety hazards (rv:45.0 / rv:38.7)

Some m

IE11

Bulletins 1-15 of 30		
Date	Bulletin Number	K
3/8/2016	MS16-023	3
2/9/2016	MS16-009	3
1/12/2016	MS16-001	3
12/8/2015	MS15-124	3
11/10/2015	MS15-112	3
10/13/2015	MS15-106	3
9/8/2015	MS15-094	3
8/18/2015	MS15-093	3
8/11/2015	MS15-079	3
7/14/2015	MS15-065	3
6/9/2015	MS15-056	3
5/12/2015	MS15-043	3

Internet Explorer 11			
Windows 7 for 32-bit Systems Service Pack 1	Internet Explorer 11 (3139929)	Remote Code Execution	Critical
Windows 7 for x64-based Systems Service Pack 1	Internet Explorer 11 (3139929)	Remote Code Execution	Critical
Windows 8.1 for 32-bit Systems	Internet Explorer 11 (3139929)	Remote Code Execution	Critical
Windows 8.1 for x64-based Systems	Internet Explorer 11 (3139929)	Remote Code Execution	Critical
Windows Server 2008 R2 for x64-based Systems Service Pack 1	Internet Explorer 11 [1] (3139929)	Remote Code Execution	Moderate
Windows Server 2012 R2	Internet Explorer 11 (3139929)	Remote Code Execution	Moderate
Windows RT 8.1	Internet Explorer 11 [1] [2] (3139929)	Remote Code Execution	Critical
Windows 10 for 32-bit Systems [3] (3140745)	Internet Explorer 11	Remote Code Execution	Critical
Windows 10 for x64-based Systems [3] (3140745)	Internet Explorer 11	Remote Code Execution	Critical
Windows 10 Version 1511 for 32-bit Systems [3] (3140768)	Internet Explorer 11	Remote Code Execution	Critical
Windows 10 Version 1511 for x64-based Systems [3]	Internet Explorer 11	Remote Code Execution	Critical

Some memory corruption bugs

https://www.mdsec.co.uk/2018/02/adobe-flash-exploitation-then-and-now-from-cve-2015-5119-to-cve-2018-4878/

"It is this object that is free'd, however the "*danglingpointer*" variable still references this memory, meaning that we have a **use-after-free** condition"

Programs and Data

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Definition of a “program”:

> “A program is a set of instructions which modifies data”

Definition of a “program”:

“A program is a set of instructions ~~which modifies data”~~

which is controlled by data”

Definition of a “program”:

“A program is a set of instructions ~~which modifies data”~~ which is controlled by data”

Or in other words:

Data is manipulating the instruction flow of a program, not the other way round

Weird machines

Weird machines:

> In computer security, the weird machine is a computational artifact where additional code execution can happen outside the original specification of the program.
>
> It is closely related to the concept of weird instructions, which are the building blocks of an exploit based on crafted input data.

Programming of “Weird Machines”

Gain very detailed understanding of:

- Program logic
- Implementation of “hidden mechanics”
  - Stack, Heap etc.
- Error conditions

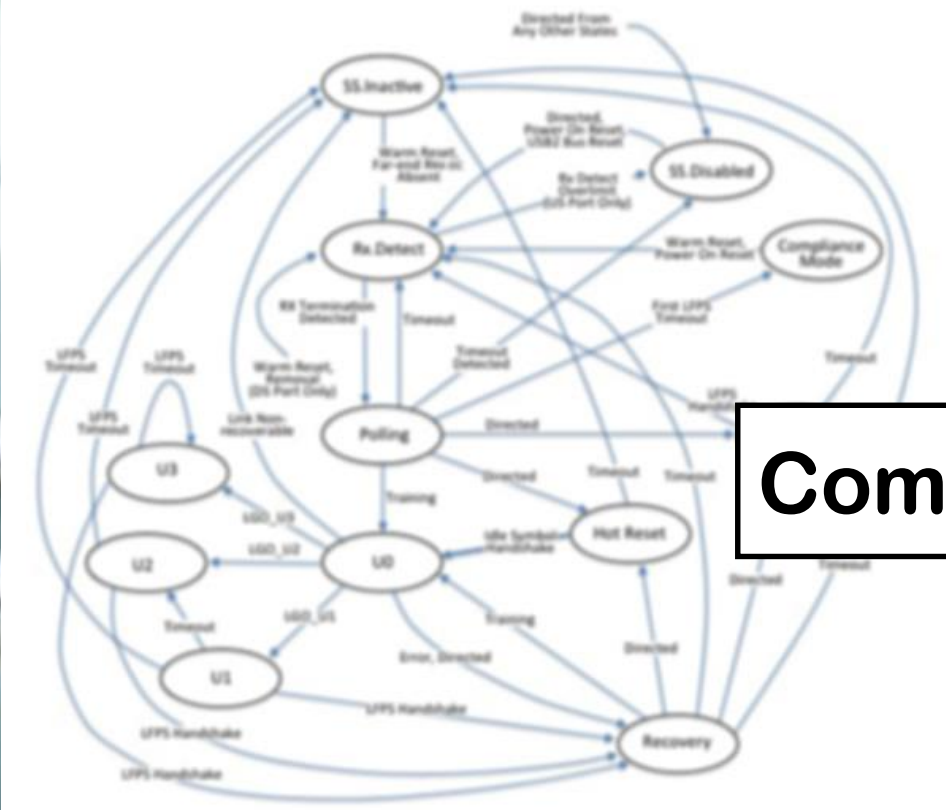
Leads from from:

- “This bugs randomly crashes my program!”

To:

- “This bugs lets me reliably execute arbitrary code”

What are memory corruption bugs doing?

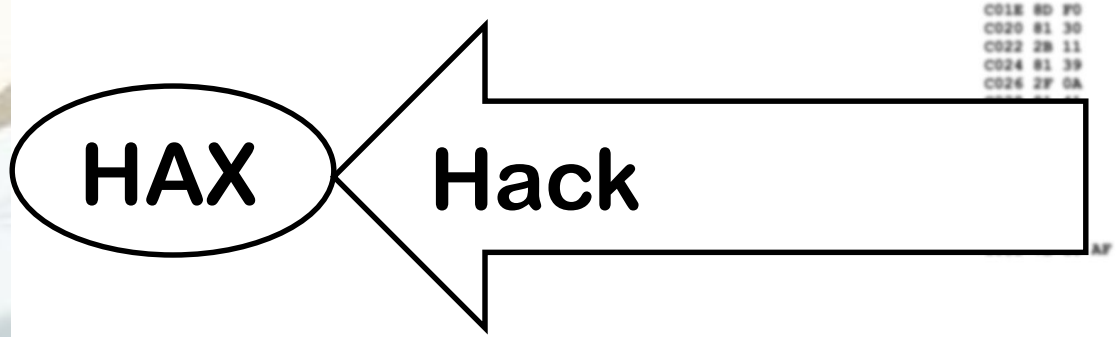


Compile

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                        ****************************************
                        * FUNCTION: INITA - Initialise ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013                    RESETA  EQU     %00010011
0011                    CTLREG  EQU     %00010001

C003 86 13     INITA    LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ****************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0     INHEX    BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F     HEXRTS   AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF  HEXERR   JMP     CTRL      RETURN TO CONTROL LOOP
```

What are memory corruption bugs doing?


```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2

C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                        ****************************************
                        * FUNCTION: INITA - Initialise ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013                    RESETA  EQU     %00010011
0011                    CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ****************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ****************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
                        CMP A   #'A
                        BMI     HEXERR    NOT HEX
                        CMP A   #'F
                        BGT     HEXERR
                        SUB A   #7        FIX A-F
C03A 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
                        RTS
C03F 7E C0 AF   HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP
```



«Weird Machines, Exploitability, Non-Exploitability»

Consequences (IV)

- Exploitation is programming. Automated exploitation is a form of code synthesis.
- Contrary to real-world programs, attacks are successful even if they only work probabilistically.

Recap

Software:

- Important software is written in **C/C++**
- Memory corruption bugs are very, very prevalent

- We are concerned with **memory corruption vulnerabilities**
  - Modify stuff in a program which should not be possible
- A program which misuses a memory corruption vulnerability is called an **exploit**
  - There can be local-, server- and client exploits
- A exploit injects **additional code** into a trusted app and executes it

- For attacker, **data influences execution of code** (weird machines)

Is exploit writing hacking?

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Is exploit writing hacking?

Hack

1. to cut, notch, slice, chop, or sever (something) with or as with heavy, irregular blows (often followed by *up* or *down*):
   to hack meat; to hack down trees.

Is exploit writing hacking?

An aspect of hack value is performing feats for the sake of showing that they can be done, even if others think it is difficult. **Using things in a unique way outside their intended purpose** is often perceived as having hack value.

Examples:

- using a dot matrix impact printer to produce musical notes
- using an optical mouse as barcode reader.
- making soup with your coffee machine

https://en.wikipedia.org/wiki/Hacker_culture

Is exploit writing hacking?

hack: *Computers.*

to modify (a computer program or electronic device) or write (a program) in a skillful or clever way:
Developers have hacked the app.
I hacked my tablet to do some very cool things.

to circumvent security and break into (a network, computer, file, etc.), usually with malicious intent:
Criminals hacked the bank's servers yesterday.
Our team systematically hacks our network to find vulnerabilities.

http://www.dictionary.com/browse/hacking

Is exploit writing hacking?

Hackerethik:

Freier Zugriff auf Computer

Freier Zugriff auf Wissen

Misstrauen gegenüber Autoritäten und Bevorzugung von Dezentralisierung.

Hacker sollten nur nach ihrer Fähigkeit beurteilt werden.

Du kannst Kunst und Schönheit mittels Computer erzeugen

Verbesserung der Welt durch das Verbreiten von Technologien

https://de.wikipedia.org/wiki/Hackerethik

\/\The Conscience of a Hacker/\/

http://phrack.org/issues/7/3.html

Another one got caught today, it's all over the papers. "Teenager Arrested in Computer Crime Scandal", "Hacker Arrested after Bank Tampering"... Damn kids. They're all alike.

I made a discovery today. I found a computer. Wait a second, this is cool. It does what I want it to. If it makes a mistake, it's because I screwed it up. Not because it doesn't like me... And then it happened... a door opened to a world... rushing through the phone line like heroin through an addict's veins, an electronic pulse is sent out, a refuge from the day-to-day incompetencies is sought... a board is found.

This is our world now... the world of the electron and the switch, the beauty of the baud. **We explore... and you call us criminals. We seek after knowledge... and you call us criminals.** We exist without skin color, without nationality, without religious bias... and you call us criminals.

Obligatory history lesson…

Compass Security Schweiz AG
Werkstrasse 20
Postfach 2038
CH-8645 Jona

Tel +41 55 214 41 60
Fax +41 55 214 41 61
team@csnc.ch
www.csnc.ch

Morris worm

- 02.11.1988
- Written by graduate student Robert Morris, MIT
- He wanted to count the number of computers on the internet (~60’000)
- Worm had a bug, re-infected already infected computers, killed the Internet
- Attacked fingerd and sendmail (and some more things)
- One of the first Buffer Overflow (memory corruption) used publicly

Morris worm

fingerd bug in BSD4 on VAX machines:

> The bug exploited to break fingerd involved **overrunning the buffer the daemon used for input.** The standard C library has a few routines that read input without checking for bounds on the buffer involved. In particular, the **gets** call takes input to a buffer without doing any bounds checking; this was the call exploited by the Worm.

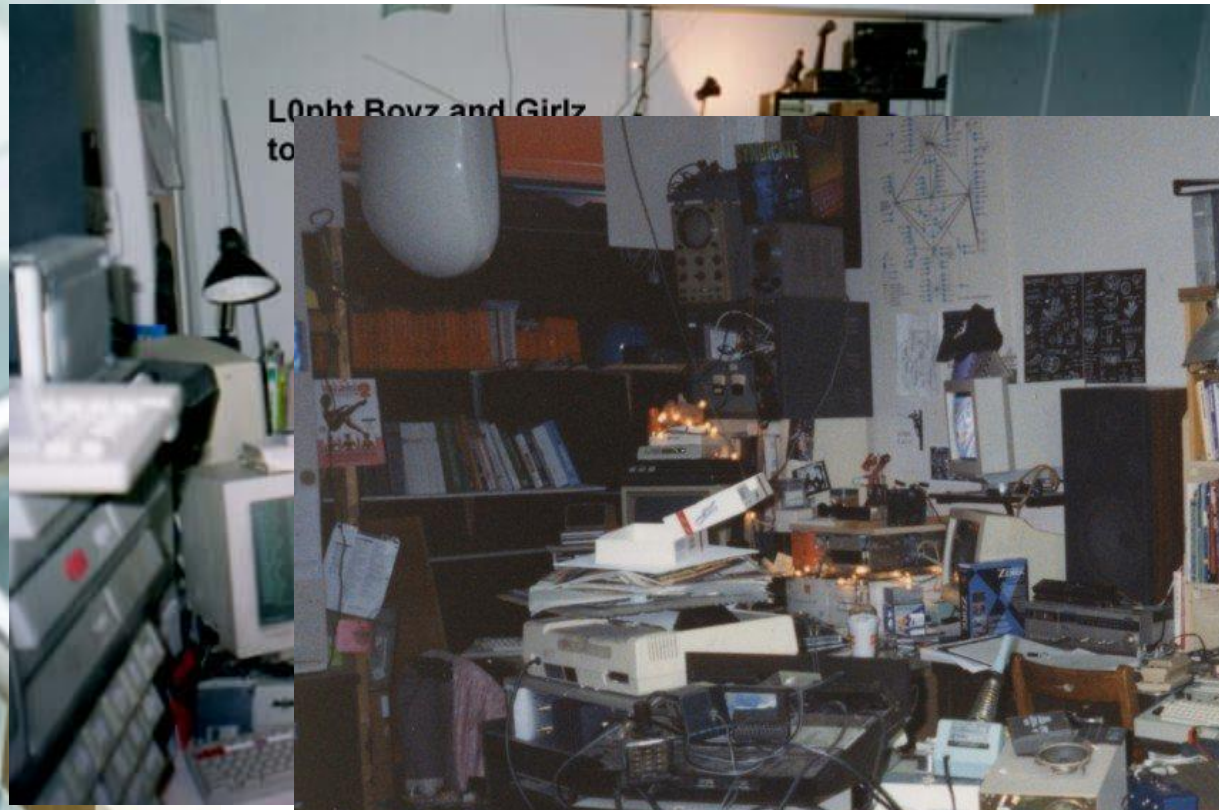
The Internet Worm Program: An Analysis (2004)

- http://spaf.cerias.purdue.edu/tech-reps/823.pdf

Morris worm


Zero Cool crashed 1,507 systems
in one day. Biggest crash in history.

l0pht



L0pht – “We can take down the internet in 30 minutes”

1992-2000

Now:

- Mudge: DARPA, Google.
- Weld Pond: Veracode.
- Kingpin: DEFCON.