Nozzle:

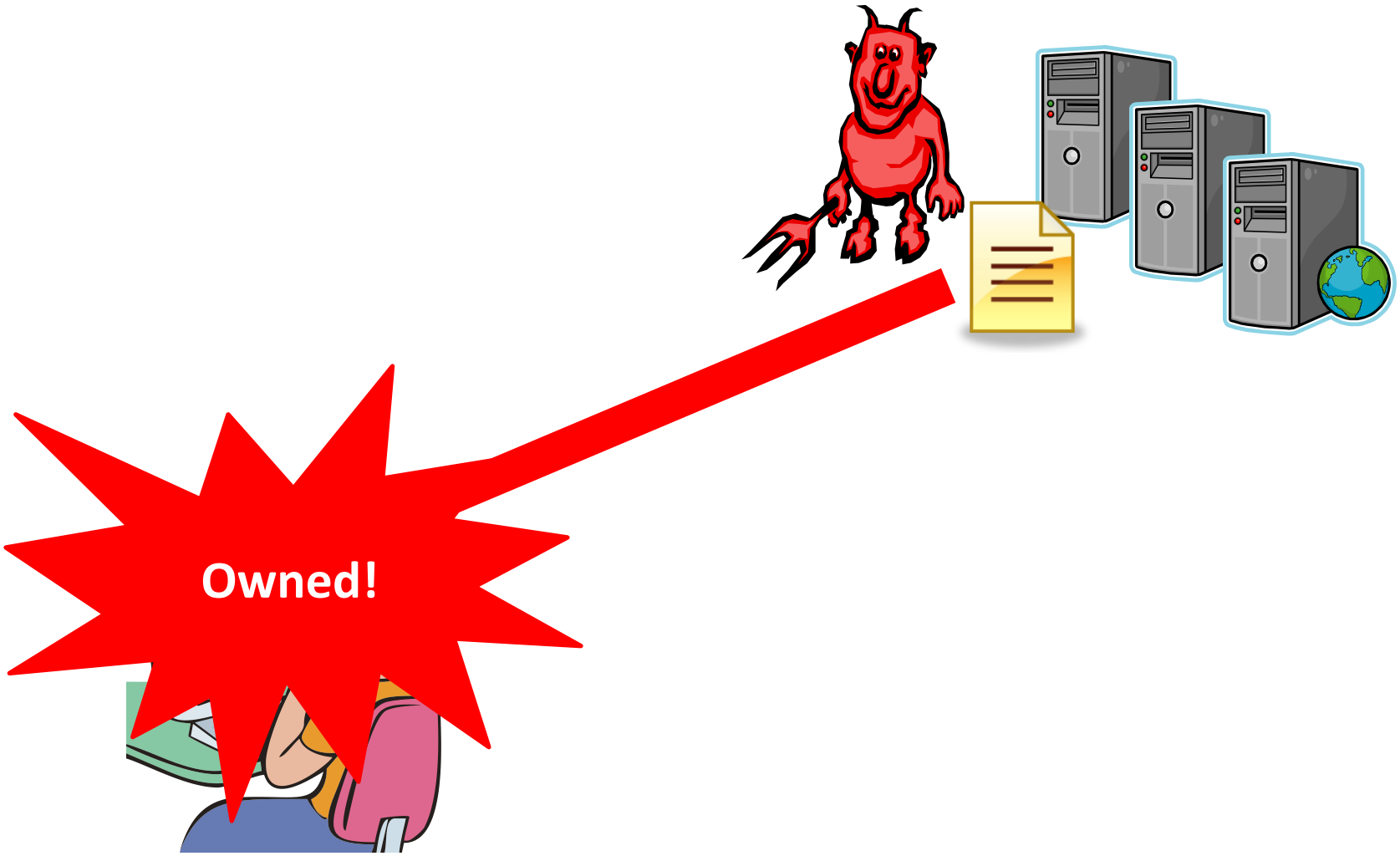
A Defense Against Heap-spraying Code Injection Attacks

Paruj Ratanaworabhan, Cornell University

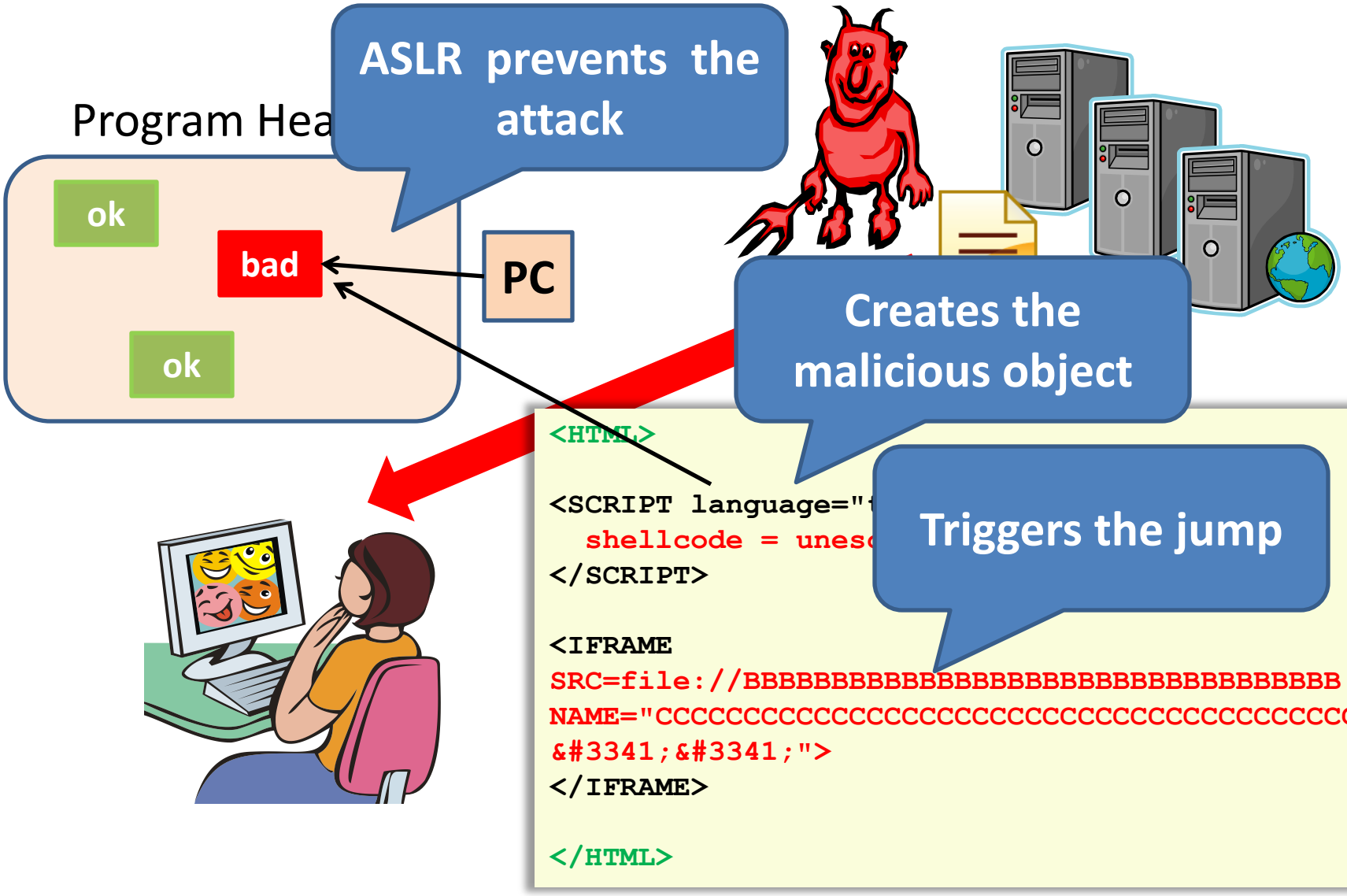
Ben Livshits and Ben Zorn, Microsoft Research (Redmond, WA)

Heap Spraying is a Problem

Drive-By Heap Spraying

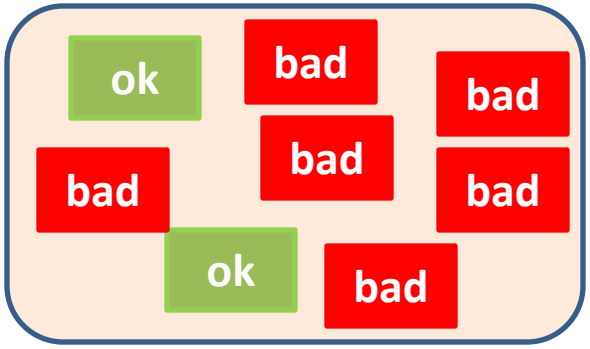


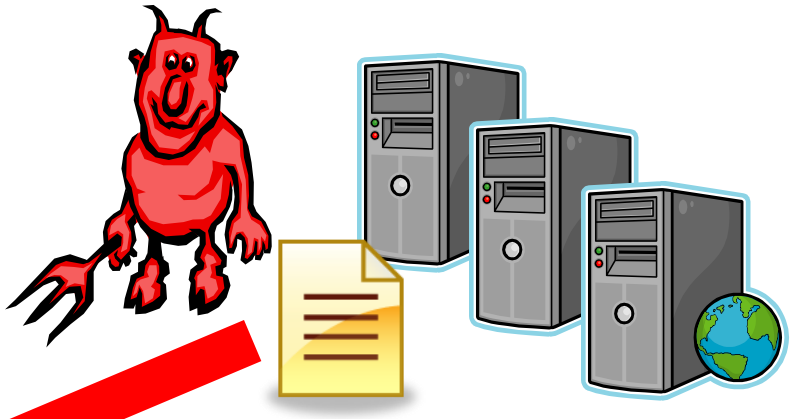
Drive-By Heap Spraying (2)



Drive-By Heap Spraying (3)

Program Heap





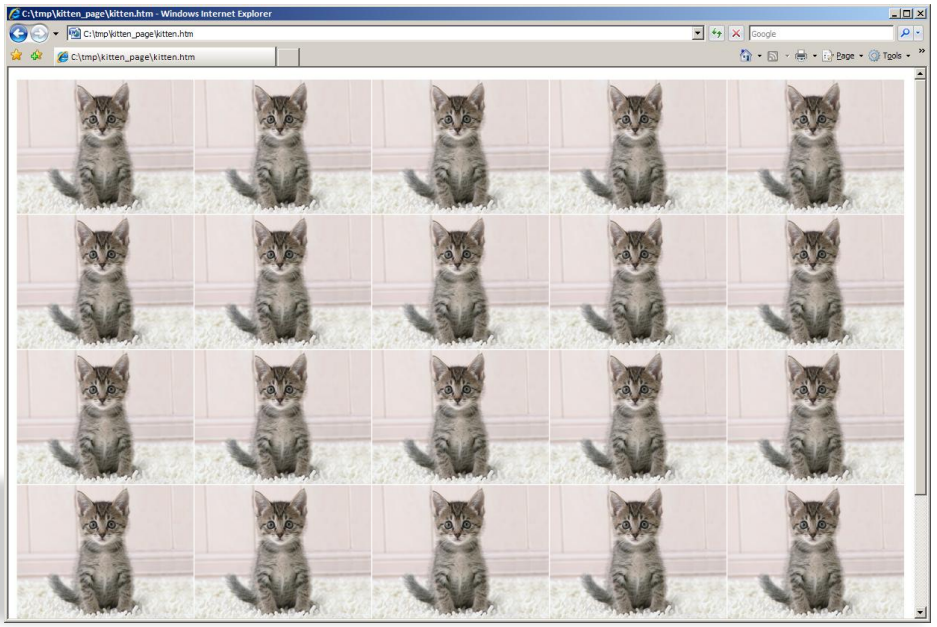
```
<SCRIPT language="text/javascript">
  shellcode = unescape("%u4343%u4343%...");
  oneblock = unescape
  var fullblock = on
  while (fullblock.l
    fullblock += ful
  }

  sprayContainer = new Array();
  for (i=0; i<1000; i++) {
    sprayContainer[i] = fullblock + shellcode;
  }
</SCRIPT>
```

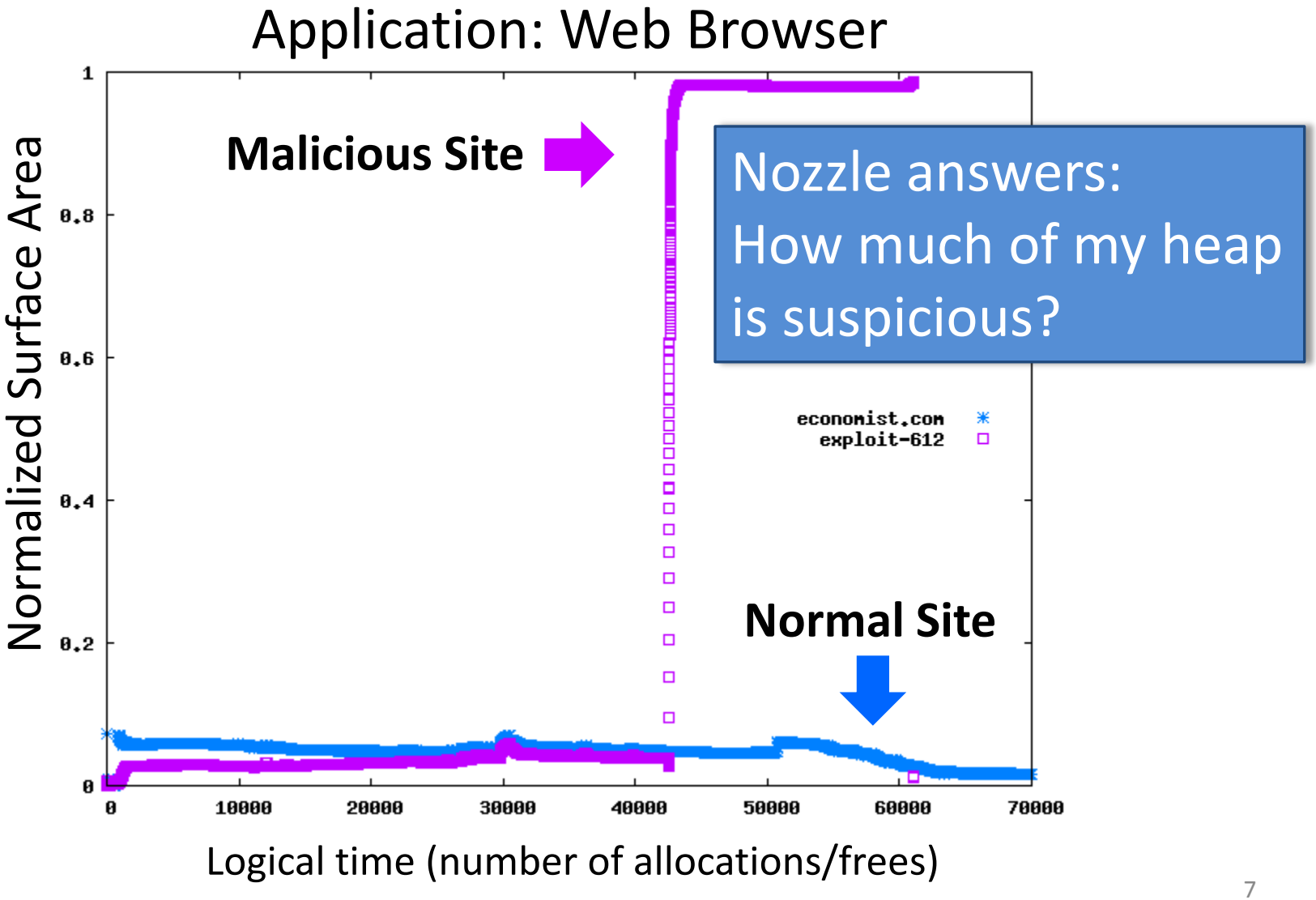
Allocate 1000s of malicious objects

Kittens of Doom
What data can you trust?

- Heap spraying is quite general, easy to implement
- Many applications allow scripts in type safe languages
  - JavaScript, ActionScript
  - Java, C#
- Many applications accept data from untrusted sources
  - Embed malicious code in images, documents, DLLs, etc.
- [Sotirov & Dowd BH'08]



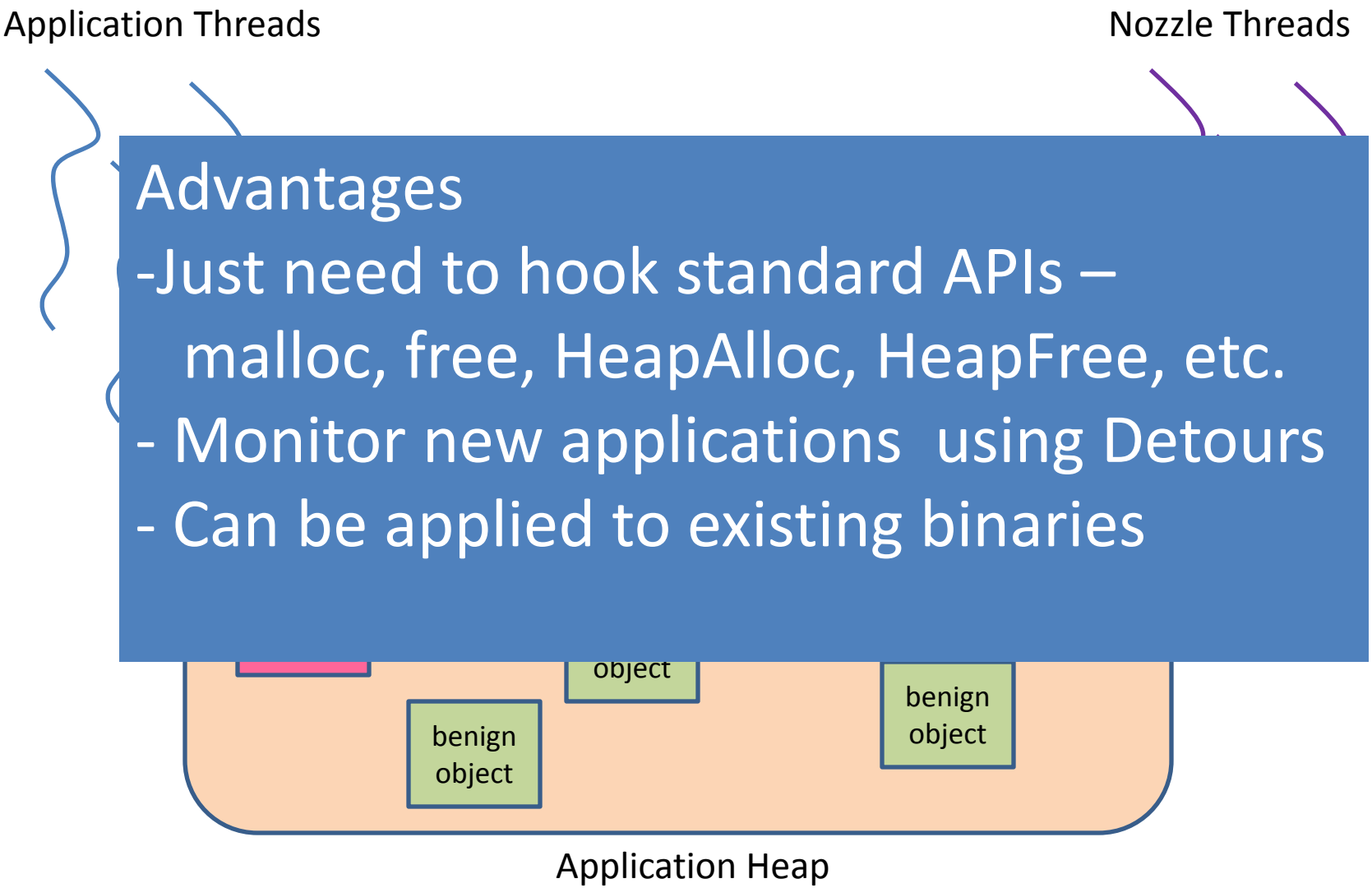
Nozzle – Runtime Heap Spraying Detection



Outline

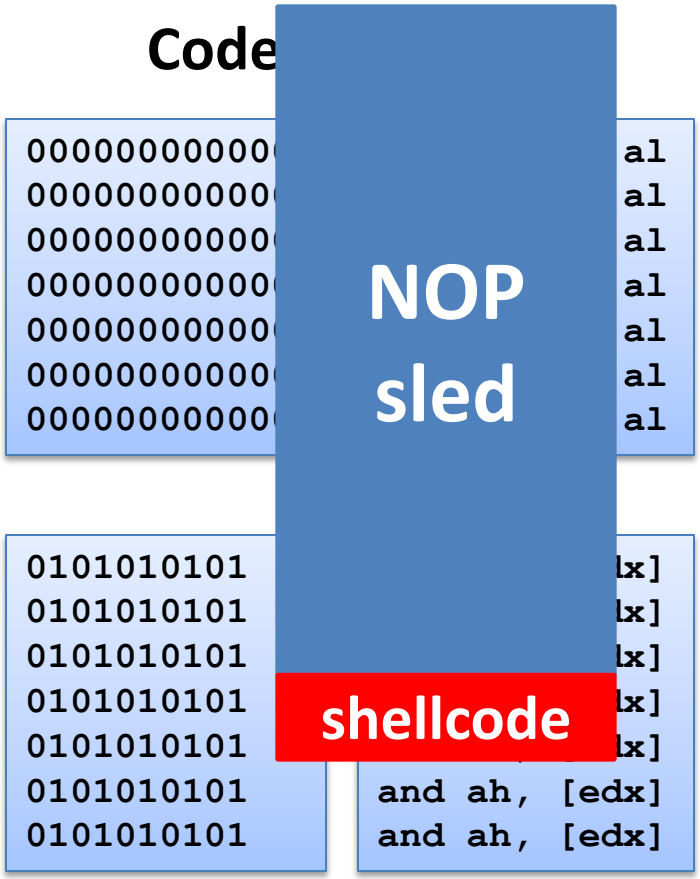
- Nozzle design & implementation
- Evaluation
  - False positives
  - False negatives
  - New threats (Adobe Reader)
- Summary

Nozzle Design



Local Malicious Object Detection

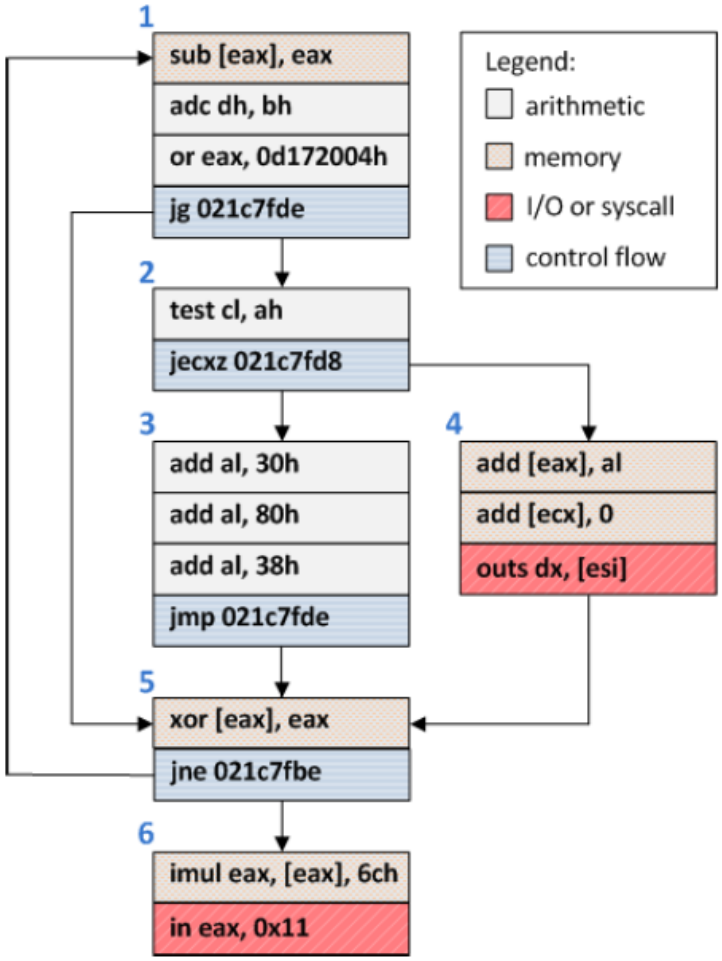
Is this object dangerous?



- 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)

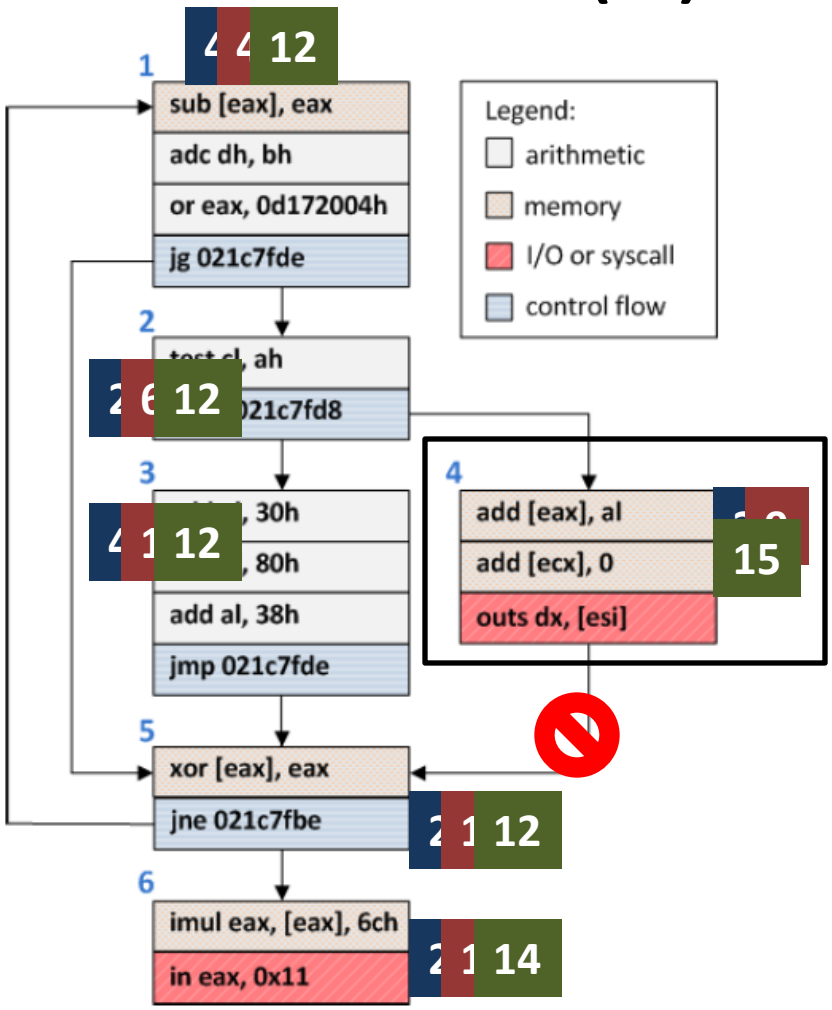
- Assume: attacker wants to reach shell code from jump to any point in object
- Goal: find blocks that are likely to be reached via control flow
- Strategy: use dataflow analysis to compute "surface area" of each block



An example object from visiting google.com

Object Surface Area Calculation (2)

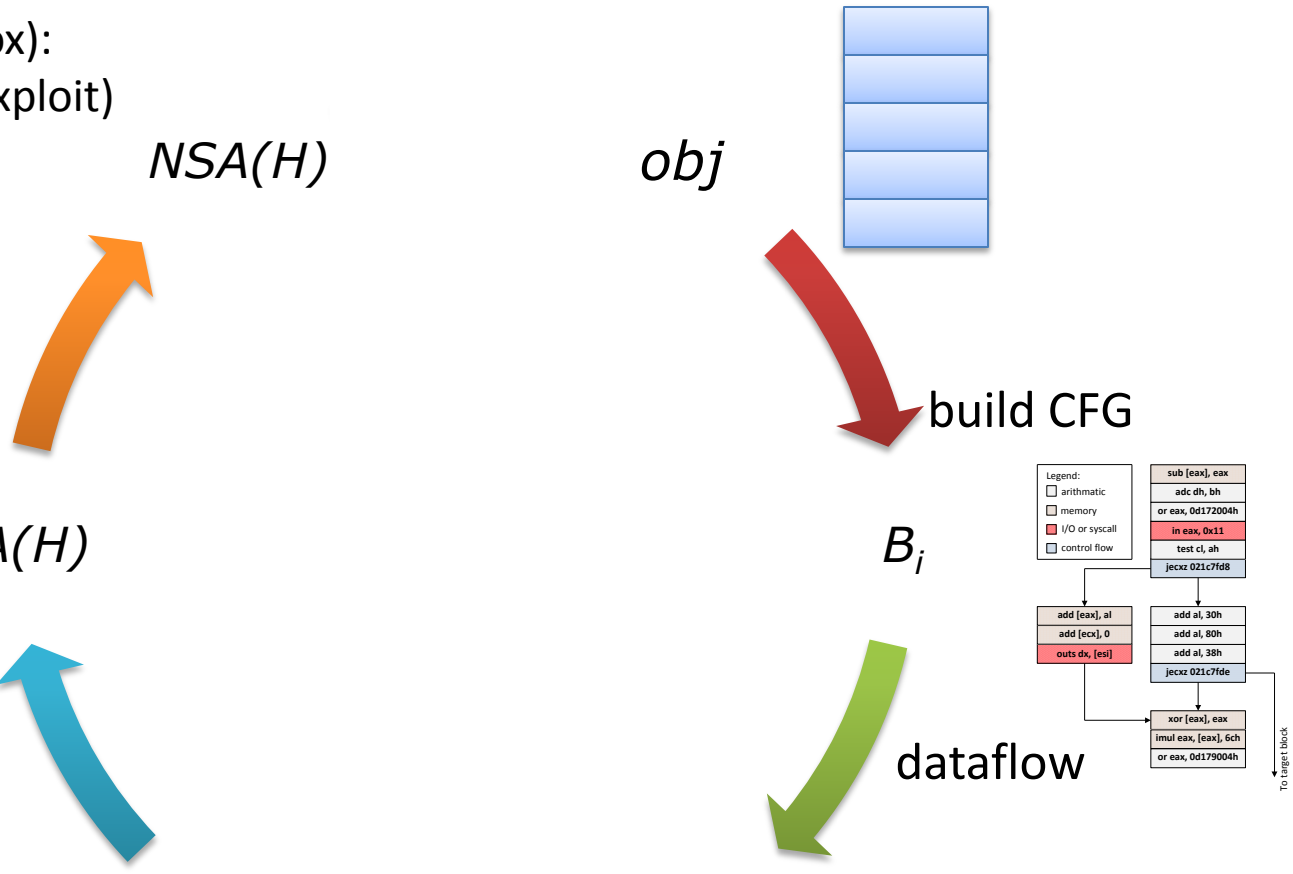
- Each block starts with its own size as weight
- Weights are propagated forward with flow
- Invalid blocks don't propagate
- Iterate until a fixpoint is reached
- Compute block with highest weight



An example object from visiting google.com

Nozzle Global Heap Metric

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



Nozzle Experimental Summary

0 False Positives

- 10 popular AJAX-heavy sites
- 150 top Web sites

0 False Negatives

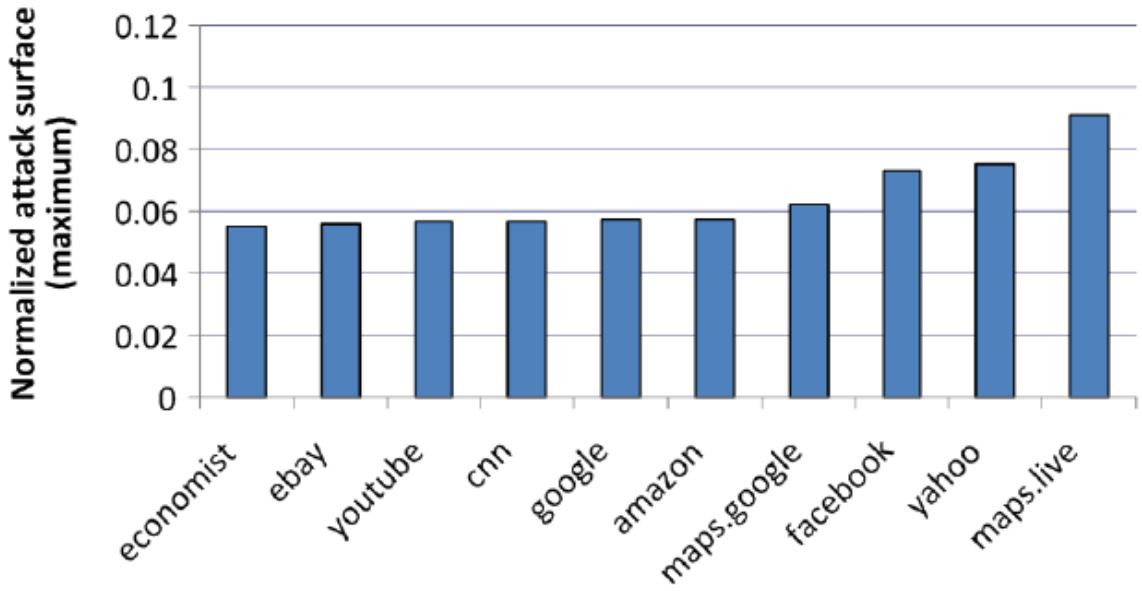
- 12 published heap spraying exploits and
- 2,000 synthetic rogue pages generated using Metasploit

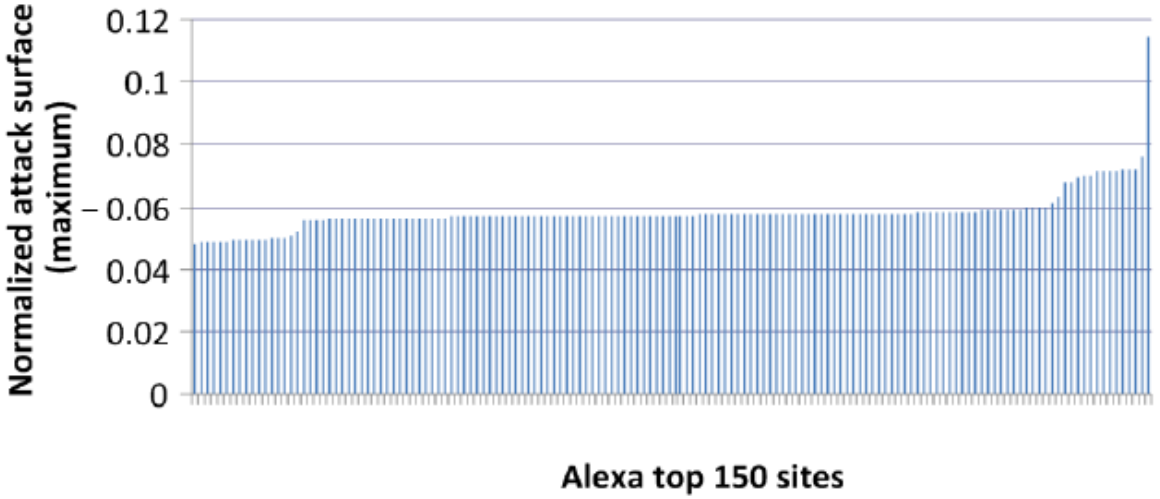
Runtime Overhead

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

Nozzle on Benign Sites

- Benign sites have low Nozzle NSA
- Max NSA always less than 12%
- Thresholds can be set much higher for detection (50% or more)





Nozzle with Known Heap Sprays

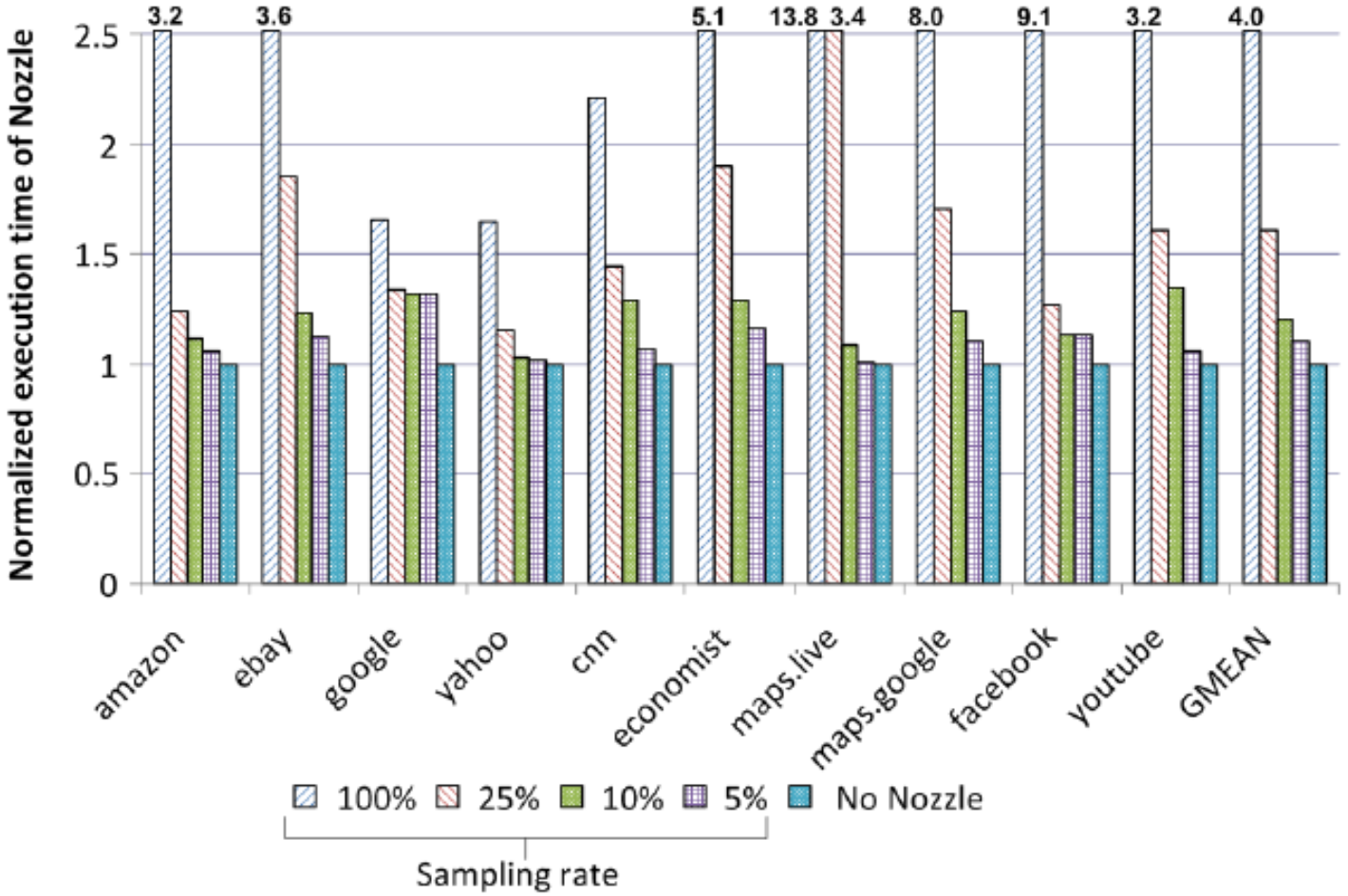
- 12 published heap spray pages in multiple browsers
- 2,000 synthetic heap spray pages using MetaSploit
  - advanced NOP engine
  - shellcode database

Date	Browser	Description	milw0rm
11/2004	IE	IFRAME Tag BO	612
04/2005	IE	DHTML Objects Corruption	930
01/2005	IE	.ANI Remote Stack BO	753
07/2005	IE	`javaprxy.dll` COM Object	1079
03/2006	IE	`createTextRang` RE	1606
09/2006	IE	VML Remote BO	2408
03/2007	IE	ADODB Double Free	3577
09/2006	IE	WebViewFolderIcon `setSlice`	2448
09/2005	FF	0xAD Remote Heap BO	1224
12/2005	FF	`compareTo()` RE	1369
07/2006	FF	`Navigator` Object RE	2082
07/2008	Safari	Quicktime Content-Type BO	6013

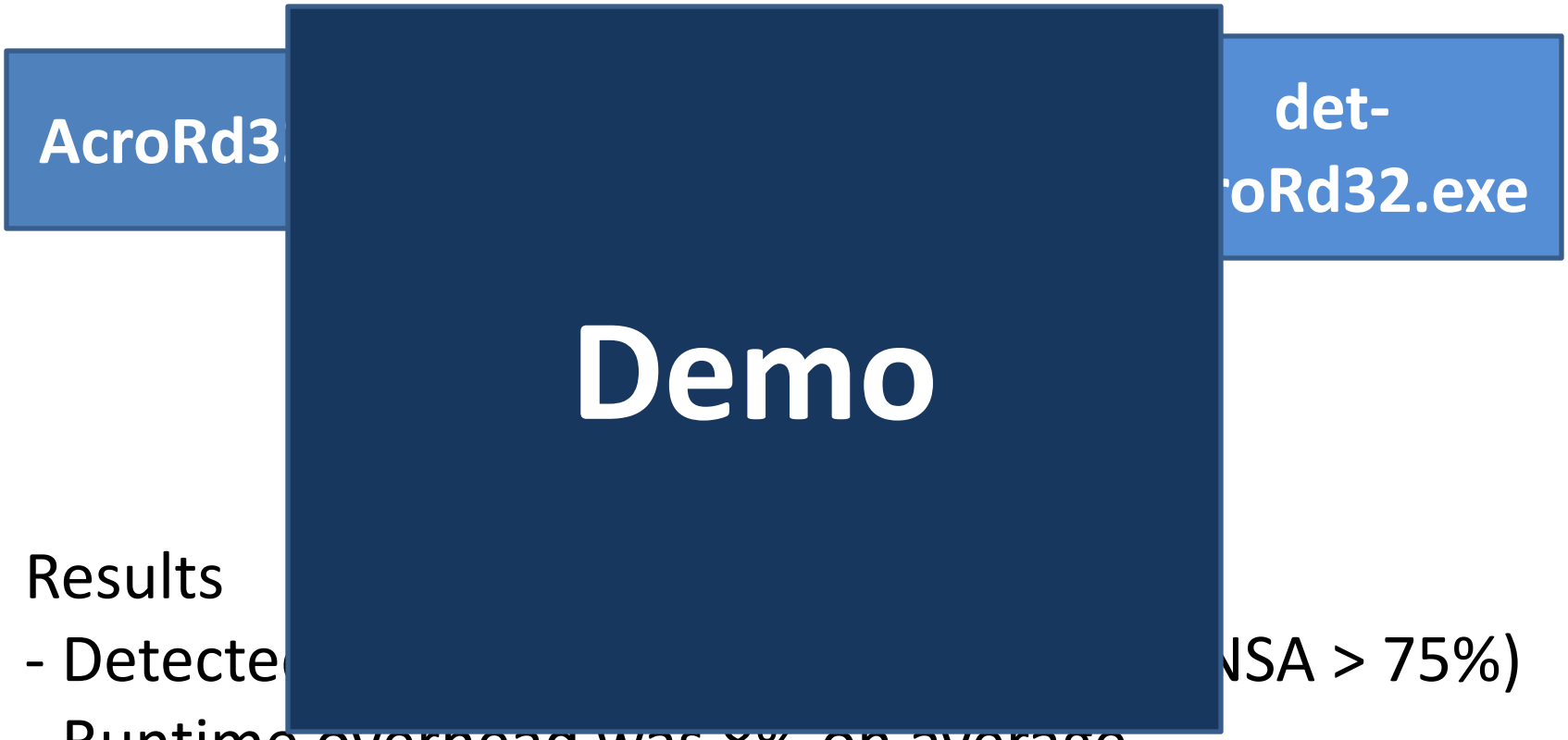
Result: max NSA between 76% and 96%

Nozzle detects real spraying attacks

Nozzle Runtime Overhead



Using Nozzle in Adobe Reader



Results

- Detecte NSA > 75%)
- Runtime overhead was 8% on average
- NSA of normal document < 10%

Summary

- Heap spraying attacks are
  - Easy to implement, easy to retarget
  - In widespread use
- Existing detection methods fail to classify malicious objects on x86 architecture
- Nozzle
  - Effectively detects published attacks (known and new)
  - Has acceptable runtime overhead
  - Can be used both online and offline

Questions?

Paruj Ratanaworabhan (paruj.r@gmail.com)

Ben Livshits (livshits@microsoft.com)

Ben Zorn (zorn@microsoft.com)

Nozzle heap spraying

See us on Channel 9:
http://channel9.msdn.com/posts/Peli/
Heap-Spraying-Attack-Detection-with-Nozzle/

Backup

Attacks on Nozzle

- Injecting junk into start of object
  - Where does the exploit code begin?
- TOCTTOU – When do you scan the object?
- Attacks on surface area calculation
  - Jumps outside of objects
  - Multiple instances of shellcode inside an object
- Hiding the code itself
  - Code that rewrites heap at last minute

What about Data Execution Prevention?

- DEP / NX bit = hardware to prevent code execution on the heap
- DEP is great , but isn't used everywhere
  - Issues with app compatibility
  - DEP can be circumvented
  - JIT compilers complicate the story
- Nozzle augments DEP for defense in depth

Normalized Surface Area Locally