RSAConference2015

San Francisco | April 20-24 | Moscone Center

CHANGE

Challenge today's security thinking

SESSION ID: EXP-R01

Patching Exploits with Duct Tape: Bypassing Mitigations and Backward Steps

James Lyne

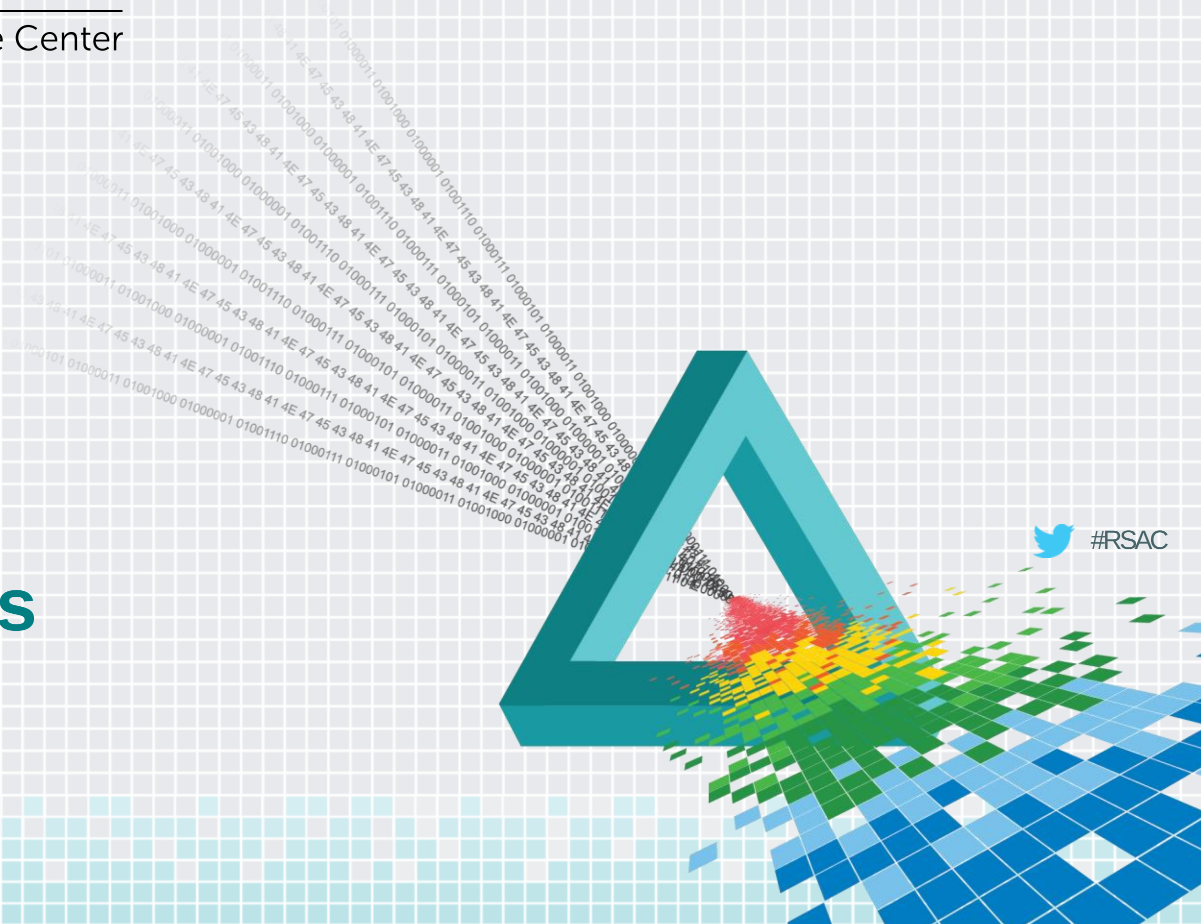
Global Head of Security Research
Sophos / SANS Institute
@jameslyne

Stephen Sims

Security Researcher
SANS Institute
@Steph3nSims

RSAConference2015

San Francisco | April 20-24 | Moscone Center



#RSAC

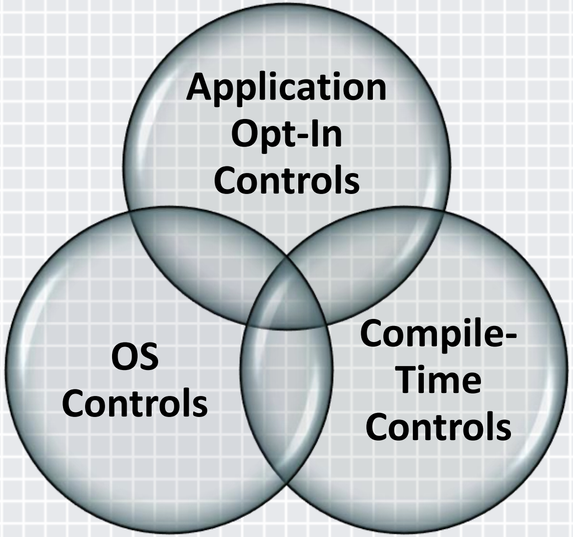
Exploit Mitigations

Purpose

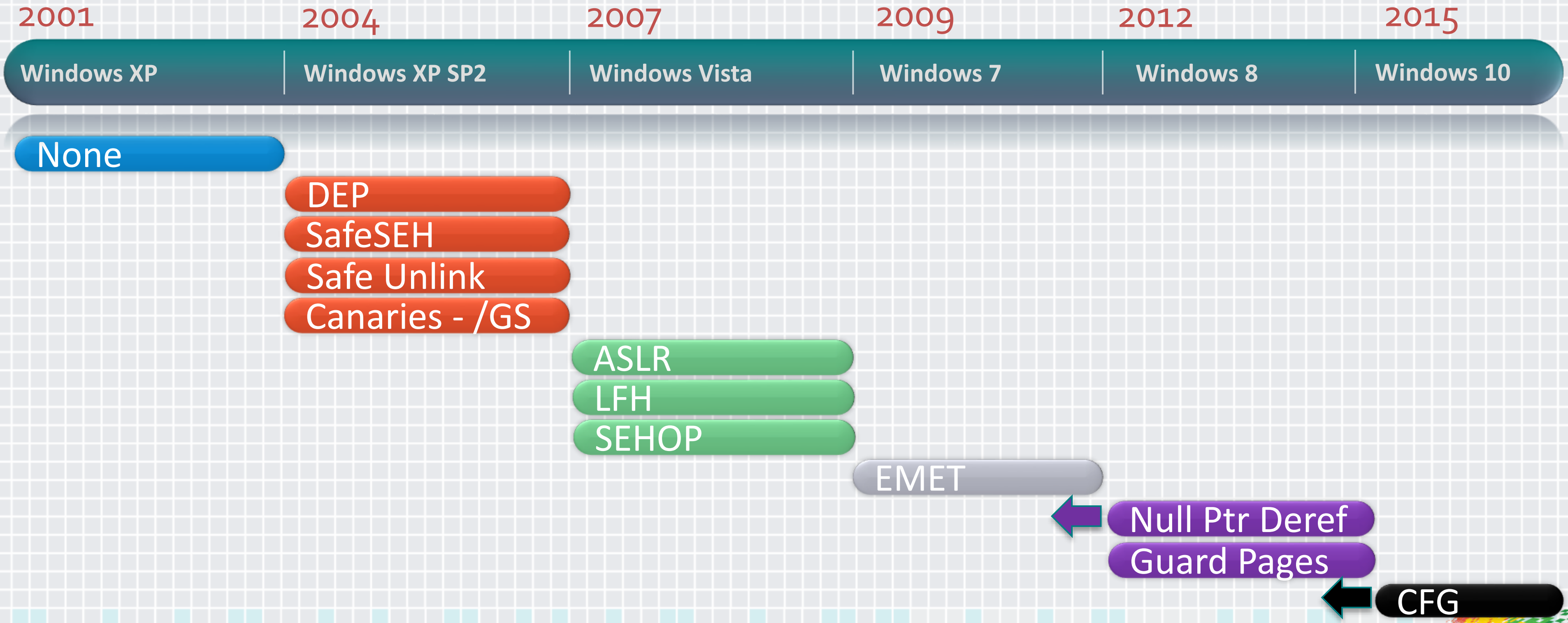
- 0-day exploit sales and bug bounties are very popular and profitable
  - In early 2014, Yang Yu earned $100K disclosing 3 exploit mitigation bypass techniques to MS
  - At CanSecWest Pwn2Own 2014 Vupen took home $400K
  - At CanSecWest Pwn2Own 2015 Jung Hoon Lee took home $225K
  - Google paid over $1.5M in 2014 in bug bounties
- Exploit writing is becoming *very* competitive
- We will focus on some of the mitigations and bypass techniques

Exploit Mitigation Controls

- Controls to mitigate the successful exploitation of a software vulnerability
- Three primary categories:
  - Compile-Time Controls – Canaries, SafeSEH
  - OS Controls – ASLR, DEP
  - Application Opt-In Controls – /dynamicbase, DEP
- Often have strict requirements to be effective
  - One bad module can break the whole protection
  - Better security when using multiple categories



High Level Timeline – Notable Client Mitigations



2001 Windows XP
2004 Windows XP SP2
2007 Windows Vista
2009 Windows 7
2012 Windows 8
2015 Windows 10

None

DEP
SafeSEH
Safe Unlink
Canaries - /GS

ASLR
LFH
SEHOP

EMET

Null Ptr Deref
Guard Pages

CFG

Exploit Mitigation Examples

- Data Execution Prevention (DEP)
- Address Space Layout Randomization (ASLR)
- Security Cookies / Canaries
- Safe-Unlink, Low Fragmentation Heap
- VTGuard, Sealed Optimization
- Ring3 and Ring0 Guard Pages
- Null Pointer Dereference Protection
- Range Checks
- SafeSEH, SEHOP

Sampling of Exploit Mitigation Bypass Techniques

- DEP – Return Oriented Programming (ROP), return-to-libc
- ASLR – Locate non-rebased modules, memory leaks and RVA offsets, brute force, memory spraying
- Security Cookies / Canaries – Canary repair, heap overflows, unprotected functions, SEH overwrites
- Safe-Unlink & LFH – Application data attacks
- SafeSEH – Locate non-protected modules, identify non-DLL executable memory regions
- SEHOP – Repair the SEH chain with local access and identification of required opcodes

Not as good as it seems?

Demo

Microsoft Enhanced Mitigation Experience Toolkit (EMET)

- Toolkit offering new and improved exploit mitigation controls
- EMET 5.2 officially released in March, 2015
- Must verify that applications are not negatively impacted due to controls
- Can help protect against 0-day attacks
- Heavily focused on ROP mitigation
- Newer control additions include EAF+, attack surface reduction, and Control Flow Guard (CFG)
- Very low adoption rate

EMET Demonstration

Demo

Isolated Heaps and New IE Protections

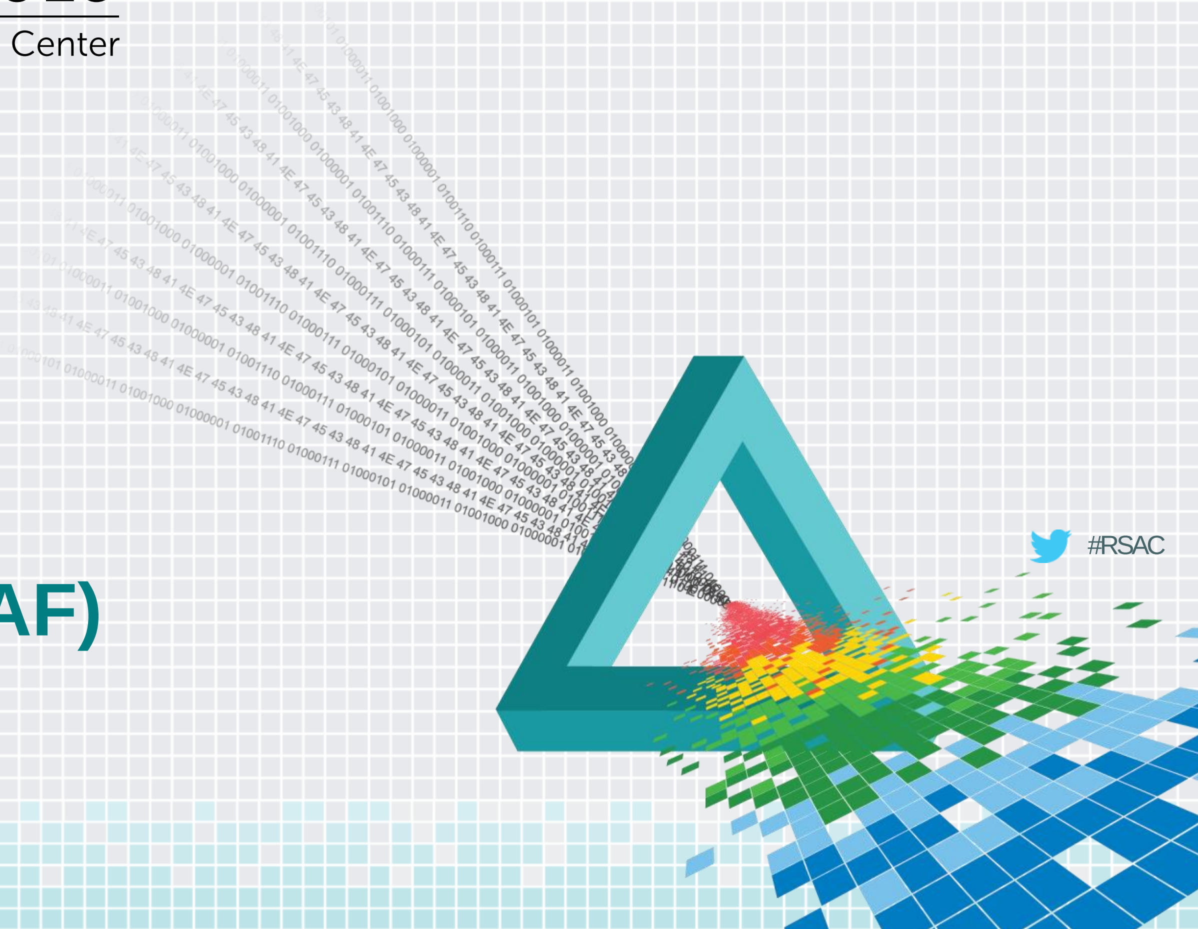
- Last year MS released patches for IE security
  - The June patch added Isolated Heaps for DOM objects to make the replacement of freed objects harder
  - The July patch added memory protection to help protect the freeing of objects with a delayed release
- The primary goal is to mitigate UAF exploitation
- Protected Free can be bypassed by meeting the release threshold
- Isolated Heaps can be bypassed by finding proper sized objects

Control Flow Guard (CFG)

- New control targeting ROP-based exploitation
- Compiler control supported by Windows 10 and Windows 8, update 3
- Creates a bitmap representing the start addressing of all functions
- If an indirect call (call EAX) is going to an address that is not the start of a valid function, the application terminates

Internet of Things (IoT)

- Typically, the more obscure an OS or device, the lower the number of exploit mitigation controls
- Lots of low-hanging fruit in home security devices, cars, power meters, electronic toll devices, wearable medical devices
- Pro tip: Don't fuzz the baby monitor



#RSAC

Use After Free (UAF)

What is Use After Free?

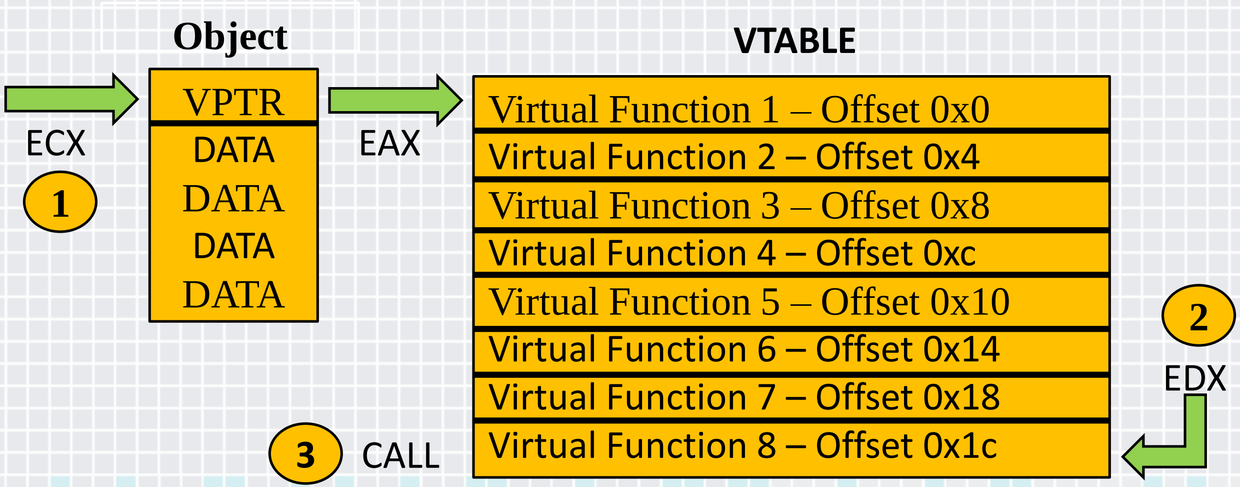
- A vulnerability class primarily affecting web browsers and large C++ applications
  - Typically detected when prematurely freed memory is later accessed by the application
  - Responsible for the bulk of Microsoft security bulletins
  - Extremely profitable, yielding $10K - $20K USD from ethical buyers and more from others
  - Difficult to detect through static analysis

Use After Free Basics

- When an object is created from a C++ class, and uses virtual functions:
  - A virtual function table (vftable) is created, holding pointers to relevant functions at static offsets
  - A virtual pointer (vptr) is allocated along with each instantiated object, pointing to the vftable
- When a virtual function is called:
  - The vptr is dereferenced into a register such as EAX
  - An offset from the [vptr] is dereferenced from the vftable
  - The virtual function is called

Normal Virtual Function Behavior

- 1) mov eax, [ecx] ← Deref the vptr from the object
- 2) mov edx, [eax+1ch] ← Deref the virtual function from vftable
- 3) call edx ← Call the virtual function



Object

VPTR

DATA

DATA

DATA

DATA

VTABLE

Virtual Function 1 – Offset 0x0

Virtual Function 2 – Offset 0x4

Virtual Function 3 – Offset 0x8

Virtual Function 4 – Offset 0xc

Virtual Function 5 – Offset 0x10

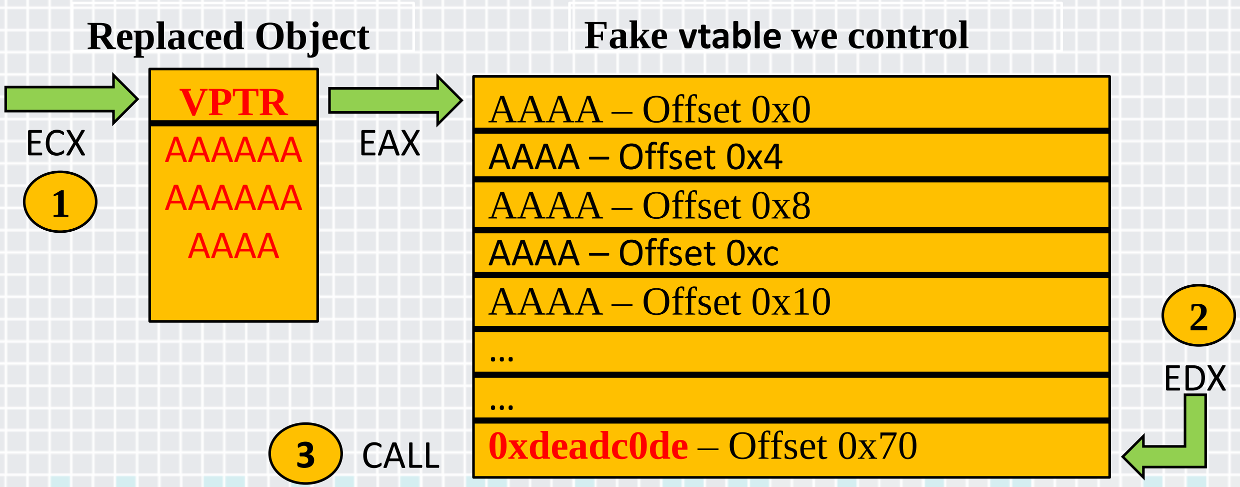
Virtual Function 6 – Offset 0x14

Virtual Function 7 – Offset 0x18

Virtual Function 8 – Offset 0x1c

UAF Exploit Behavior

- We replace the freed object with a malicious object
- If we can control the vptr and the data at that location, we can get control of the instruction pointer



Replaced Object

ECX

1

VPTR

AAAAAA

AAAAAA

AAAA

EAX

Fake vtable we control

AAAA – Offset 0x0

AAAA – Offset 0x4

AAAA – Offset 0x8

AAAA – Offset 0xc

AAAA – Offset 0x10

…

…

0xdeadc0de – Offset 0x70

2

EDX

3

CALL

Use After Free

- In other words…

Frodo the Hamster

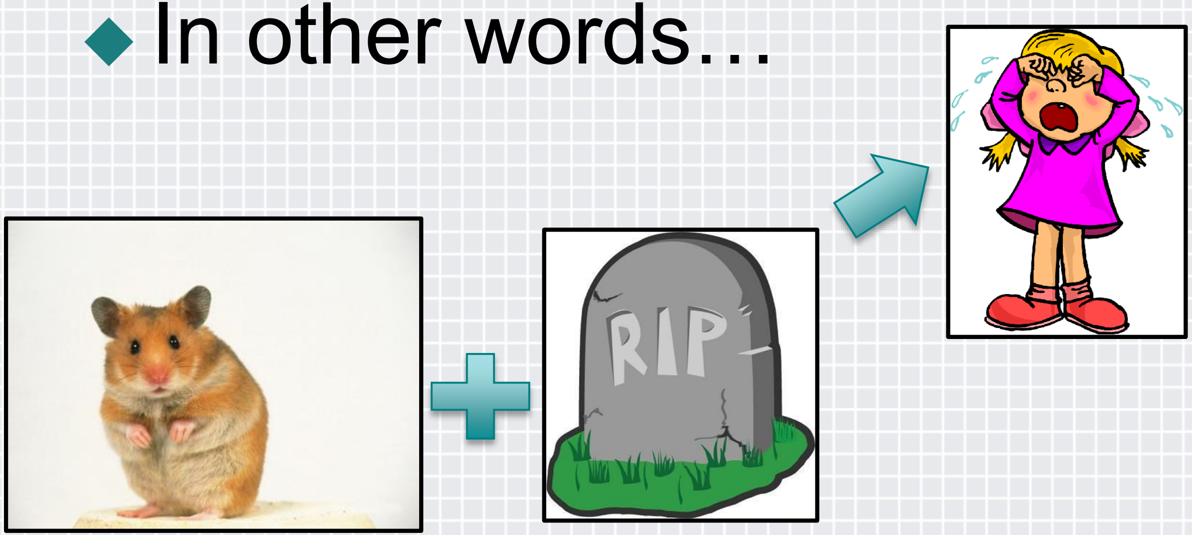
Use After Free

- In other words…

+

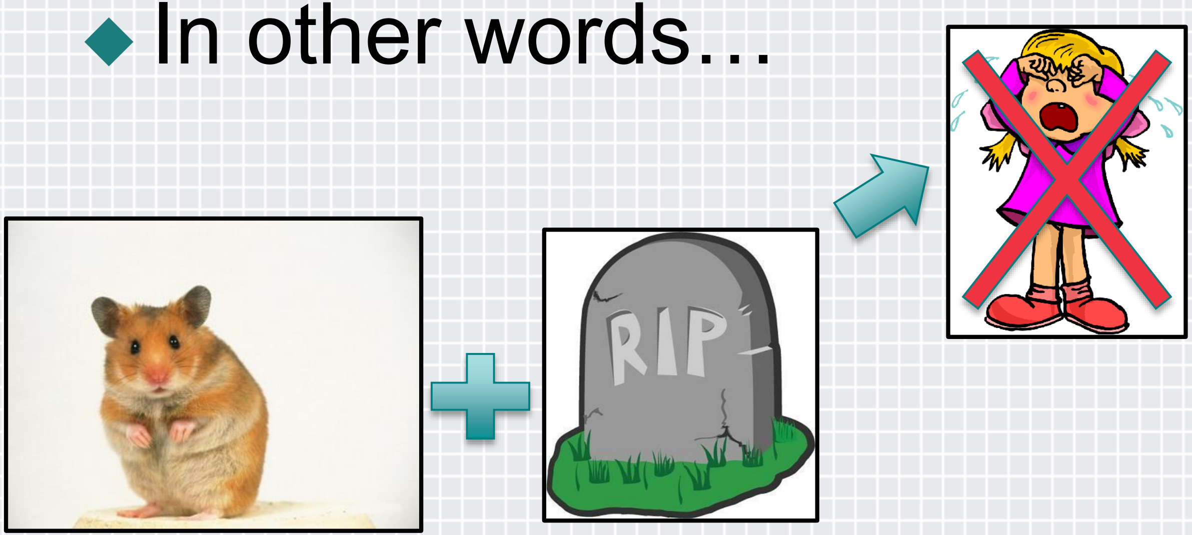
Use After Free

- In other words…



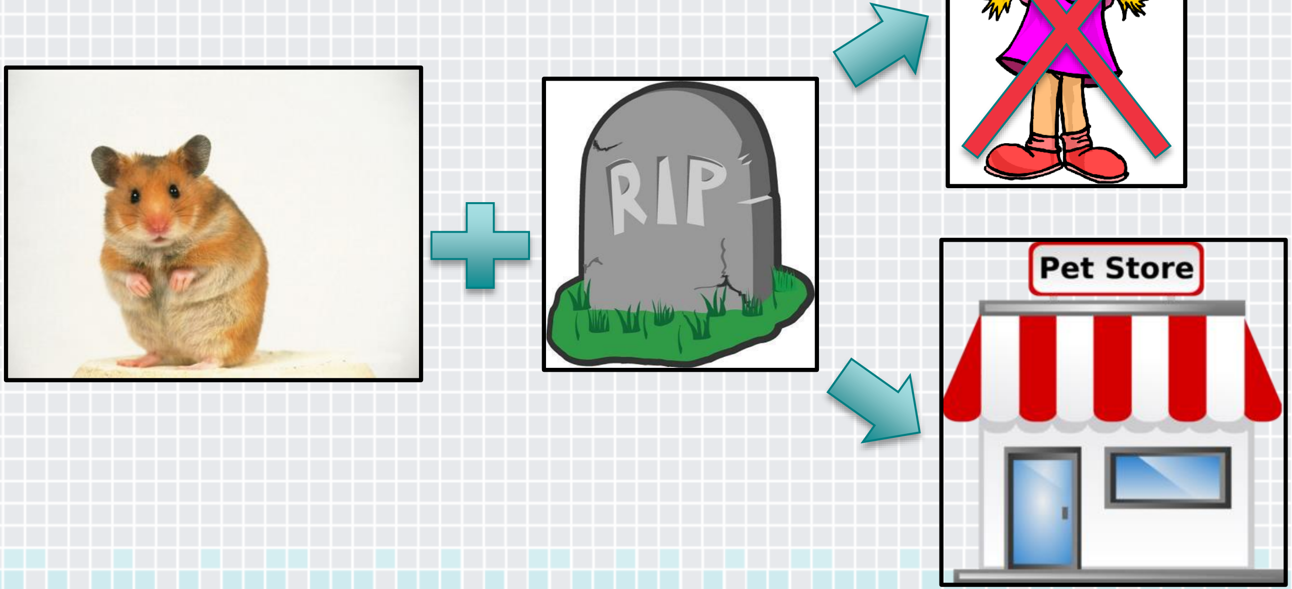
Use After Free

- In other words…



Use After Free

- In other words…



Use After Free

- In other words…

RIP
Pet Store

UAF Demo One – MS13-038

- On Tuesday, May 14th Microsoft issued the security bulletin for MS13-038
  - Critical Use After Free Vulnerability
  - http://technet.microsoft.com/en-us/security/bulletin/ms13-038
  - Allows for remote code execution on Windows XP through Windows 7 OS running IE8
- Publicly disclosed vulnerability discovered on April 30, 2013, found on the Department of Labor website, serving the exploit code to visitors
  - https://community.qualys.com/blogs/laws-of-vulnerabilities/2013/05/14/patch-tuesday-may-2013

UAF Demo Two – MS14-012

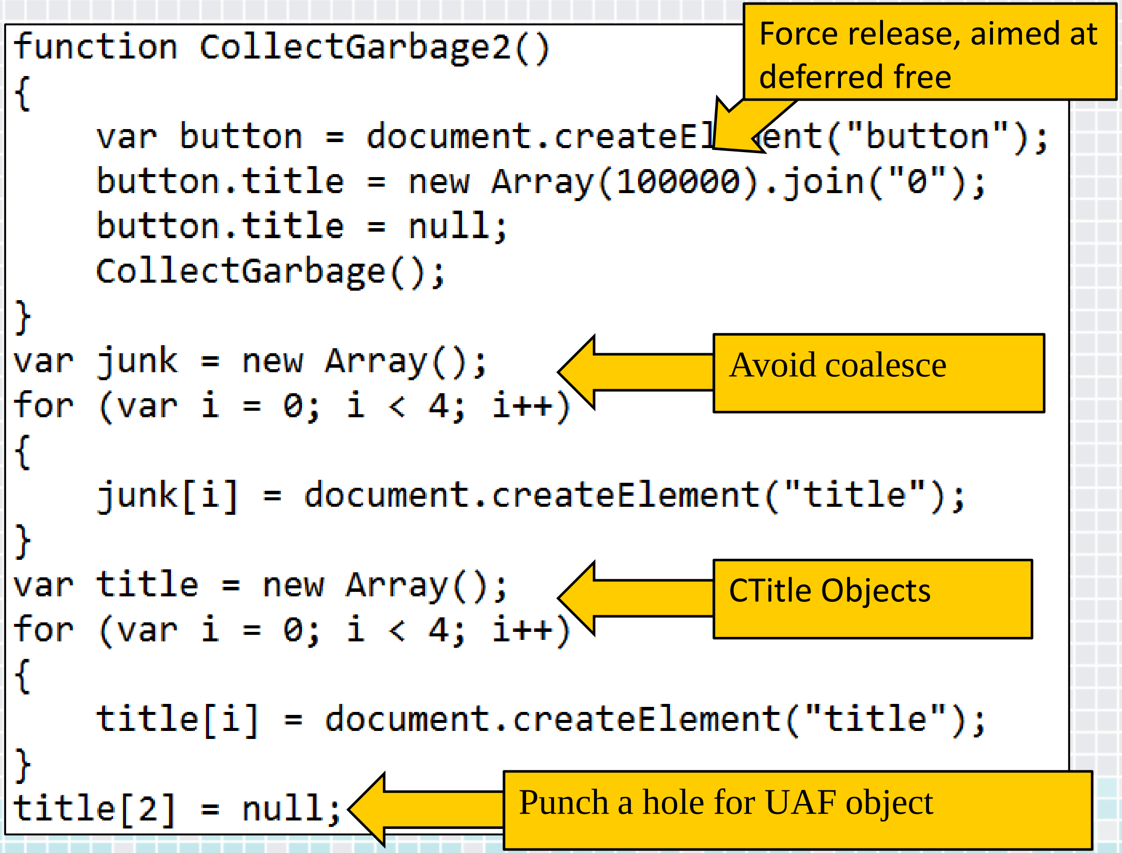
- UAF in MSHTML!Cmarkup
  - Crashes in UpdateMarkupContentsVersion
  - https://technet.microsoft.com/library/security/ms14-012
- Originally used in targeted attacks against military and industrial targets
- Original exploit checked for EMET
  - Does not bypass EMET, fails silently
  - Publicly available code does not check

Bypassing Isolated Heap

- Isolated Heap
  - k33nteam discovered a technique using heap coalescing to groom the heap and control the allocation space

http://k33nteam.org/blog-4-use-after-free-not-dead-in-internet-explorer-part-1.htm

Code (1)



```
function CollectGarbage2()
{
    var button = document.createElement("button");
    button.title = new Array(100000).join("0");
    button.title = null;
    CollectGarbage();
}
var junk = new Array();
for (var i = 0; i < 4; i++)
{
    junk[i] = document.createElement("title");
}
var title = new Array();
for (var i = 0; i < 4; i++)
{
    title[i] = document.createElement("title");
}
title[2] = null;
```

Force release, aimed at deferred free

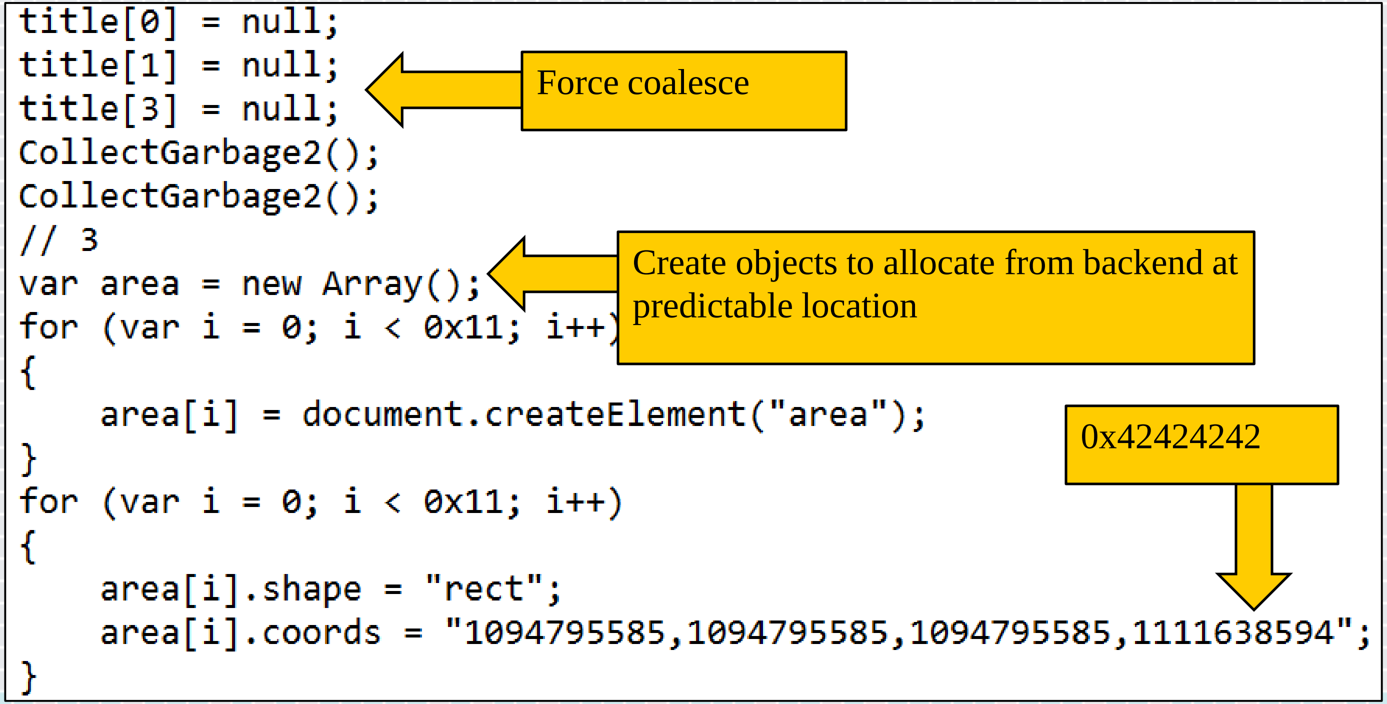
Avoid coalesce

CTitle Objects

Punch a hole for UAF object

Code (2)

Trigger UAF, then…



```
title[0] = null;
title[1] = null;
title[3] = null;
CollectGarbage2();
CollectGarbage2();
// 3
var area = new Array();
for (var i = 0; i < 0x11; i++)
{
    area[i] = document.createElement("area");
}
for (var i = 0; i < 0x11; i++)
{
    area[i].shape = "rect";
    area[i].coords = "1094795585,1094795585,1094795585,1111638594";
}
```

Force coalesce

Create objects to allocate from backend at predictable location

0x42424242

UAF Demo Three – MS14-056

- Bypassing Isolated Heap

Apply

- Consider using EMET for your environment
  - At least for high risk corporate applications
  - Profiles can be configured centrally and deployed
- Insure your development process uses latest mitigations (SDL)
- Microsoft has invested heavily in anti-exploit
  - What other platforms/devices are in your environment that haven't?
- Attackers have invested heavily in these skills – all security professionals need to develop an understanding, not just exploit writers.

Thanks!

Questions?

James Lyne - @jameslyne

Stephen Sims - @Steph3nSims