Client-side attacks: Attacking the weakest link

Antonio Hüseyin Barresi
Laboratory for Software Technology, ETH Zürich

SIGS Special Event

Zürich, 1.7.2015

100 banks, 30 countries, 2 years

$1 Billion

BUSINESS HACKING

Hackers Steal $1 Billion in Massive, Worldwide Breach

Matt Vella @mattvella | Feb. 15, 2015

A prominent cybersecurity firm says that thieves have infiltrated more than 100 banks in 30 countries over the past two years

Bloomberg/Getty Images

Hackers have stolen as much as $1 billion from banks around the world, according to a prominent cybersecurity firm. In a report scheduled to be delivered Monday, Russian security company Kaspersky Lab claims that a hacking ring has infiltrated more than 100 banks in 30 countries over the past two years.

Kaspersky says digital thieves gained access to banks' computer systems through phishing schemes and other confidence scams. Hackers then lurked in the institutions' systems, taking screen shots or even video of employees at work. Once familiar with the banks' operations, the hackers could steal funds without raising alarms, programming ATMs to dispense money at specific times for instance or transferring funds to fraudulent accounts. First outlined by the

The great bank robbery

KASPERSKY

3

1. Executive Summary

From late 2013 onwards, several banks and financial institutions have been attacked by an unknown group of cybercriminals. In all these attacks, a similar modus operandi was used. According to victims and the law enforcement agencies (LEAs) involved in the investigation, this could result in cumulative losses of up to 1 billion USD. The attacks are still active.

This report provides a technical analysis of these attacks.

The motivation for the attackers, who are making use of techniques commonly seen in Advanced Persistent Threats (APTs), appears to be financial gain as opposed to espionage.

An analysis of the campaign has revealed that the initial infections were achieved using spear phishing emails that appeared to be legitimate banking communications, with Microsoft Word 97 – 2003 (.doc) and Control Panel Applet (.CPL) files attached. We believe that the attackers also redirected to exploit kits website traffic that related to financial activity.

The email attachments exploit vulnerabilities in Microsoft Office 2003, 2007 and 2010 (CVE-2012-0158 and CVE-2013-3906) and Microsoft Word (CVE-2014-1761). Once the vulnerability is successfully exploited, the shellcode decrypts and executes the backdoor known as **Carbanak**.

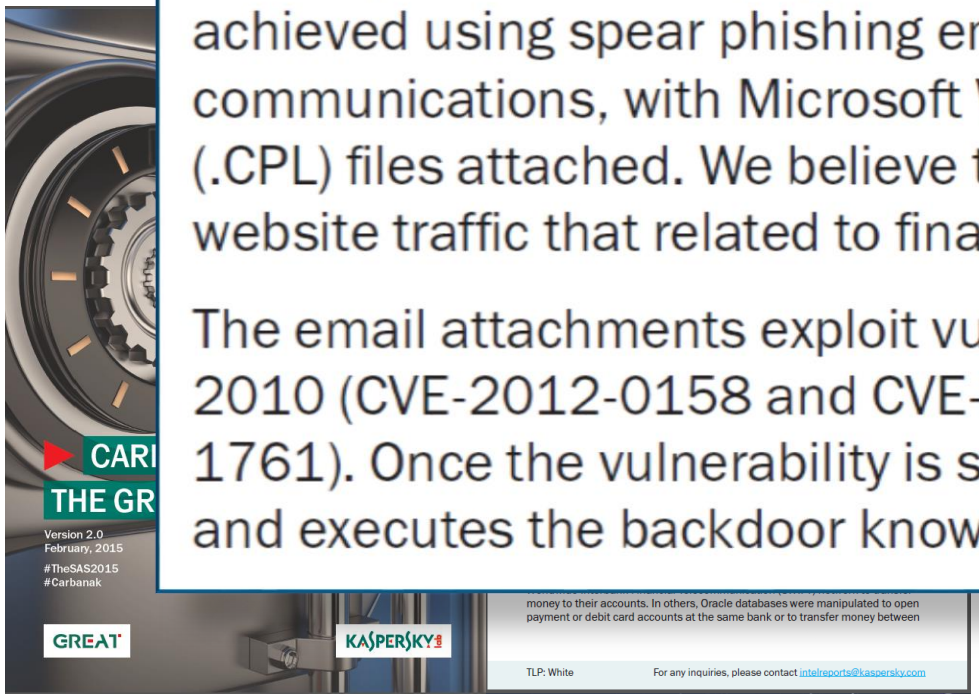
Carbanak is a remote backdoor (initially based on Carberp), designed for espionage, data exfiltration and to provide remote access to infected machines. Once access is achieved, attackers perform a manual reconnaissance of the victim's networks. Based on the results of this operation, the attackers use different lateral movement tools in order to get access to the critical systems in the victim´s infrastructure. They then install additional software such as the Ammyy Remote Administration Tool, or even compromise SSH servers. Notably, some of the latest versions of the analyzed Carbanak malware appear not to use any Carberp source code.

Once the attackers successfully compromise the victim´s network, the primary internal destinations are money processing services, Automated Teller Machines (ATM) and financial accounts. In some cases, the attackers used the Society for Worldwide Interbank Financial Telecommunication (SWIFT) network to transfer money to their accounts. In others, Oracle databases were manipulated to open payment or debit card accounts at the same bank or to transfer money between

TLP: White For any inquiries, please contact intelreports@kaspersky.com

https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/

The great bank robbery

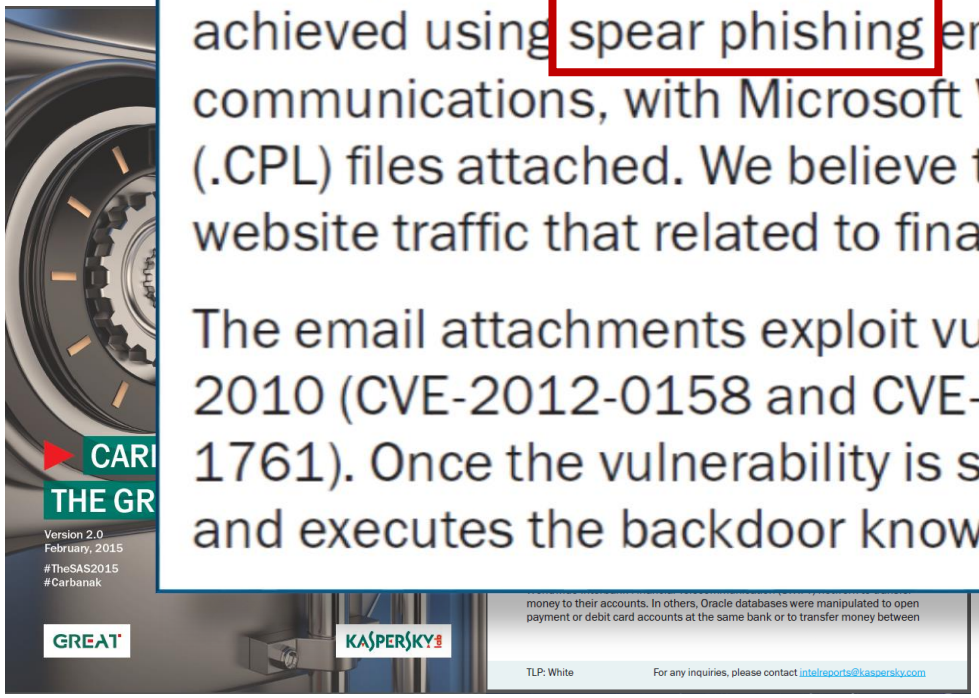


An analysis of the campaign has revealed that the initial infections were achieved using spear phishing emails that appeared to be legitimate banking communications, with Microsoft Word 97 – 2003 (.doc) and Control Panel Applet (.CPL) files attached. We believe that the attackers also redirected to exploit kits website traffic that related to financial activity.

The email attachments exploit vulnerabilities in Microsoft Office 2003, 2007 and 2010 (CVE-2012-0158 and CVE-2013-3906) and Microsoft Word (CVE-2014-1761). Once the vulnerability is successfully exploited, the shellcode decrypts and executes the backdoor known as **Carbanak**.

https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/

The great bank robbery

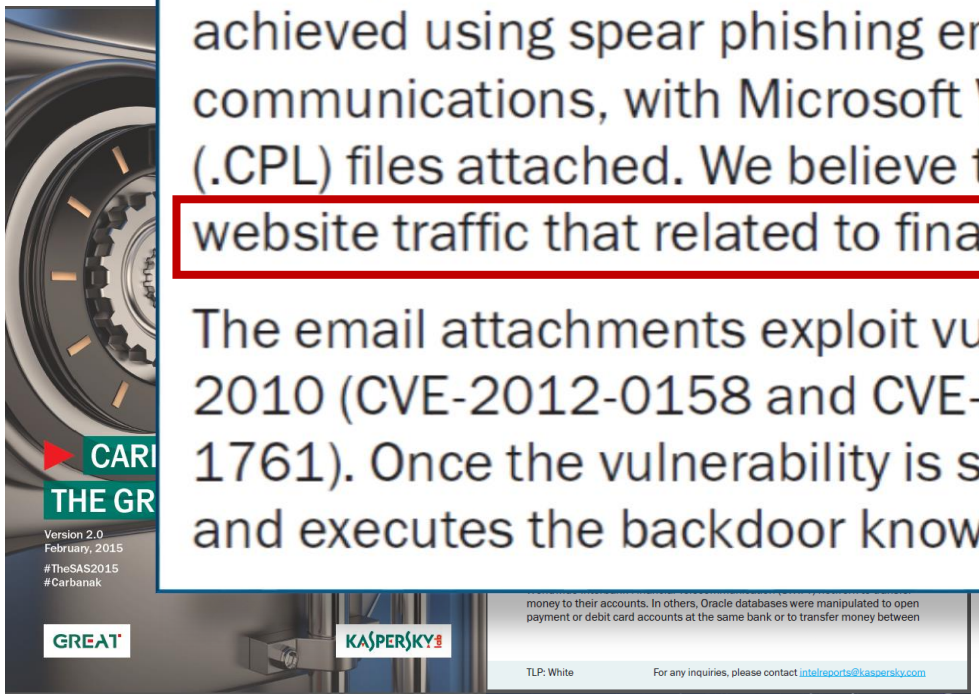


An analysis of the campaign has revealed that the initial infections were achieved using spear phishing emails that appeared to be legitimate banking communications, with Microsoft Word 97 – 2003 (.doc) and Control Panel Applet (.CPL) files attached. We believe that the attackers also redirected to exploit kits website traffic that related to financial activity.

The email attachments exploit vulnerabilities in Microsoft Office 2003, 2007 and 2010 (CVE-2012-0158 and CVE-2013-3906) and Microsoft Word (CVE-2014-1761). Once the vulnerability is successfully exploited, the shellcode decrypts and executes the backdoor known as **Carbanak**.

https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/

The great bank robbery

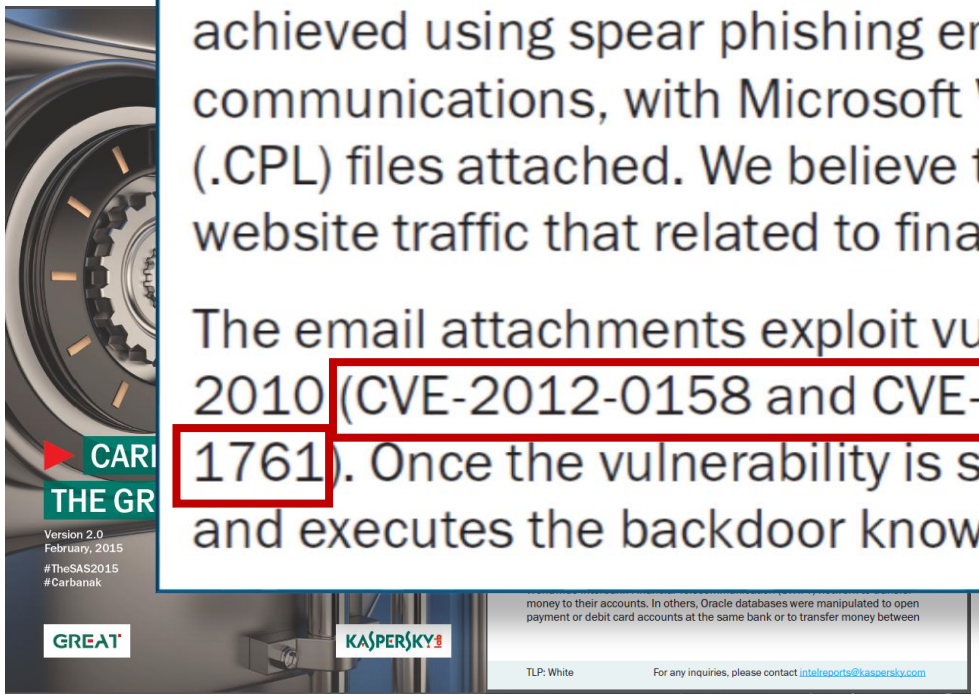


An analysis of the campaign has revealed that the initial infections were achieved using spear phishing emails that appeared to be legitimate banking communications, with Microsoft Word 97 – 2003 (.doc) and Control Panel Applet (.CPL) files attached. We believe that the attackers also redirected to exploit kits website traffic that related to financial activity.

The email attachments exploit vulnerabilities in Microsoft Office 2003, 2007 and 2010 (CVE-2012-0158 and CVE-2013-3906) and Microsoft Word (CVE-2014-1761). Once the vulnerability is successfully exploited, the shellcode decrypts and executes the backdoor known as **Carbanak**.

https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/

The great bank robbery



An analysis of the campaign has revealed that the initial infections were achieved using spear phishing emails that appeared to be legitimate banking communications, with Microsoft Word 97 – 2003 (.doc) and Control Panel Applet (.CPL) files attached. We believe that the attackers also redirected to exploit kits website traffic that related to financial activity.

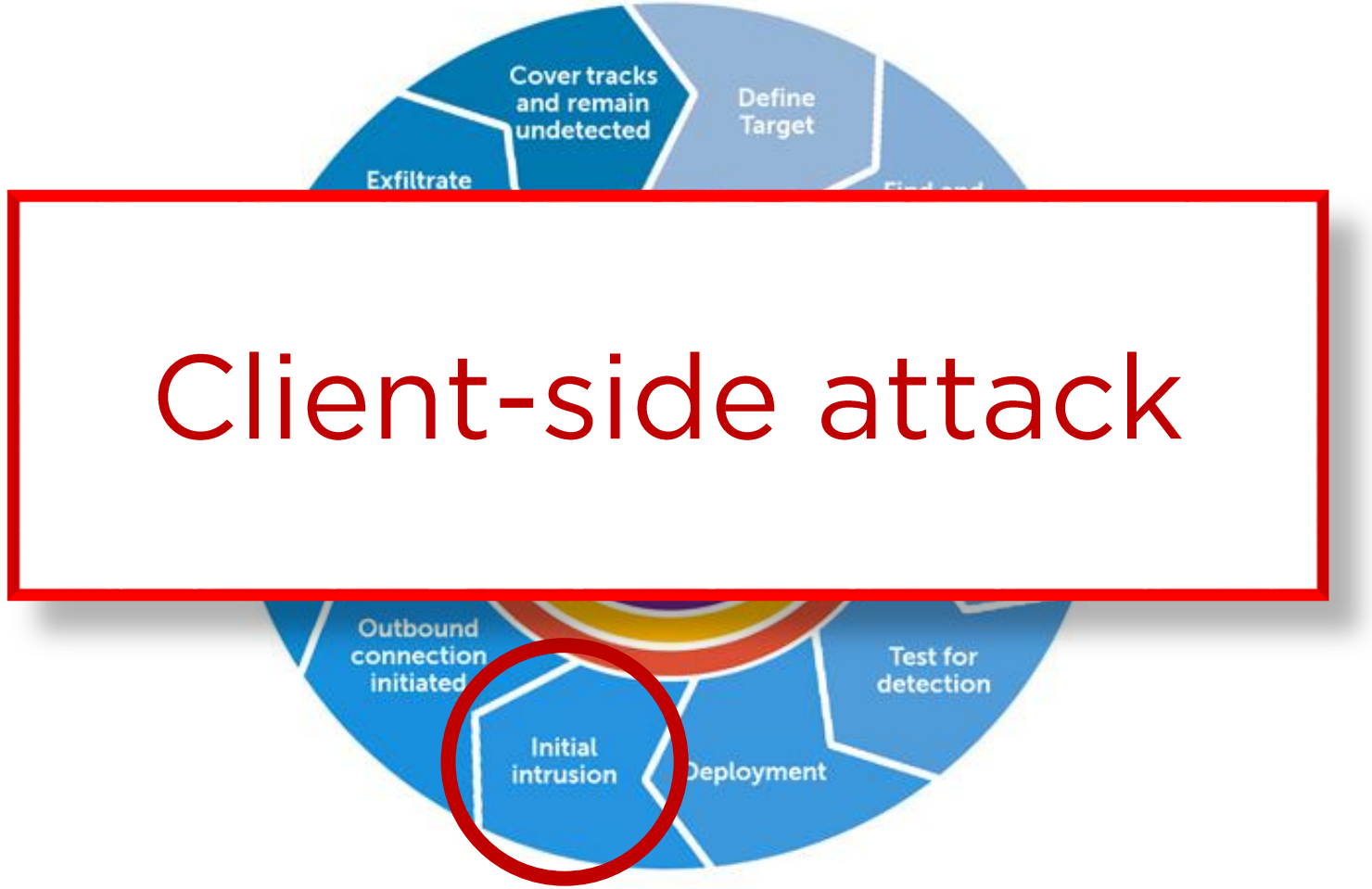
The email attachments exploit vulnerabilities in Microsoft Office 2003, 2007 and 2010 (CVE-2012-0158 and CVE-2013-3906) and Microsoft Word (CVE-2014-1761). Once the vulnerability is successfully exploited, the shellcode decrypts and executes the backdoor known as **Carbanak**.

https://securelist.com/blog/research/68732/the-great-bank-robbery-the-carbanak-apt/

APT lifecycle & client-side attacks

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

APT lifecycle & client-side attacks



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

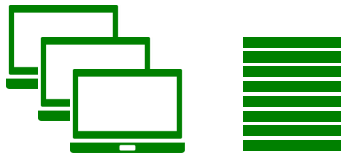
APTs & client-side attacks

APT	Client-side attack used	Type
Carbanak/Anunak	Yes	Multiple, file based exploits and web exploit kits
Equation	Yes	Multiple client-side exploits
Darkhotel	Yes	Multiple client-side exploits
Mandiant APT1	Yes	Executables (zipped)
Uroburos	Unknown	Unknown
Cozyduke	Yes	Executables (zipped)
Inception Framework	Yes	File based exploits
FIN4	Not clear	Not clear

Client-side attack

Internet

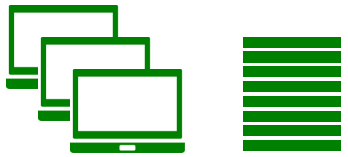
Company
network



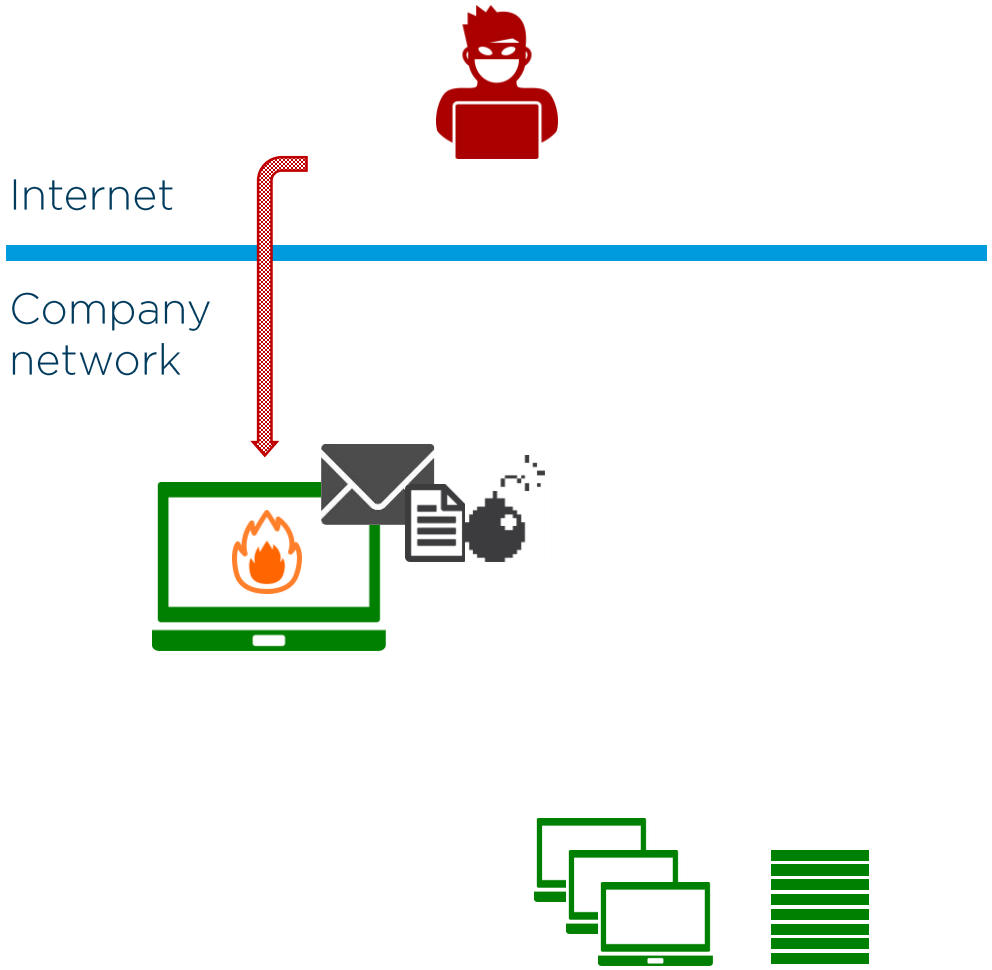
Client-side attack

Internet

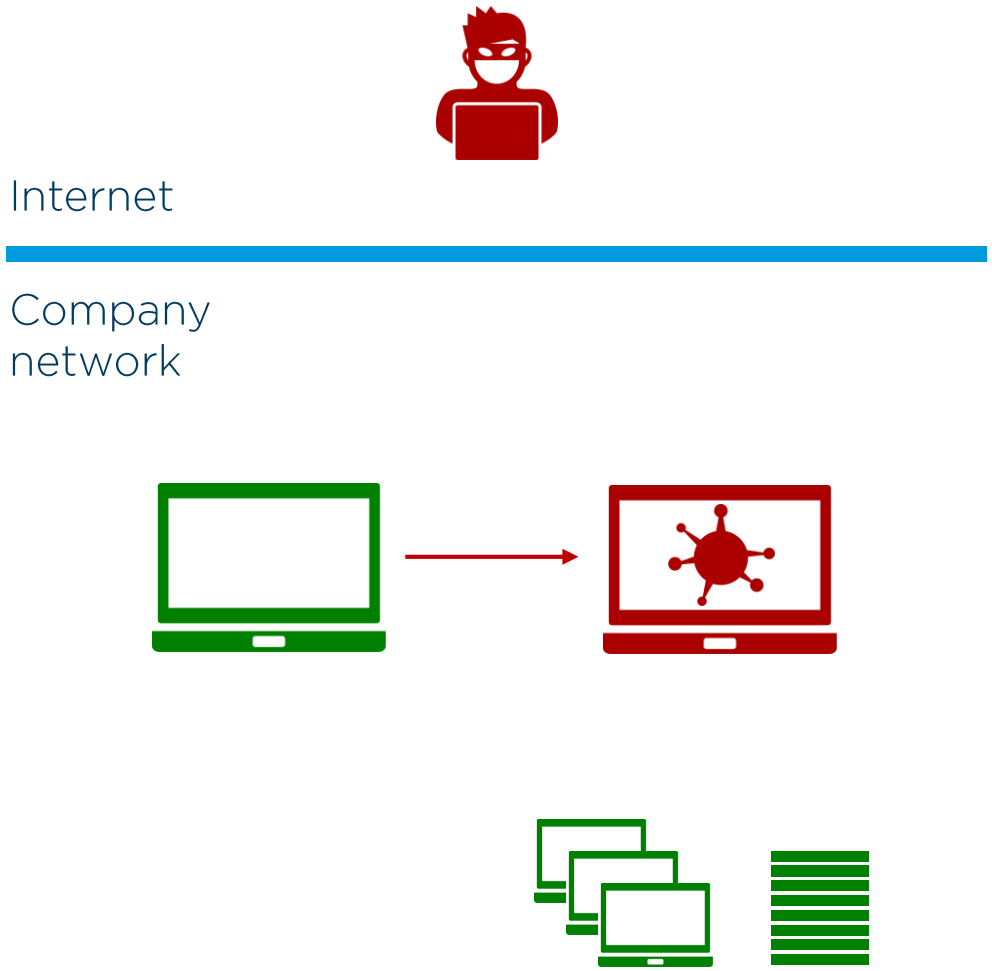
Company
network



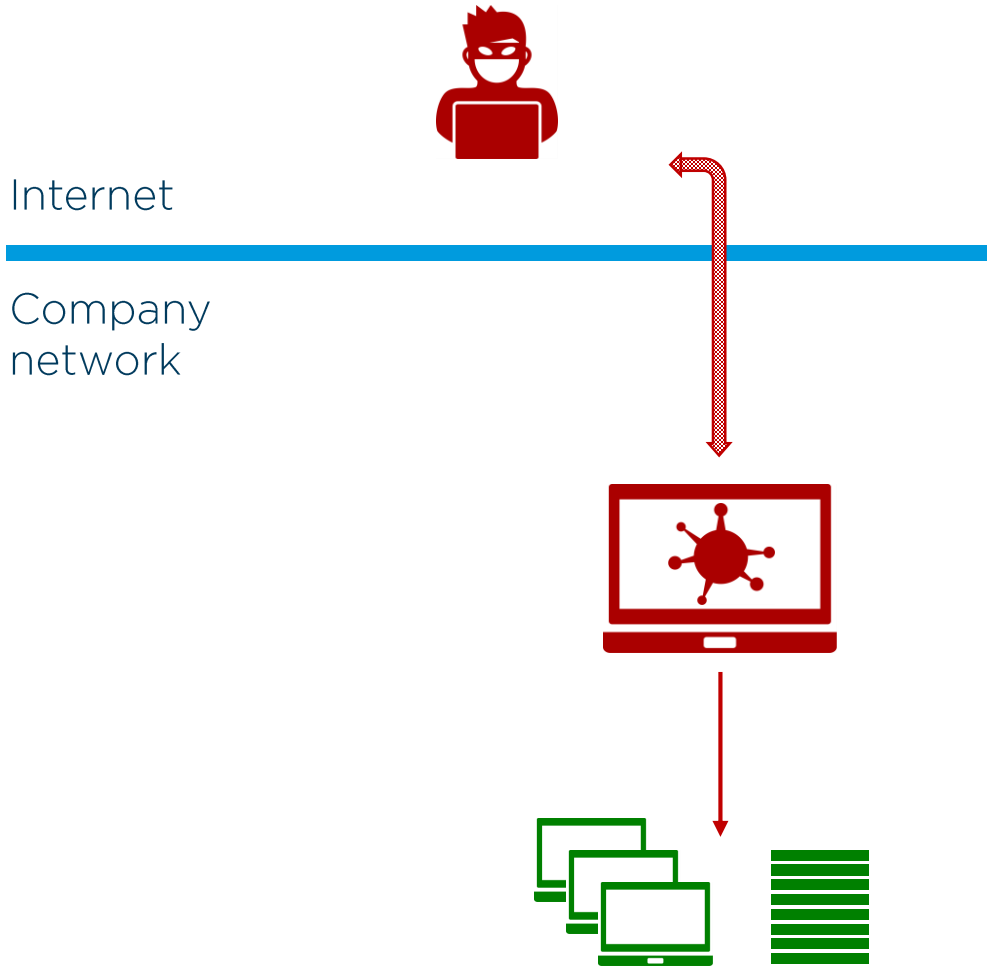
Client-side attack



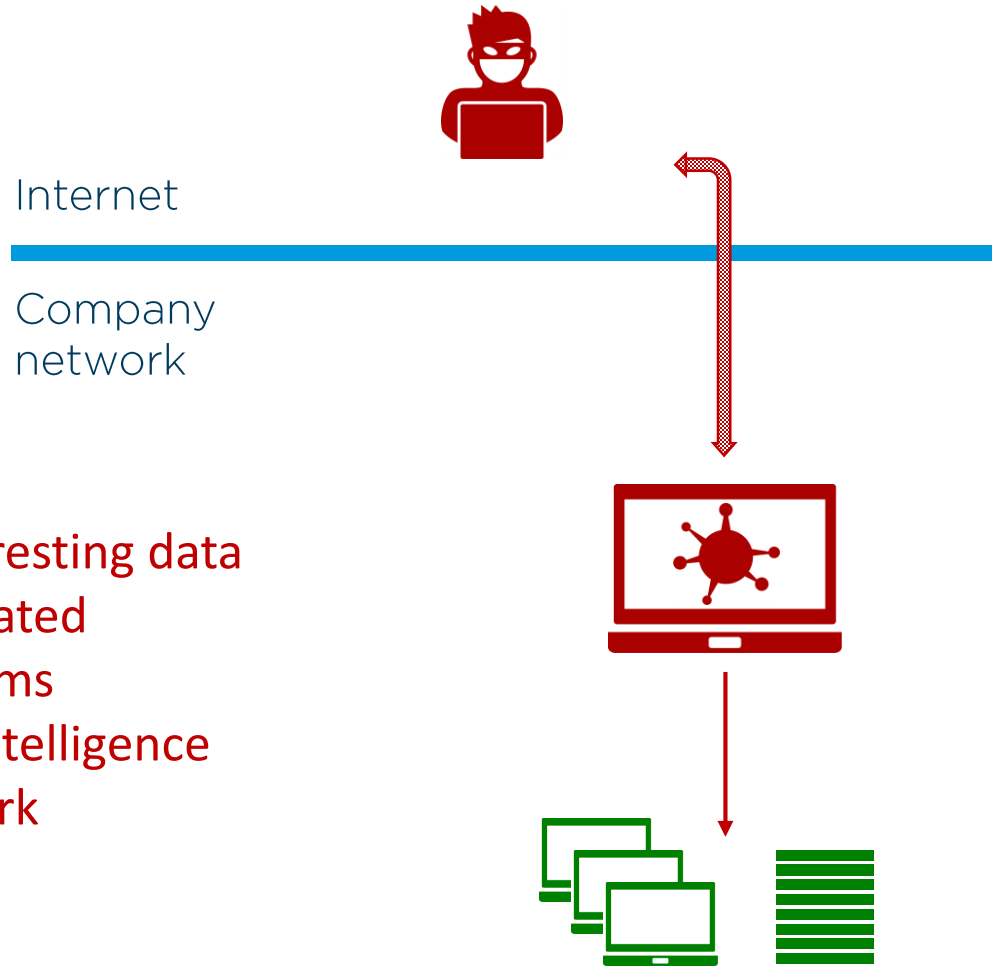
Client-side attack



Client-side attack



Client-side attack



- Might already have interesting data
- Is trusted and authenticated
- Has access to core systems
- Can be used to gather intelligence
- Gateway into the network

Arbitrary code-execution

Types of client-side attacks

Type	Executable	Embedded script	Code-execution exploit
Description	Malicious payload is an executable or disguised as a benign program.	Malicious payload is a script embedded in documents.	Malicious payload is data that triggers code-execution when processed.
Example	Send an .exe to a victim.	Send a .doc with embedded VBScript to a victim.	Send a .pdf to a victim or make the victim visit a website.
Sophistication	Low	Medium	High
User interaction	High	Medium	Low
Noise	High	Medium	Low

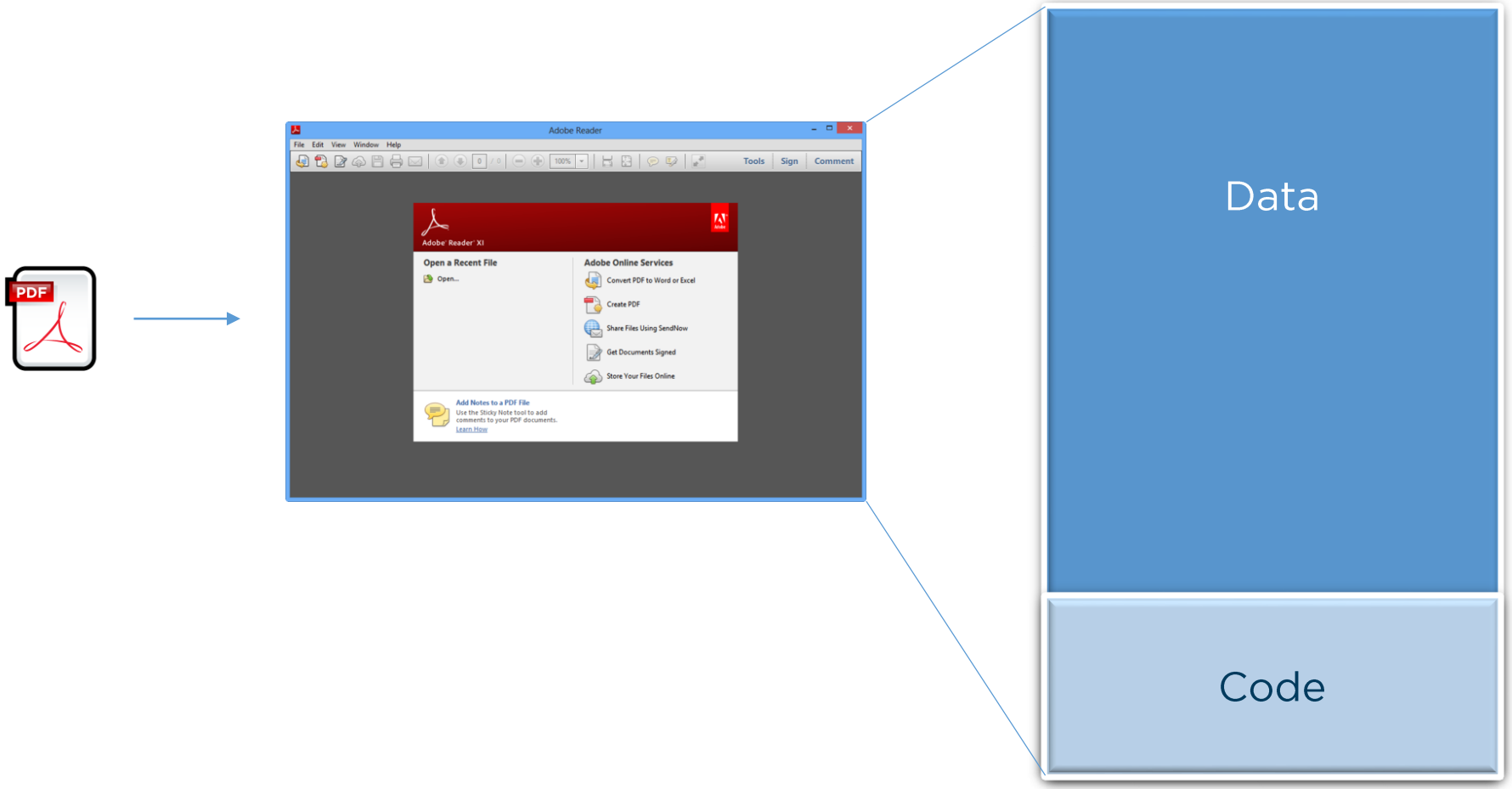
Types of client-side attacks

Type	Executable	Embedded script	Code-execution exploit
Description	Malicious payload is an executable or disguised as a benign program.	Malicious payload is a script embedded in documents.	Malicious payload is data that triggers code-execution when processed.
Example	Send an .exe to a victim.	Send a .doc with embedded VBScript to a victim.	Send a .pdf to a victim or make the victim visit a website.
Sophistication	Low	Medium	High
User interaction	High	Medium	Low
Noise	High	Medium	Low

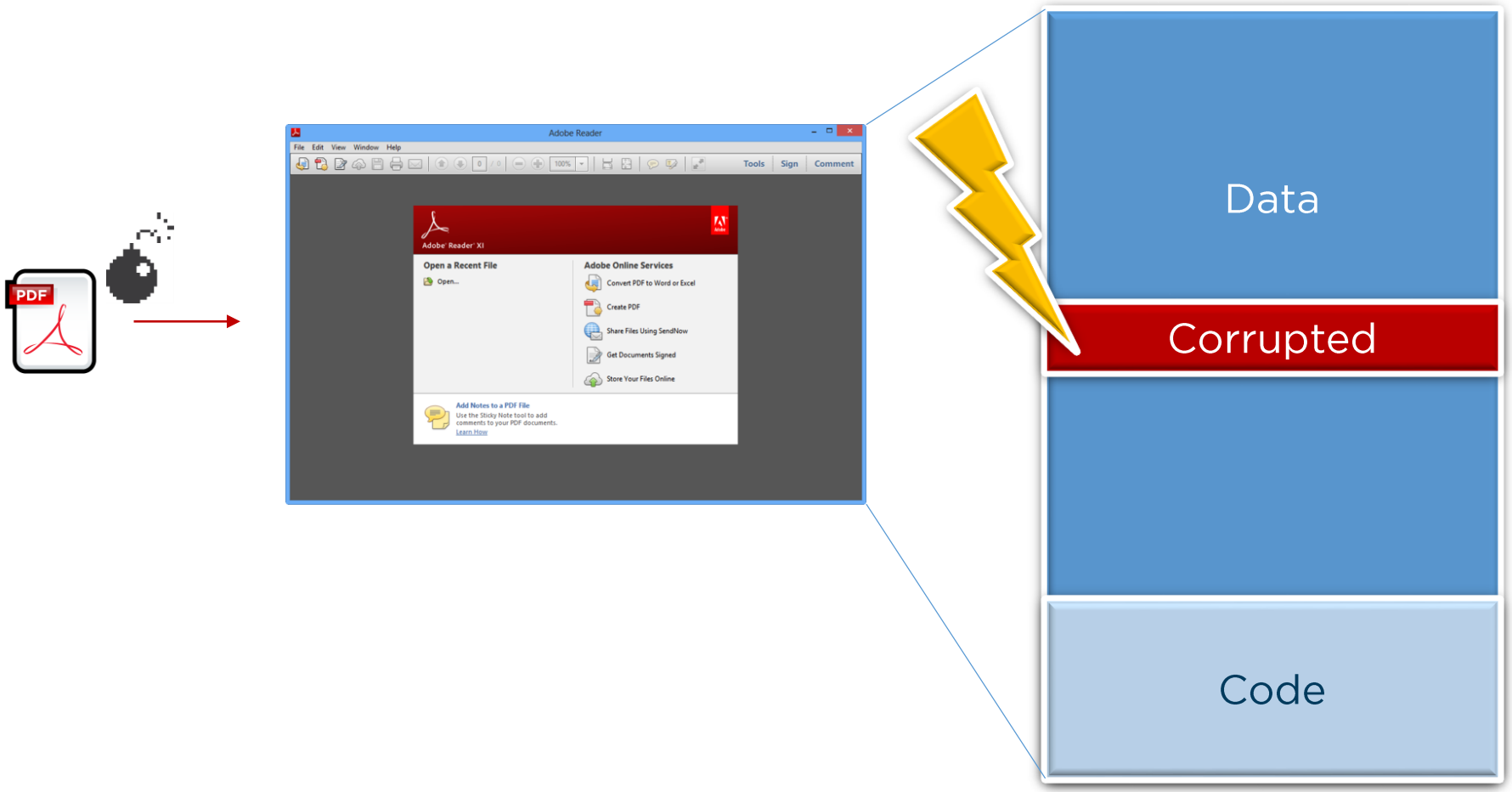
Memory corruption

- Many client-side applications are potentially prone to memory corruption vulnerabilities
- Exist in programs written in “unsafe“ languages
  - C/C++
- Two basic types exist
  - Spatial and temporal memory errors

Memory corruption



Memory corruption



Attack types

- Code corruption attack
- **Control-flow hijack attack**
- Data-only attack
- Information leak

Attack model according to: “SoK: Eternal War in Memory“ Laszlo Szekeres, Mathias Payer, Tao Wei, Dawn Song
http://www.cs.berkeley.edu/~dawnsong/papers/Oakland13-SoK-CR.pdf

Modern exploitation

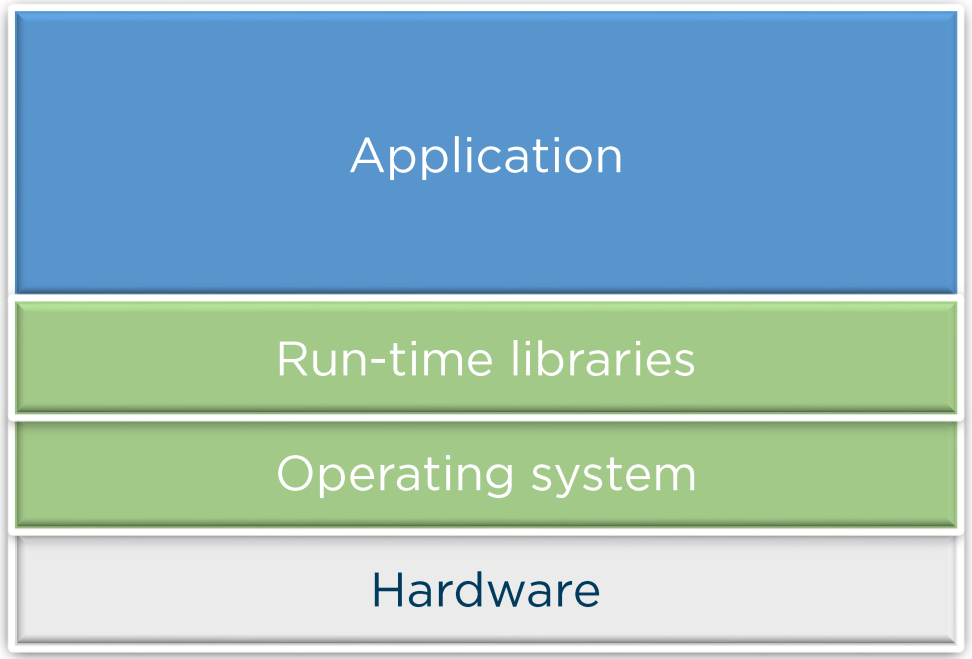
- Very application specific
- First step:
  construct reliable read and write primitives
- Second step:
  hijack control-flow and perform code-reuse attack
- Many exploit mitigation measures have to be bypassed

Attacked client-side software stack

Written in memory unsafe language (C/C++)

Supports client-side scripting or has VM functionality

Large code-base

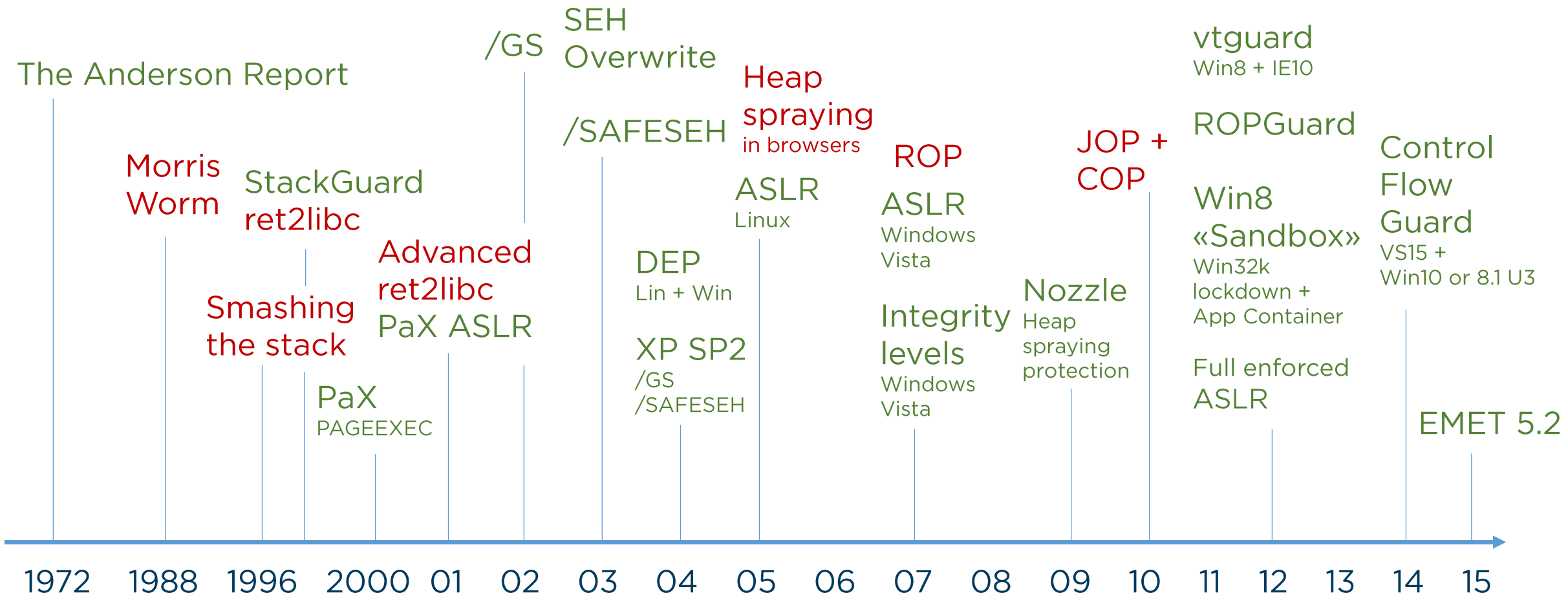


Web browser, document processor, viewer

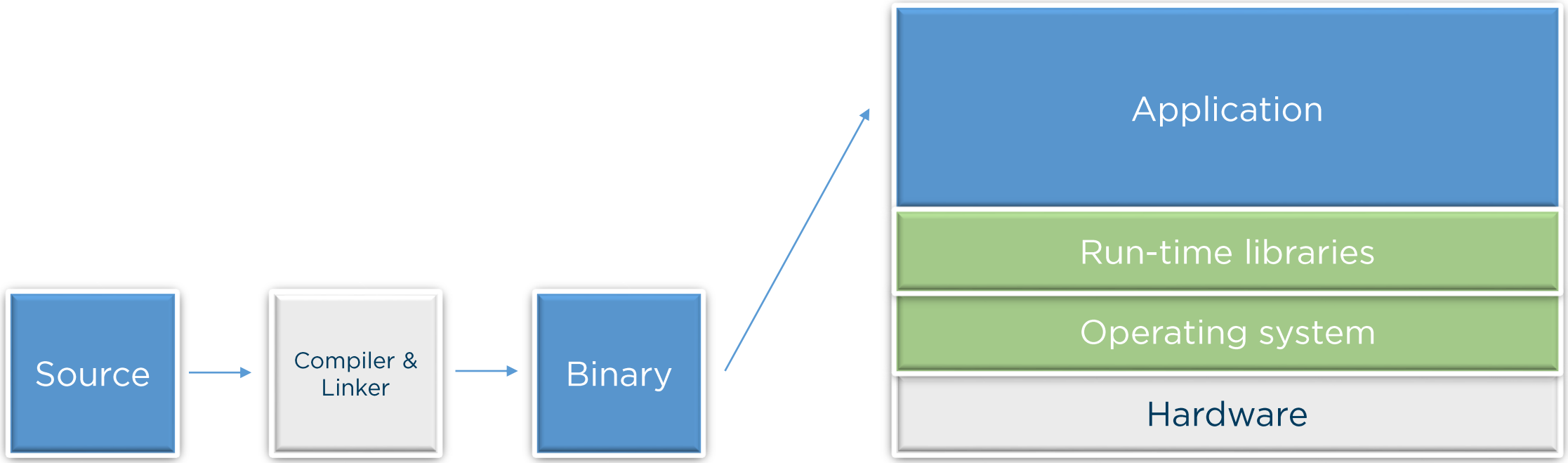
Typical client OS (Windows)

Commodity hardware

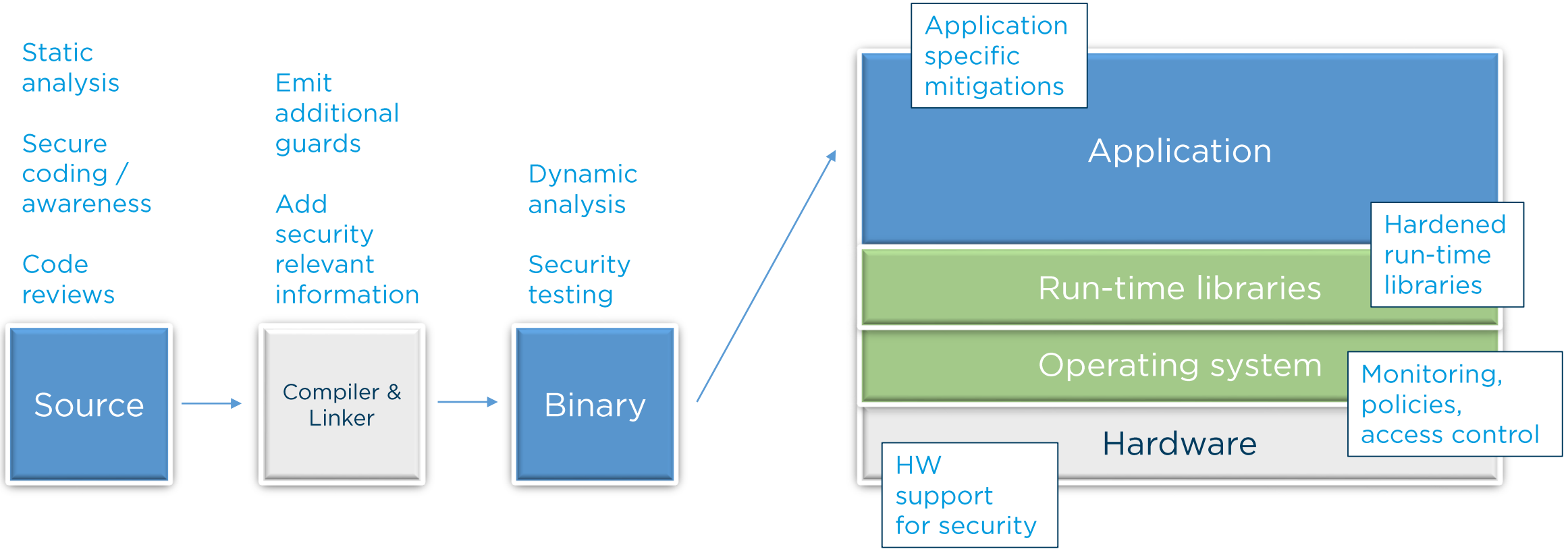
Client-side mitigations



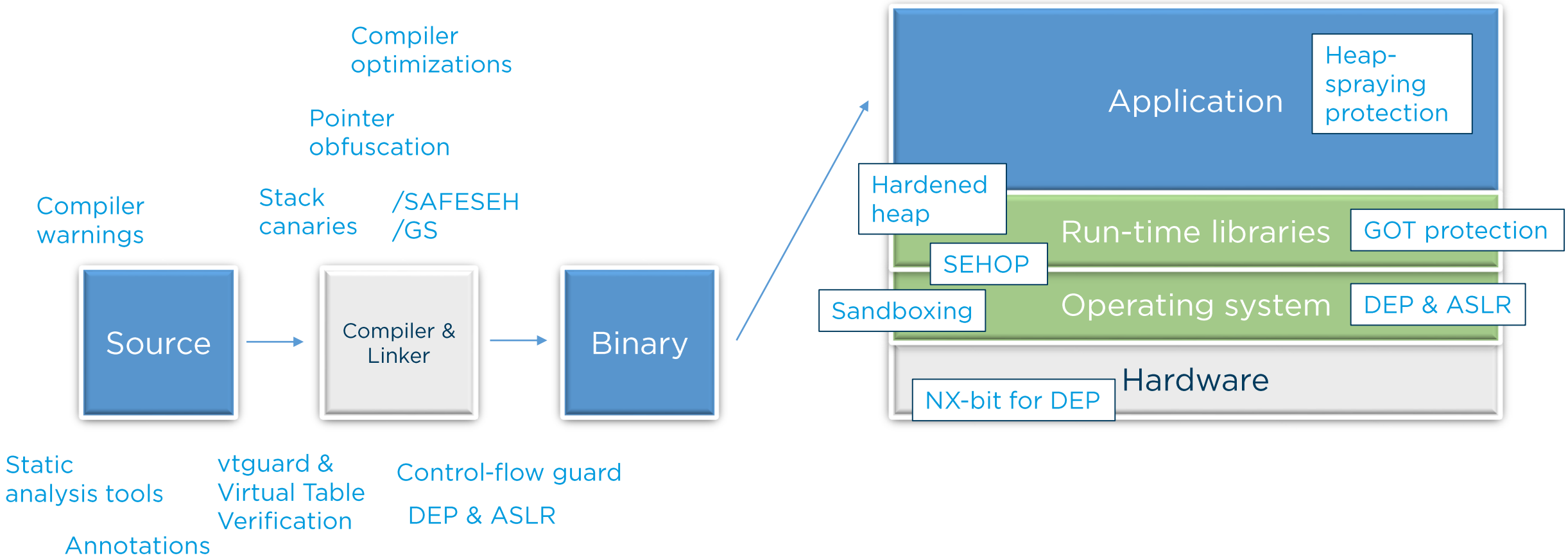
Hardening value chain



Hardening value chain



Hardening value chain



Mitigations deployment delay

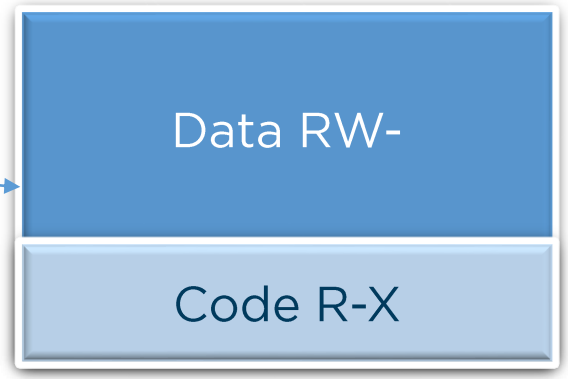
- Deployment of exploit mitigations requires time
- Source or binary compatibility risk
- Support from multiple sides required

Attackers have time to adapt!

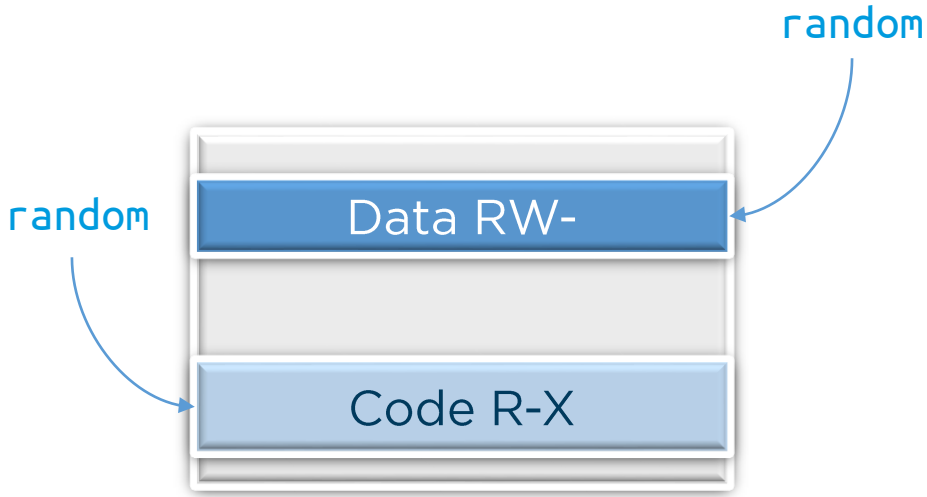
DEP & ASLR

Data Execution Prevetion

Data is not executable



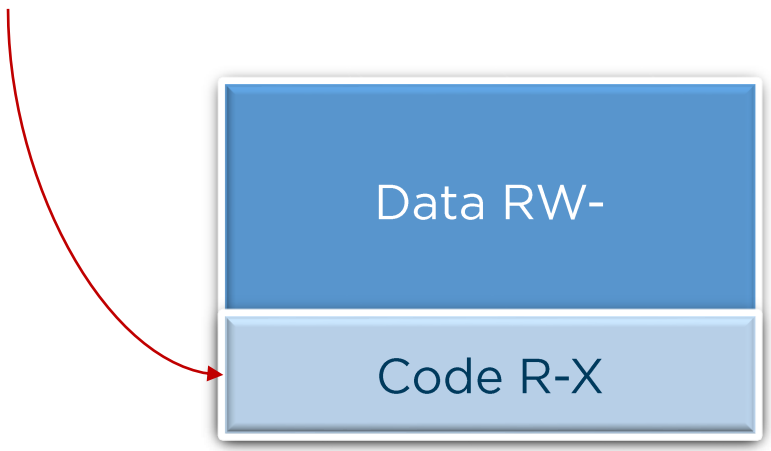
Address Space Layout Randomization



Bypassing DEP & ASLR

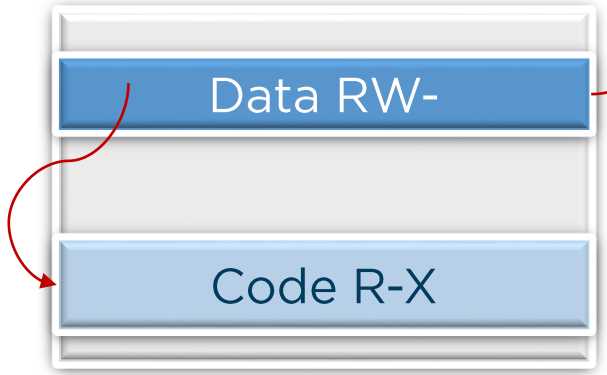
Data Execution Prevetion

Reuse existing code



Address Space Layout Randomization

Leak memory containing randomized addresses



Control-flow guard (CFG)

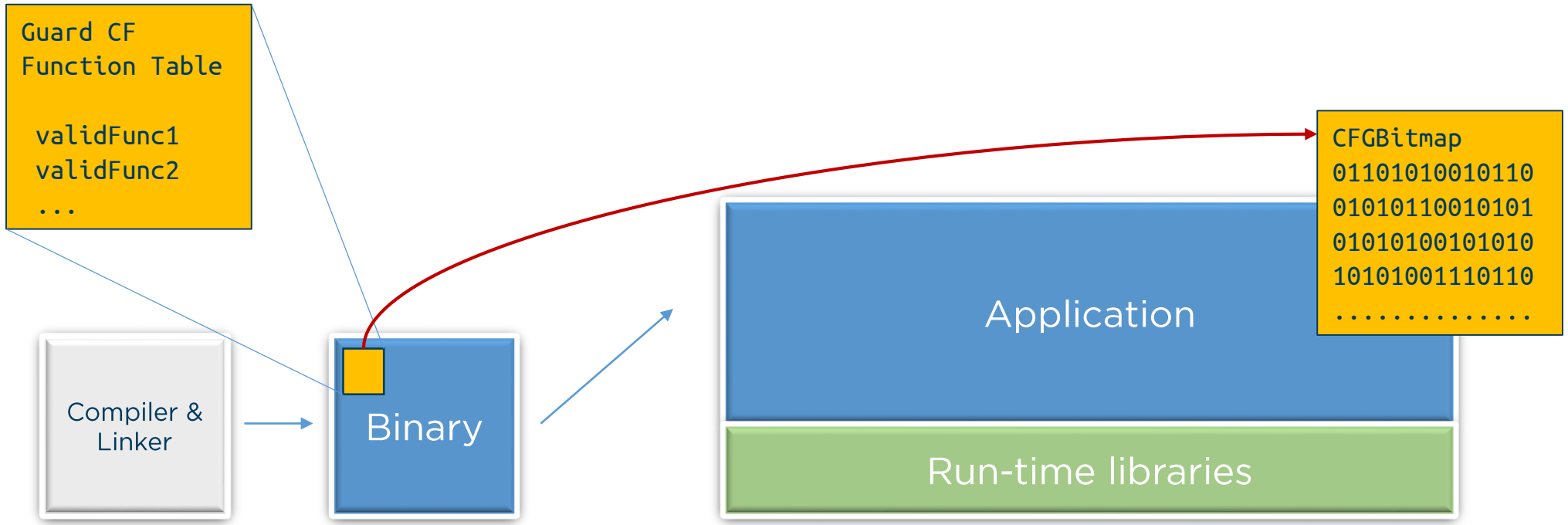
- First adoption of a practical Control-Flow Integrity (CFI) implementation
- Requires recompilation and OS support
  - VS15 + Windows 10 or 8.1 U3
- Restricts indirect call targets to a static global set of locations

  Prevent indirect call hijacking (e.g., Use-after-free) and code-reuse attacks

Control-flow guard (CFG)

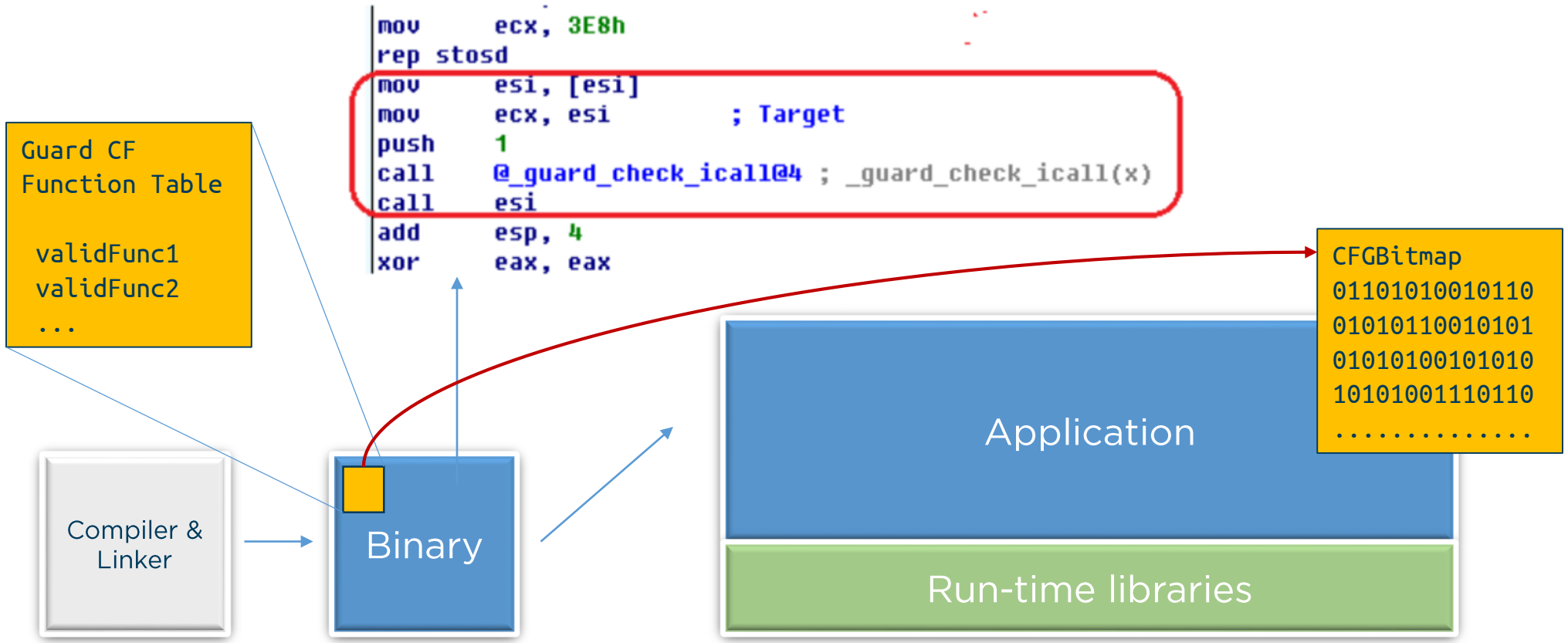
http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf

Control-flow guard (CFG)



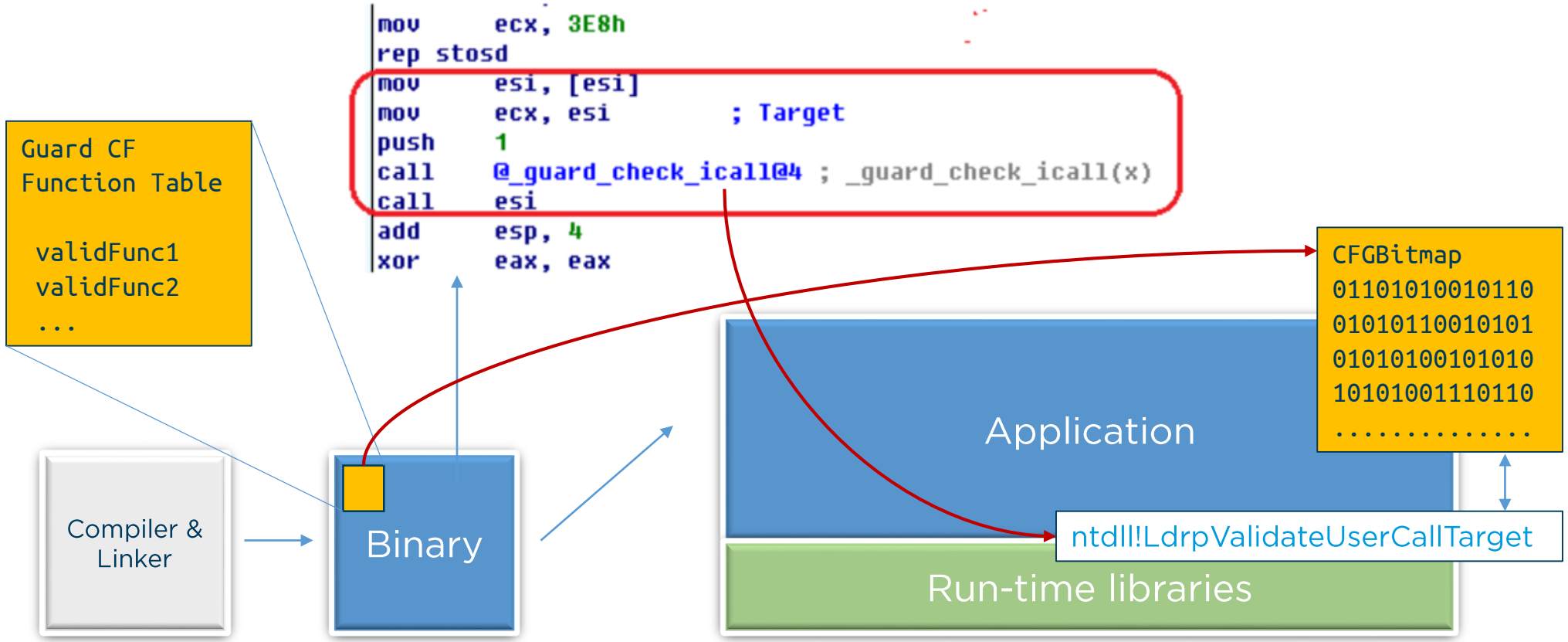
http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf

Control-flow guard (CFG)



http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf

Control-flow guard (CFG)



http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf

CFG weaknesses

- CFGBitmap is at a fixed address
- Main executable has to support CFG
  (no modular support)
- Still too permissive (coarse-grained CFI)
  - Every bit represents 8 bytes in memory
- Does not cover JIT code
  - NTAllocVirtualMemory will set CFGBitmapt to ‘1’s

http://sjc1-te-ftp.trendmicro.com/assets/wp/exploring-control-flow-guard-in-windows10.pdf

Bypassing CFG

Exploiting CVE-2015-0311, Part II: Bypassing Control Flow Guard on Windows 8.1 Update 3

March 25, 2015 by Francisco Falcón

- CVE-2015-0311 Use-After-Free in Adobe Flash Player
- IE 11 Flash in Windows 8.1 U3 has CFG enabled
- Hijack indirect call generated by the Flash JIT

```
0864D88C   8B01          MOV EAX,DWORD PTR DS:[ECX]
0864D88E   8B50 08       MOV EDX,DWORD PTR DS:[EAX+8]
0864D891   8B8A D4000000 MOV ECX,DWORD PTR DS:[EDX+D4]
0864D897   8D55 FC       LEA EDX,DWORD PTR SS:[EBP-4]
0864D89A   8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
0864D89D   8B41 04       MOV EAX,DWORD PTR DS:[ECX+4]
0864D8A0   83EC 04       SUB ESP,4
0864D8A3   52            PUSH EDX
0864D8A4   6A 00         PUSH 0
0864D8A6   51            PUSH ECX
0864D8A7   FFD0          CALL EAX
0864D8A9   83C4 10       ADD ESP,10
0864D8AC   8B4D F0       MOV ECX,DWORD PTR SS:[EBP-10]
0864D8AF   890D 50406908 MOV DWORD PTR DS:[8694050],ECX
0864D8B5   8BE5          MOV ESP,EBP
0864D8B7   5D            POP EBP
0864D8B8   C3            RETN
```

https://blog.coresecurity.com/2015/03/25/exploiting-cve-2015-0311-part-ii-bypassing-control-flow-guard-on-windows-8-1-update-3/

Control-flow integrity (CFI)

- Original publication in 2005
  «Control-Flow Integrity – Principles, Implementations, and Applications» Abadi, Budiu, Erlingsson, Ligatti, CCS'05

- Many CFI implementations proposed since then
  - Compiler-based, binary-only (static rewriting)

- Focus on CFI implementation rather than CFI security

Control-flow integrity (CFI)

- CFI approaches
  XFI (2006), HyperSafe (2010), CFIMon (2012), MoCFI (2012), BinCFI (2013), KCoFI (2014), MCFI (2014)

- CFI evasion

  «Out Of Control: Overcoming Control-Flow Integrity»
  Davi et al.

  «Stitching the Gadgets: On the Ineffectiveness of Coarse-Grained Control-Flow Integrity Protection»
  Göktas et al.

Future?

- CFG like many other exploit mitigations make exploit development harder
- But there is no magic bullet
- Attacker's still leverage memory leaks and code-reuse techniques to bypass all mitigations (ret2libc is 1997)

Zero-day exploits

- Zero-day vulnerabilities in 2014 according to Symantec
- 23 out of 24 potentially used in client-side attacks
- Vulnerability trends in the “Internet Security Threat Report 2015”

CVE Identifier	Description
CVE-2014-0493	Adobe Acrobat And Reader CVE-2014-0493 Remote Code Execution Vulnerability
CVE-2014-0495	Adobe Acrobat and Reader CVE-2014-0495 Remote Code Execution Vulnerability
CVE-2014-0496	Adobe Acrobat And Reader CVE-2014-0496 Remote Code Execution Vulnerability
CVE-2014-0491	Adobe Flash Player And AIR CVE-2014-0491 Remote Security Bypass Vulnerability
CVE-2014-0492	Adobe Flash Player and AIR CVE-2014-0492 Information Disclosure Vulnerability
CVE-2014-0497	Adobe Flash Player CVE-2014-0497 Remote Code Execution Vulnerability
CVE-2014-0322	Microsoft Internet Explorer CVE-2014-0322 Use-After-Free Remote Code Execution Vulnerability
CVE-2013-7331	Microsoft XMLDOM ActiveX Control Multiple Information Disclosure Vulnerabilities
CVE-2014-0502	Adobe Flash Player and AIR CVE-2014-0502 Remote Code Execution Vulnerability
CVE-2014-0502	Adobe Flash Player and AIR CVE-2014-0502 Remote Code Execution Vulnerability
CVE-2014-0498	Adobe Flash Player and AIR CVE-2014-0498 Remote Stack Overflow Vulnerability
CVE-2014-0324	Microsoft Internet Explorer CVE-2014-0324 Memory Corruption Vulnerability
CVE-2014-1761	Microsoft Word CVE-2014-1761 Remote Memory Corruption Vulnerability
CVE-2014-1776	Microsoft Internet Explorer CVE-2014-1776 Remote Code Execution Vulnerability
CVE-2014-0515	Adobe Flash Player CVE-2014-0515 Buffer Overflow Vulnerability
CVE-2014-0517	Adobe Flash Player and AIR CVE-2014-0517 Unspecified Remote Security Bypass Vulnerability
CVE-2014-0518	Adobe Flash Player and AIR CVE-2014-0518 Unspecified Remote Security Bypass Vulnerability
CVE-2014-0520	Adobe Flash Player and AIR CVE-2014-0520 Unspecified Remote Security Bypass Vulnerability
CVE-2014-0519	Adobe Flash Player and AIR CVE-2014-0519 Unspecified Remote Security Bypass Vulnerability
CVE-MAP-NOMATCH	Linux Kernel 'ptrace' Function Call Local Privilege Escalation Vulnerability
CVE-2014-0546	Adobe Acrobat and Reader CVE-2014-0546 Unspecified Security Bypass Vulnerability
CVE-2014-4114	Microsoft Windows CVE-2014-4114 OLE Package Manager Remote Code Execution Vulnerability
CVE-2014-6352	Microsoft Windows CVE-2014-6352 OLE Remote Code Execution Vulnerability
CVE-2014-9163	Adobe Flash Player CVE-2014-9163 Unspecified Stack Based Buffer Overflow Vulnerability

Zero-Day Vulnerabilities Identified in 2014

Source: Symantec

http://www.symantec.com/security_response/publications/threatreport.jsp

Pwn2own 2015

Windows-based targets:

- Google Chrome (64-bit): $75,000
- Microsoft Internet Explorer (IE) 11 (64-bit with Enhanced Protected Mode(EPM)-enabled): $65,000
- Mozilla Firefox: $30,000
- Adobe Reader running in Internet Explorer 11 (64-bit with EPM-enabled): $60,000
- Adobe Flash (64-bit) running in Internet Explorer 11 (64-bit with EPM-enabled): $60,000

Mac OS X-based targets:

- Apple Safari (64-bit): $50,000

http://www.zdnet.com/article/pwn2own-2015-the-year-every-browser-went-down/

Pwn2own 2015

Pwn2Own 2015: The year every web browser went down

Every major web browser showed up, every web browser got hacked.

By Steven J. Vaughan-Nichols for Networking | March 23, 2015 -- 11:53 GMT (11:53 GMT) | Topic: Security

http://www.zdnet.com/article/pwn2own-2015-the-year-every-browser-went-down/

Pwn2own 2015

Pwn2Own 2015: The year every web browser went down

Every major web browser showed up, every web browser got hacked.

By Steven J. Vaughan-Nichols for Networking | March 23, 2015 -- 11:53 GMT (11:53 GMT) | Topic: Security

Making its Pwn2Own debut, the 360Vulcan team compromised the 64-bit version of IE 11 with an uninitialized memory vulnerability in 17 seconds, bringing them $32,500 in prize money. The browser was guarded by an enhanced sandbox, full 64-bit process, Microsoft EMET, Windows 8.1 security mechanisms and other protections.

http://www.zdnet.com/article/pwn2own-2015-the-year-every-browser-went-down/
http://searchsecurity.techtarget.com/news/2240242771/At-2015-Pwn2Own-competition-browser-exploits-in-the-spotlight

The weakest link

Human factor	Reachability	Vulnerable software
User interaction is crucial	Required	Enabler
Ordinary level of interaction expected	Offline client systems make no sense	Bugs exist, testing is not enough, software systems are complex

Client-side attack countermeasures

Human factor	Reachability	Vulnerable software
Awareness	Reduce attack surface	Patch software and systems
Awareness	Monitoring	Use client-side hardening!
Support the user with technical measures		Deploy proactive protections
Awareness		

Thanks!

antonio.barresi@inf.ethz.ch

or

antonio@barresi.net

www.antoniobarresi.com