Security: Empirical measures and theoretical bases

Andrew Ruef

EVERYTHING IS HACKED

RISK ASSESSMENT / SECURITY & HACKTIVISM

All four major browsers tak
Pwn2Own hacking competi
Security defenses keep getting better, but so too do hac

Schneier on Security

Blog | Newsletter | Books | Essays | News | Schedule | Crypto | About Me

← "Unbreakable" Encryption Almost Certainly Isn't Police Disabling Their Own Voice Recorders →

Heartbleed

Heartbleed is a catastrophic bug in OpenSSL:

Maryland Hacked Again: 4 Weeks
March 20, 2014 2:41 PM

Security on NBCNEWS.com

Heartland Payment Systems hacked

Payments processor said breach did not involve merchant, cardholder data

Target credit card hack: What you need to know

What can science do

- Measure the world

- Develop theories

MEASURE THE WORLD

Hackers are not tornados

Spam

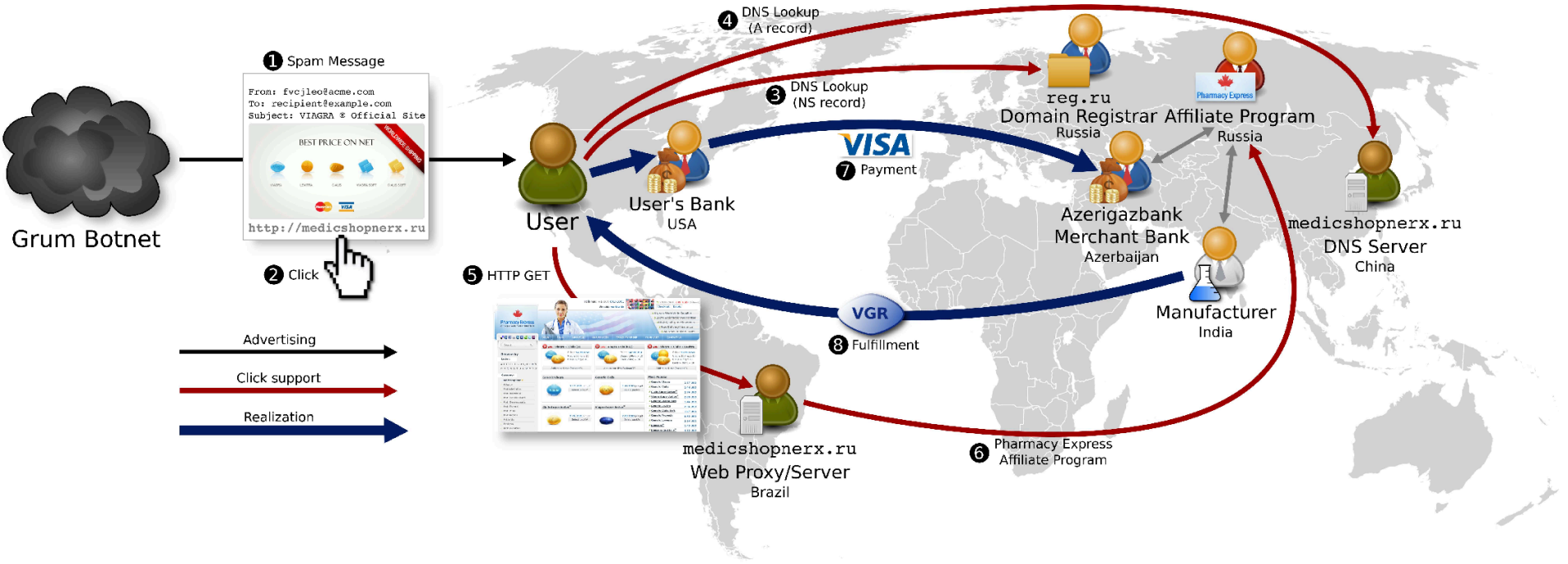


Figure 1: Infrastructure involved in a single URL's value chain, including advertisement, click support and realization steps.

Zeus

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF NEW YORK

- -

MICROSOFT CORP., FS-ISAC, INC., and NATIONAL AUTOMATED CLEARING HOUSE ASSOCIATION,

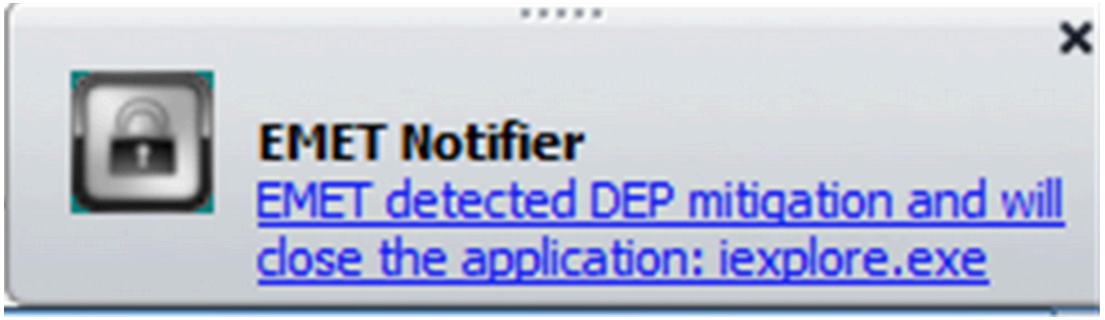
Plaintiffs

v.

JOHN DOES 1-39 D/B/A Slavik, Monstr, IOO, Nu11, nvidiag, zebra7753, lexa_Mef, gss, iceIX, Harderman, Gribodemon, Aqua, aquaSecond, it, percent, cp01, hct, xman, Pepsi, miami, miamibc, petr0vich, Mr. ICQ, Tank, tankist, Kusunagi, Noname, Lucky, Bashorg, Indep, Mask, Enx, Benny, Bentley, Denis Lubimov, MaDaGaSka, Vkontake, rfcid, parik, reronic, Daniel, bx1, Daniel Hamza, Danielbx1, jah, Jonni, jtk, D frank, duo, Admin2010, h4x0rdz, Donsft, mary.J555, susanneon, kainehabe, virus_e_2003, spanishp, sere.bro, muddem, mechan1zm, vlad.dimitrov, jheto2002, sector.exploits AND JabberZeus Crew AND YEVHEN KULIBABA AND YURIY KONOVALENKO, CONTROLLING COMPUTER BOTNETS THEREBY INJURING PLAINTIFFS, AND THEIR CUSTOMERS AND MEMBERS,

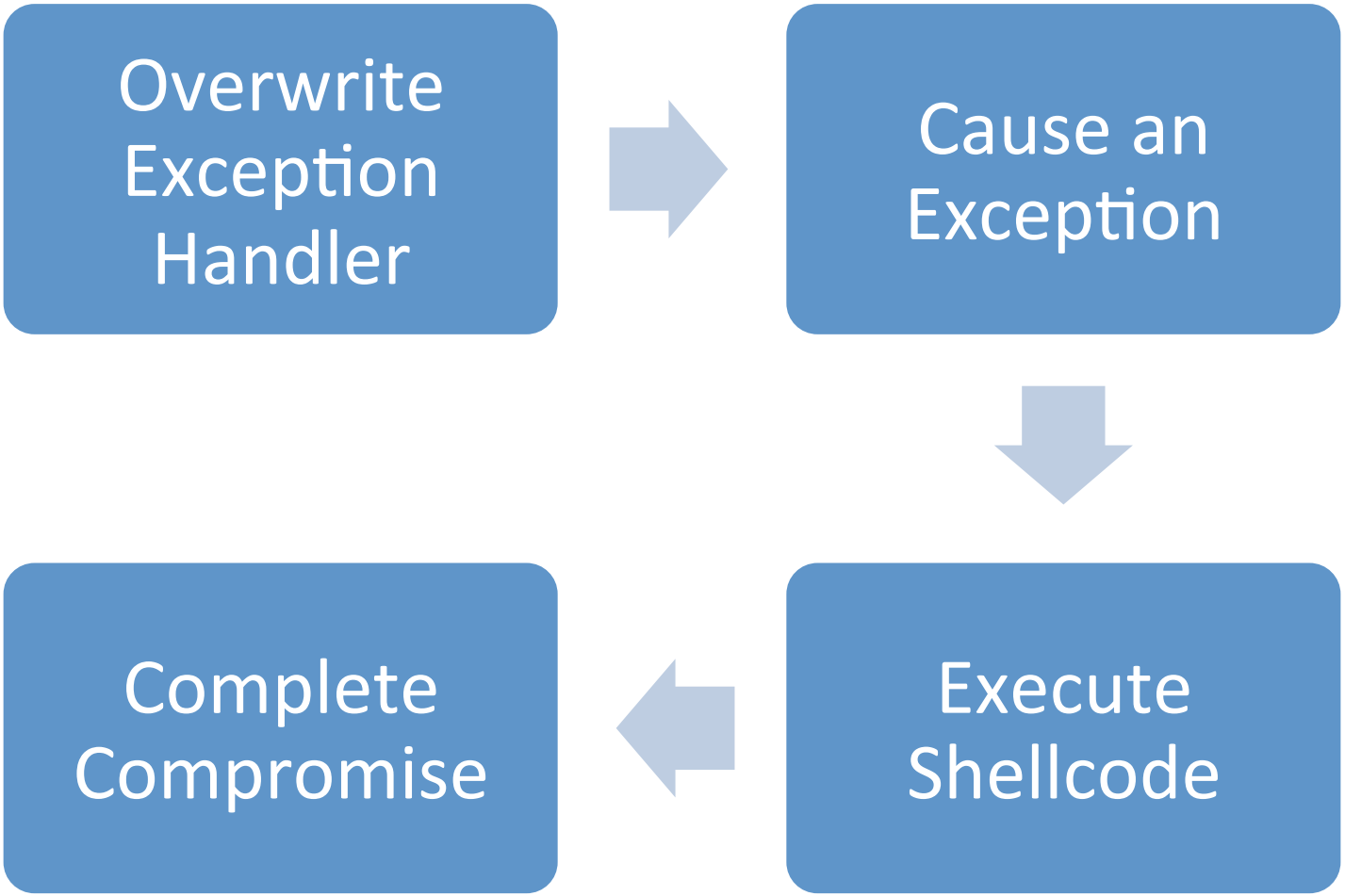
Defendants.

Develop mitigations

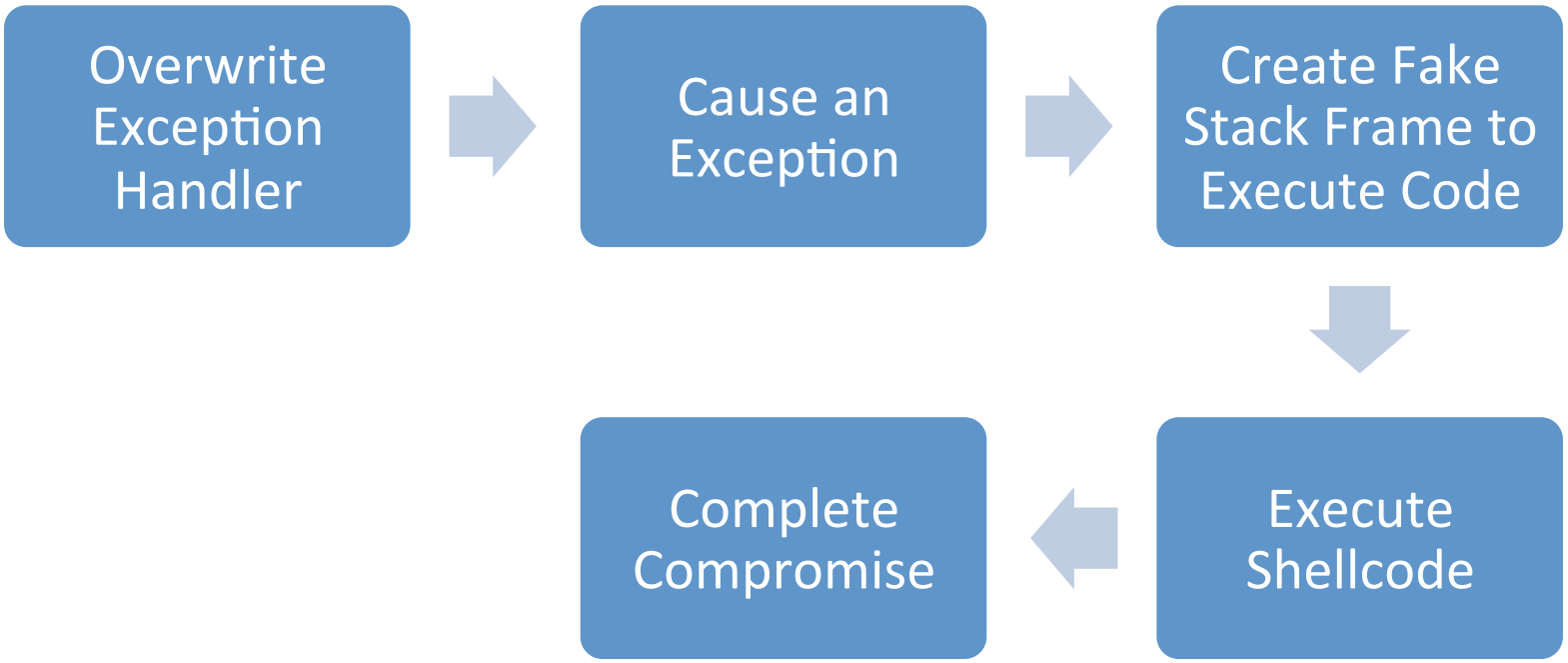


Problem (1996)

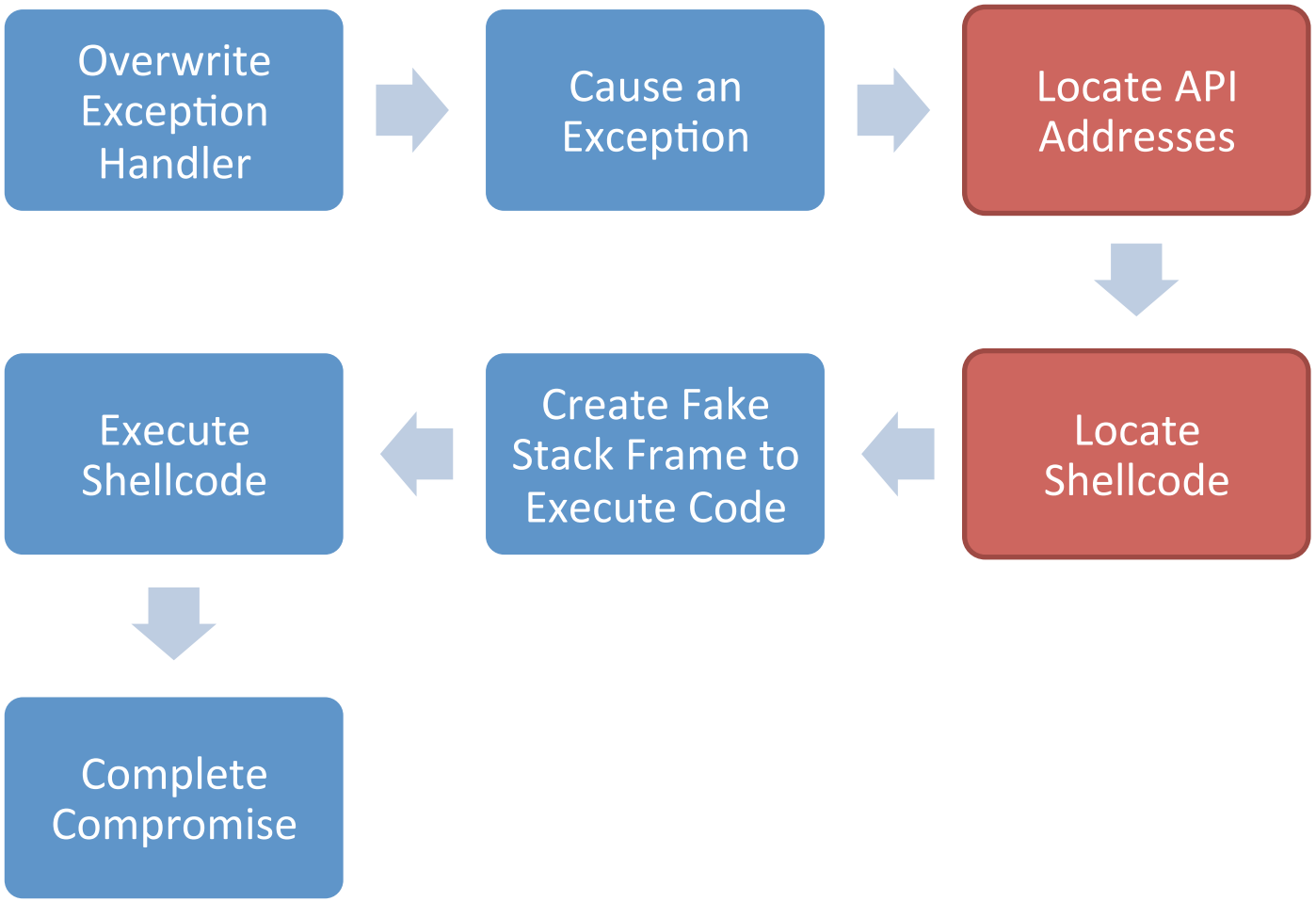
Problem (2003)



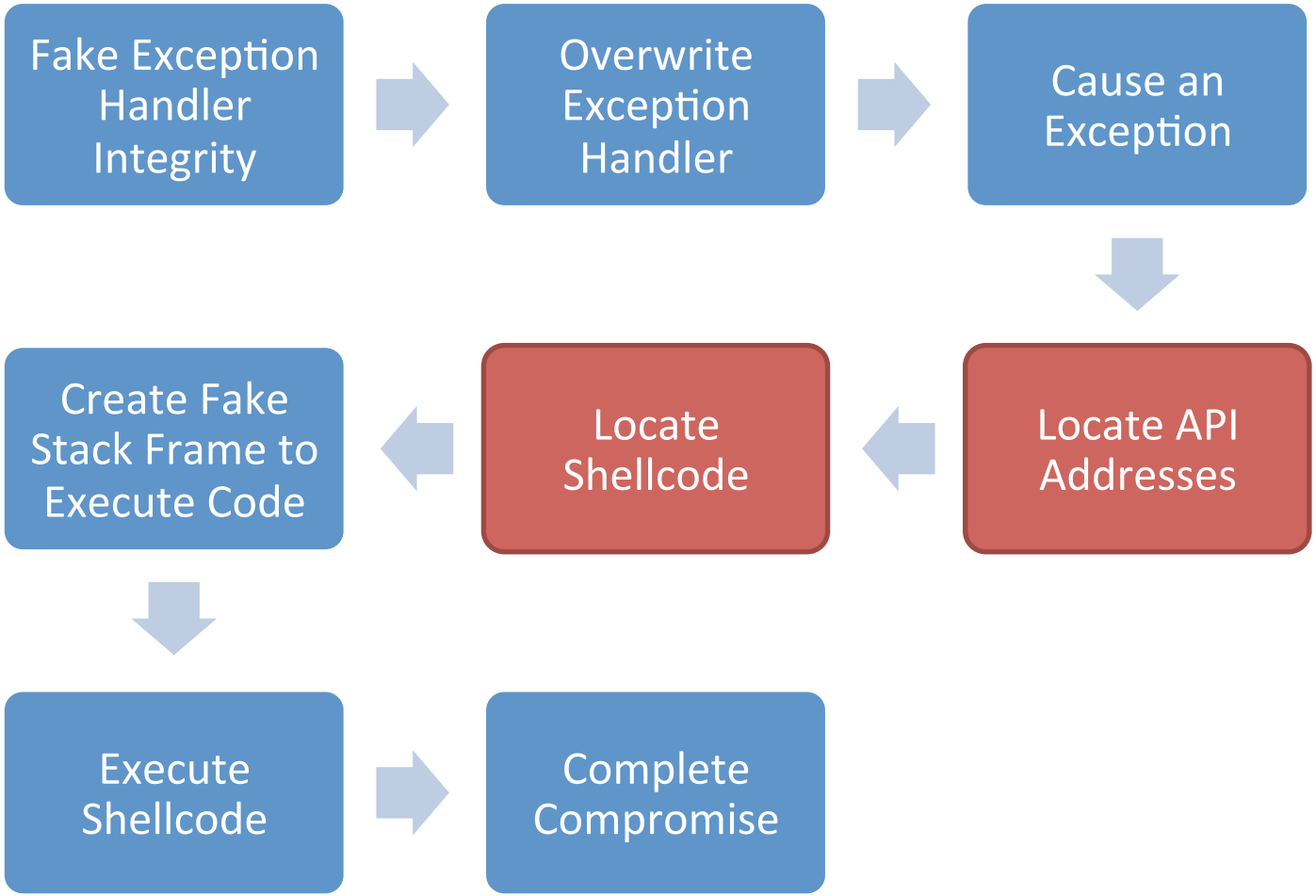
Problem (2004)



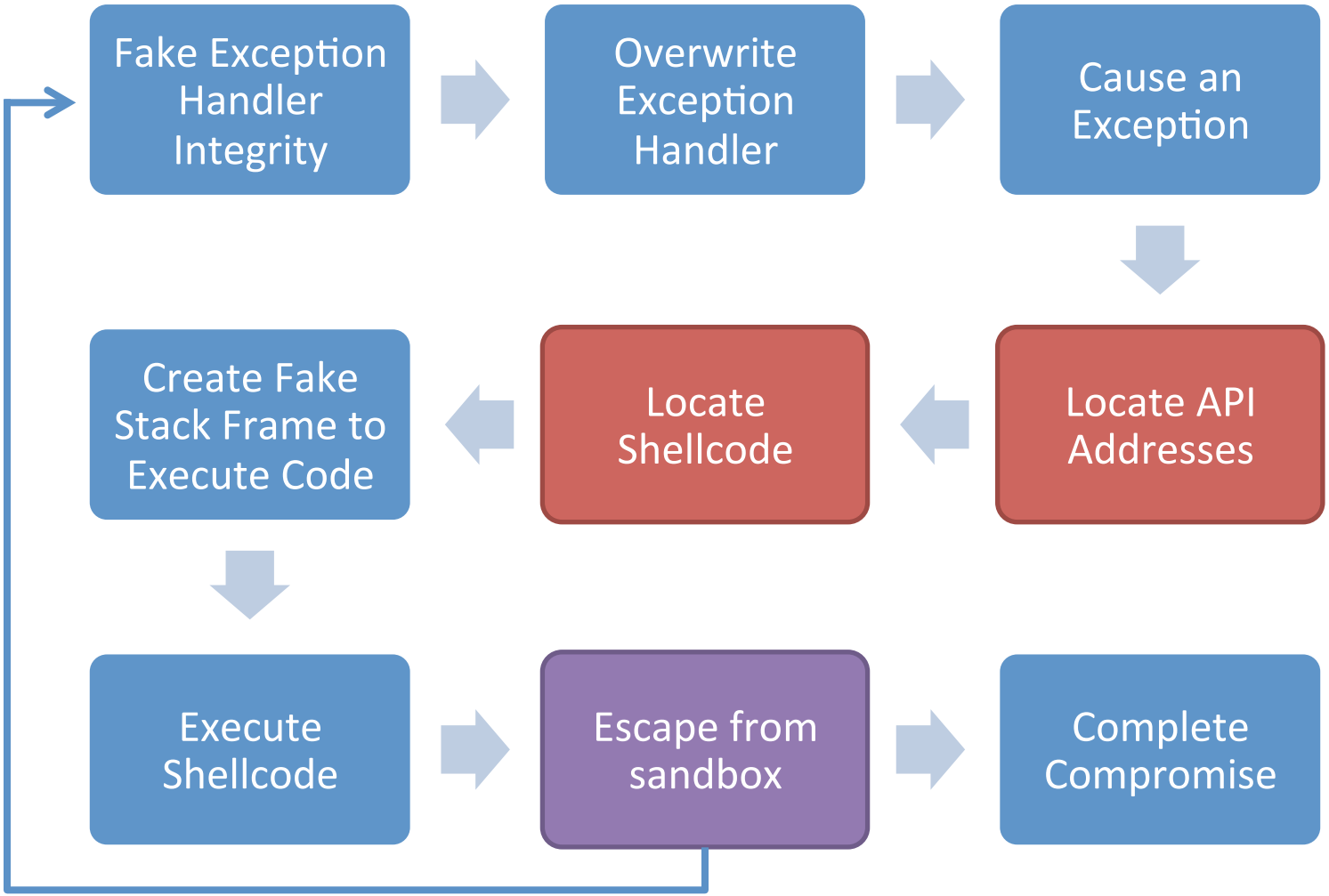
Problem (2007)



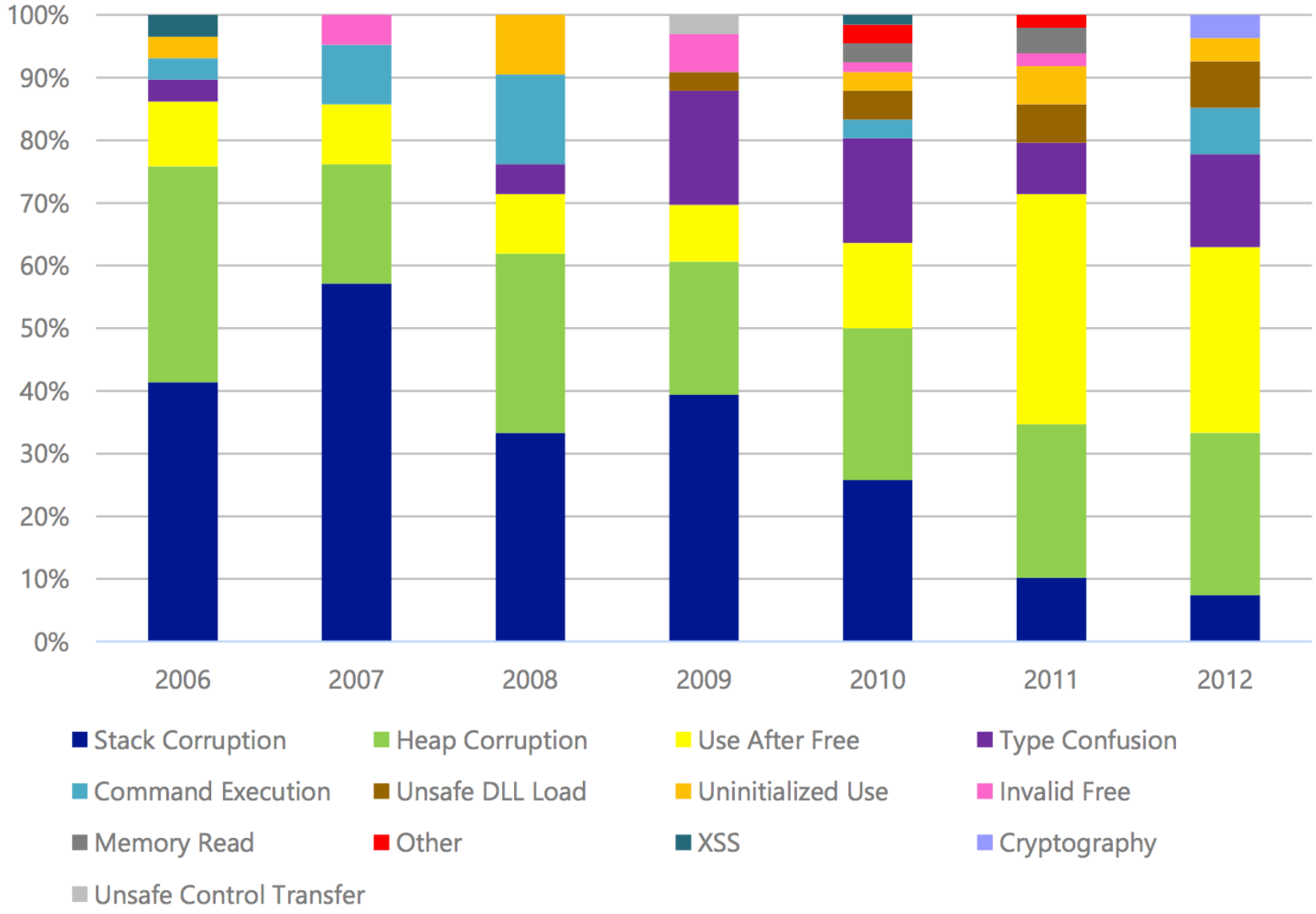
Problem (2008)



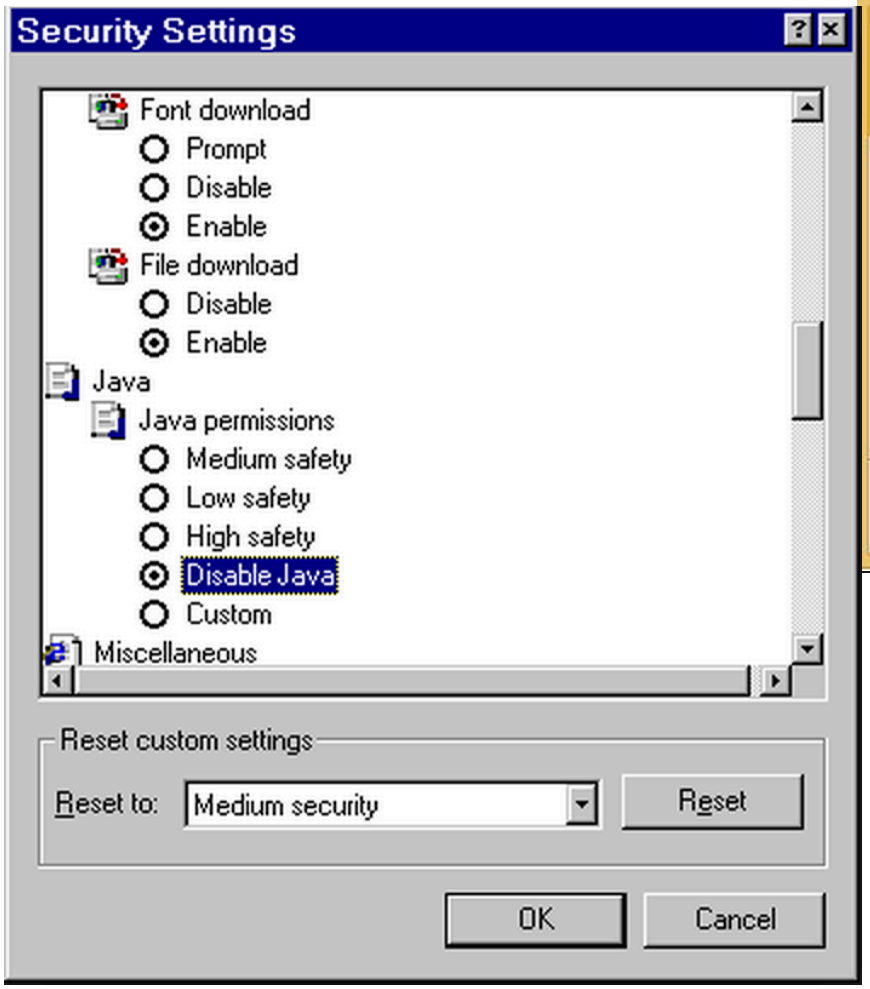
Problem (2013)

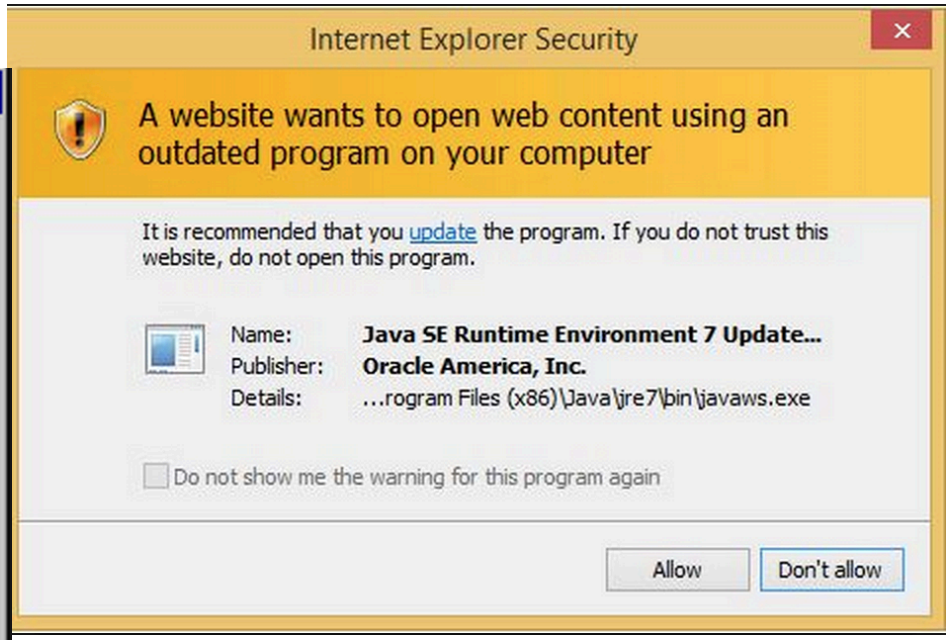


IT WORKS

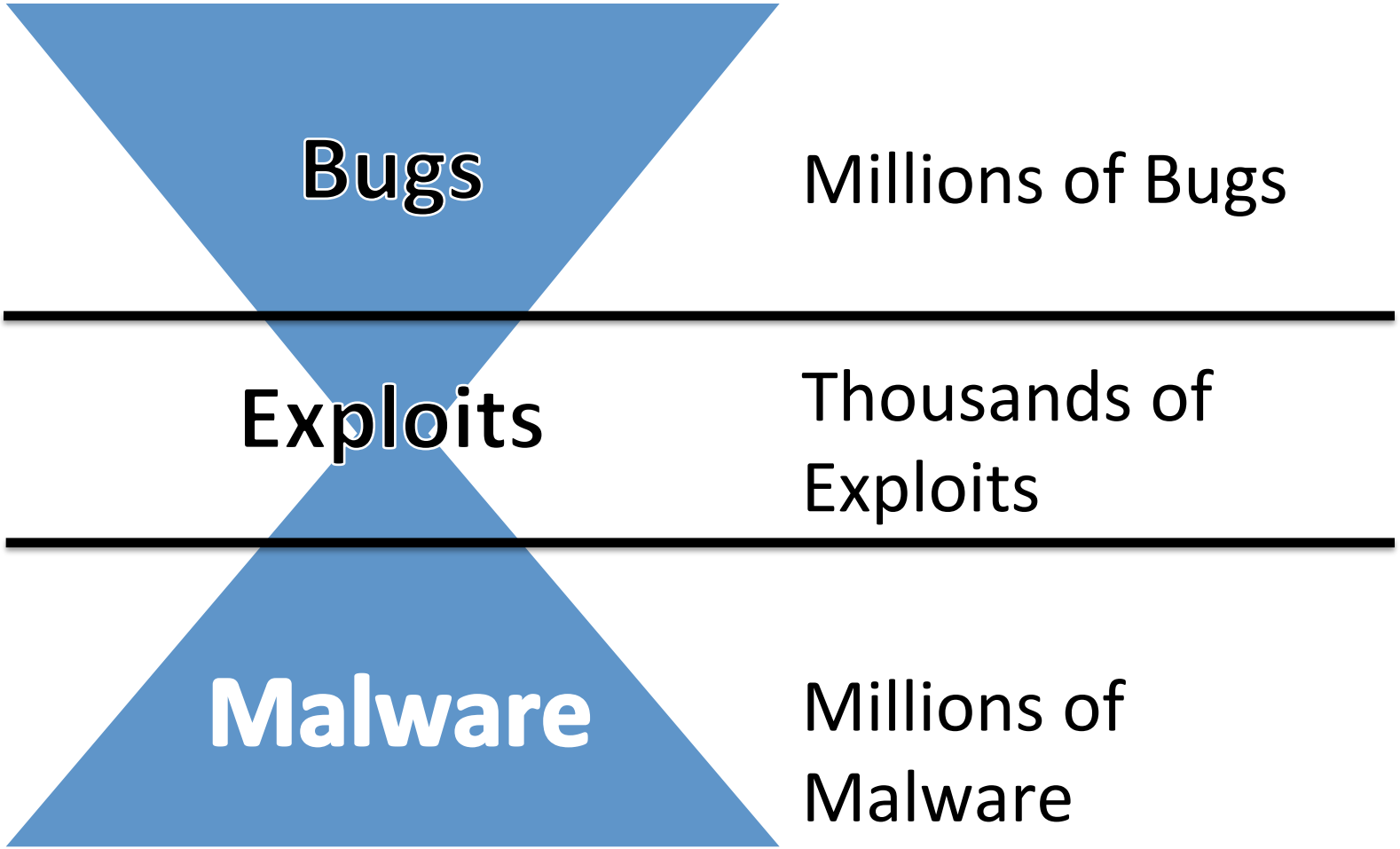

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
2006
2007
2008
2009
2010
2011
2012
Stack Corruption
Heap Corruption
Use After Free
Type Confusion
Command Execution
Unsafe DLL Load
Uninitialized Use
Invalid Free
Memory Read
Other
XSS
Cryptography
Unsafe Control Transfer

Enterprise configuration

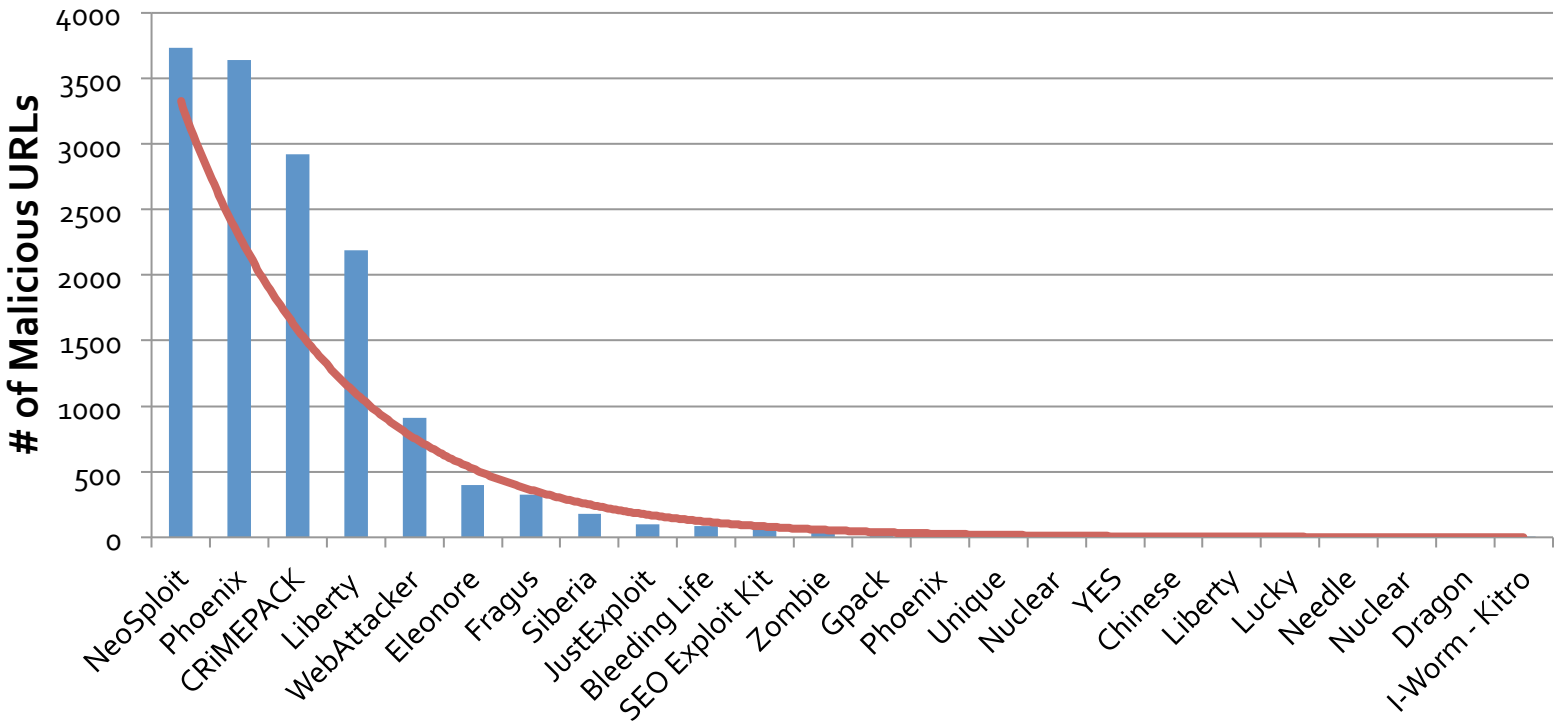




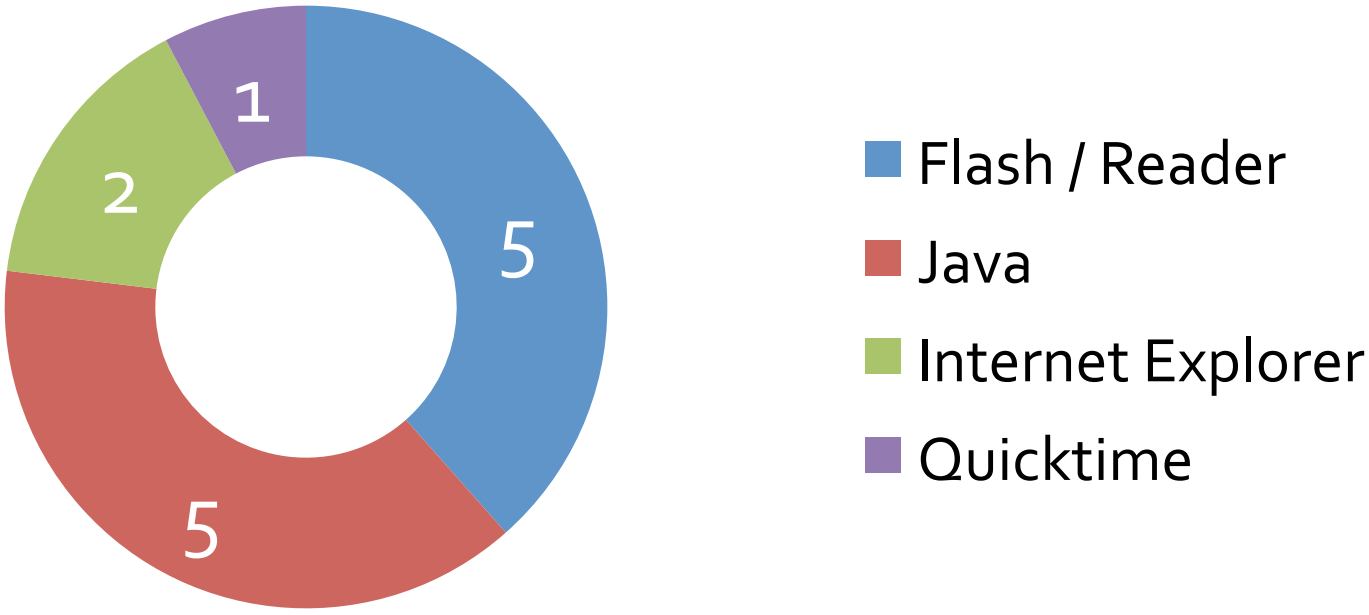
Why Study Exploitation?



Clear Market Leaders



Limited Target Support

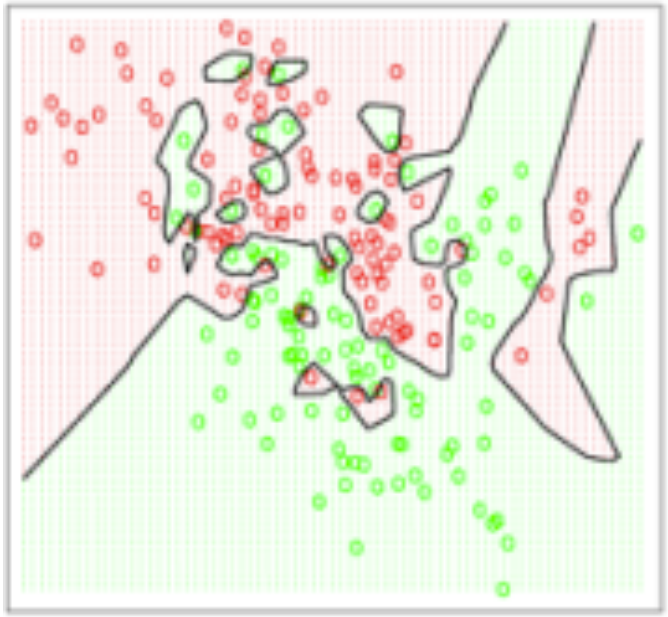


Low Quality Exploits

Memory Corruption (19)	
Defeated by DEP	14
Defeated by ASLR	17
Defeated by EMET	19

Logic Flaws (8)	
No Java in Internet Zone	4
No EXEs in PDFs	1
No Firefox or FoxIt Reader	2

Empiricism can be dangerous



still teaching computer janitors of tomorrow to lose the battles of yesterday
drive.google.com/file/d/0B_mjsP…

Google Docs

DEVELOP THEORIES

Full abstraction

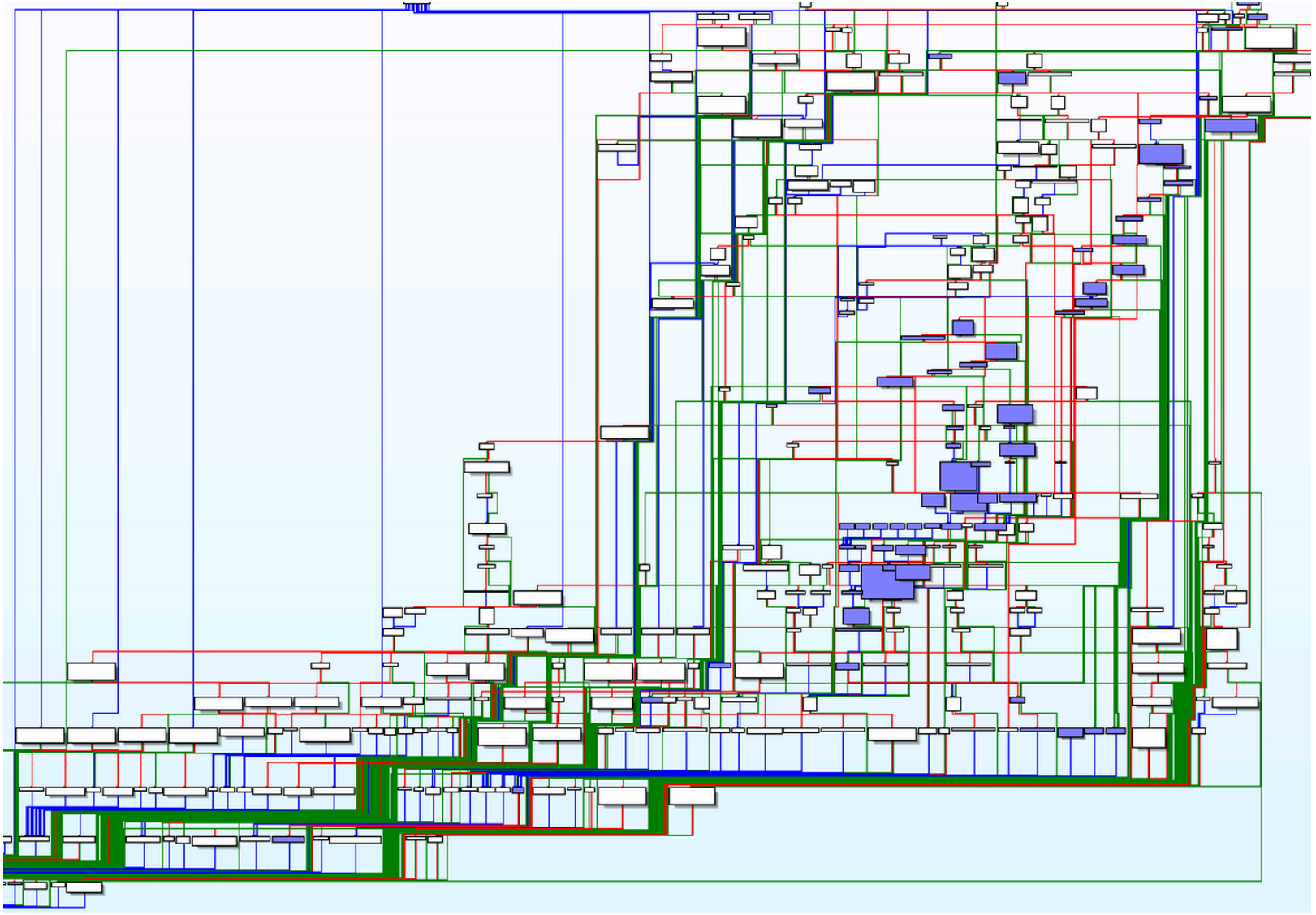
```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <arpa/inet.h>

void serveur1(portServ ports)
{
    int sockServ1, sockServ2, sockClient;
    struct sockaddr_in monAddr, addrClient, addrServ2;
    socklen_t lenAddrClient;

    if ((sockServ1 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }
    if ((sockServ2 = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Erreur socket");
    exit(1);
    }

    bzero(&monAddr, sizeof(monAddr));
    monAddr.sin_family = AF_INET;
    monAddr.sin_port = htons(ports.port1);
    monAddr.sin_addr.s_addr = INADDR_ANY;
    bzero(&addrServ2, sizeof(addrServ2));
```

Full abstraction



Control Flow Integrity

```
bool lt(int x, int y) {
    return x < y;
}

bool gt(int x, int y) {
    return x > y;
}

sort2(int a[], int b[], int len)
{
    sort( a, len, lt );
    sort( b, len, gt );
}
```

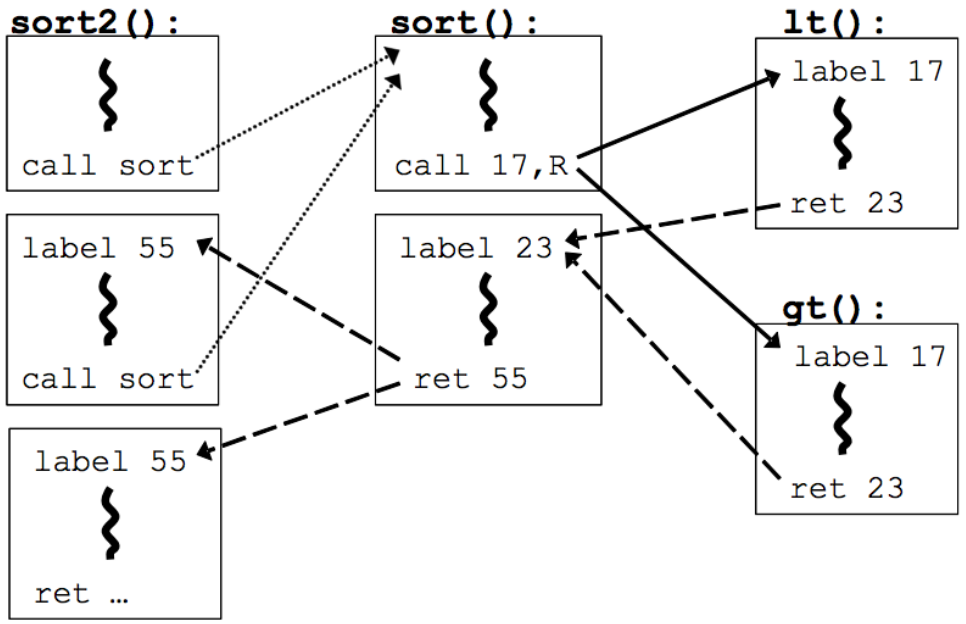
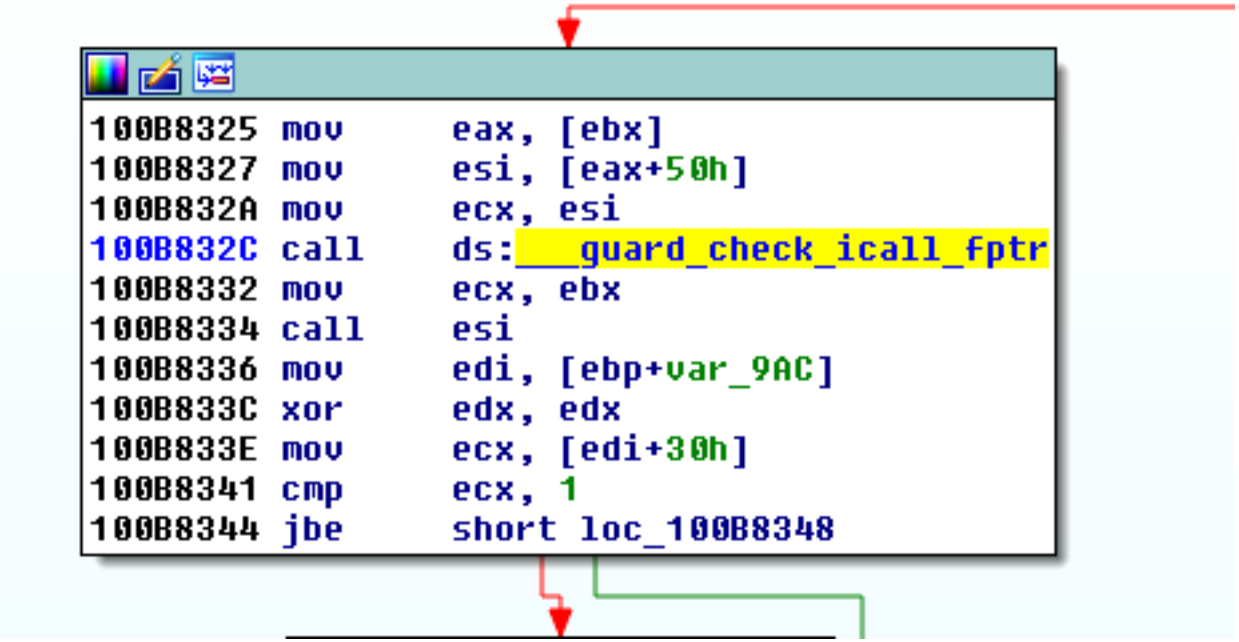
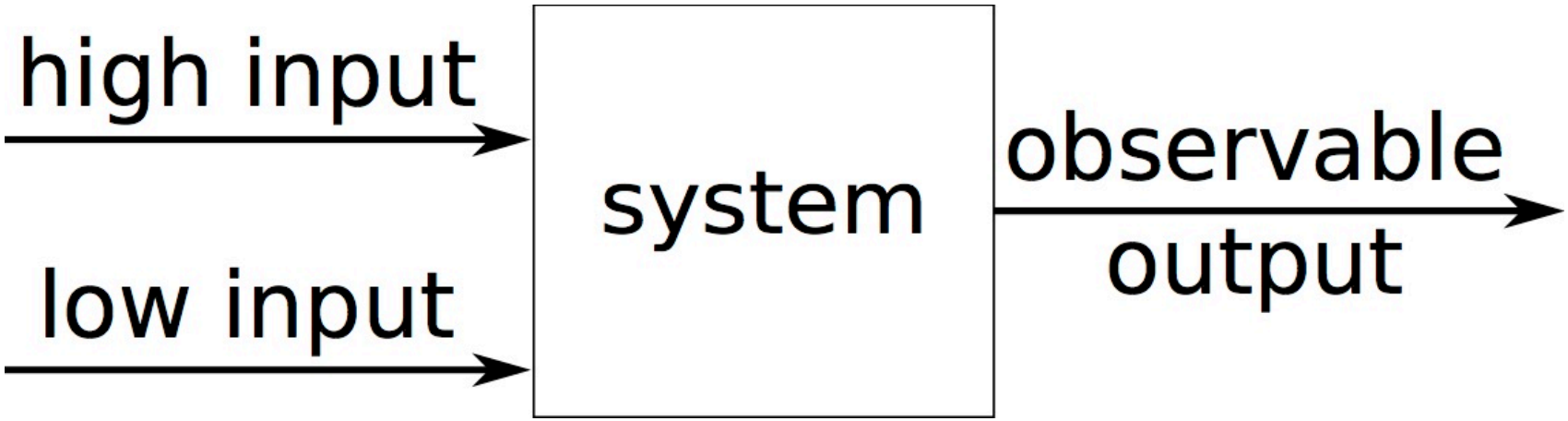


Figure 1: Example program fragment and an outline of its CFG and CFI instrumentation.

Control Flow Integrity (CFI)



Noninterference



Enforcing noninterference

[E1–2] $\vdash exp : high$ $\qquad \dfrac{h \notin Vars(exp)}{\vdash exp : low}$

[C1–3] $[pc] \vdash \mathsf{skip}$ $\qquad [pc] \vdash h := exp$ $\qquad \dfrac{\vdash exp : low}{[low] \vdash l := exp}$

[C4–5] $\dfrac{[pc] \vdash C_1 \quad [pc] \vdash C_2}{[pc] \vdash C_1; C_2}$ $\qquad \dfrac{\vdash exp : pc \qquad [pc] \vdash C}{[pc] \vdash \mathsf{while}\ exp\ \mathsf{do}\ C}$

[C6–7] $\dfrac{\vdash exp : pc \quad [pc] \vdash C_1 \quad [pc] \vdash C_2}{[pc] \vdash \mathsf{if}\ exp\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2}$ $\qquad \dfrac{[high] \vdash C}{[low] \vdash C}$

Verification

IT WORKS

development version of CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task. The apparent unbreakability of CompCert supports a strong argument that developing compiler optimizations within a proof framework, where safety checks are explicit and machine-checked, has tangible benefits for compiler users. **(PLDI 2011)**

Better testing (Heartbleed)

```
2561        /* Read type and payload length first */
2562        hbtype = *p++;
2563        n2s(p, payload);
2564    p += 2;
2565        pl = p;
2566
2567        if (s->msg_callback)
```

1 Taking false branch →

```
2568                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
2569                        &s->s3->rrec.data[0], s->s3->rrec.length,
2570                        s, s->msg_callback_arg);
2571
2572        if (hbtype == TLS1_HB_REQUEST)
```

2 ← Assuming 'hbtype' is equal to 1 →

3 ← Taking true branch →

Better testing (Heartbleed)

```
2581        buffer = OPENSSL_malloc(1 + 2 + payload + padding);
2582        bp = buffer;
2583
2584        /* Enter response type, length and copy payload */
2585        *bp++ = TLS1_HB_RESPONSE;
2586        s2n(payload, bp);
2587        memcpy(bp, pl, payload);
```

4 ← Tainted, unconstrained value used in memcpy size

CONCLUSION

Measure things

- It can tell you what you should focus on
- It can tell you when what you are doing is working

Develop theories

- Understand and explain phenomenon
- Build systems right the first time