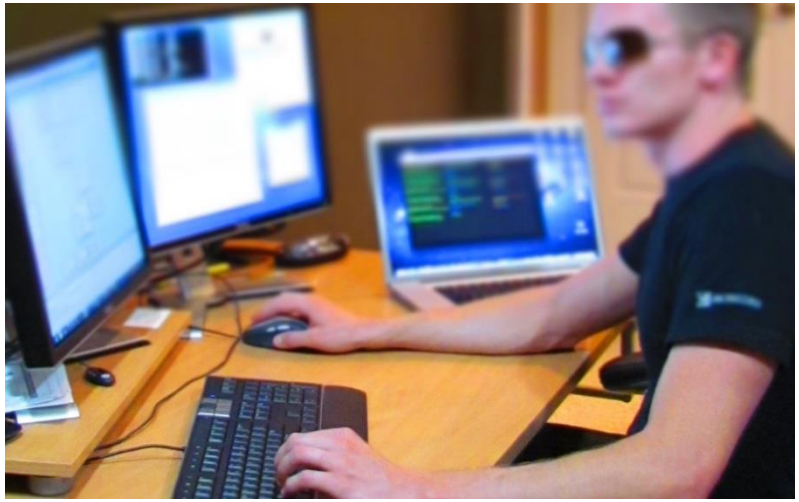
AppSec: Overview, Deep Dive, and Trends

Who am I?

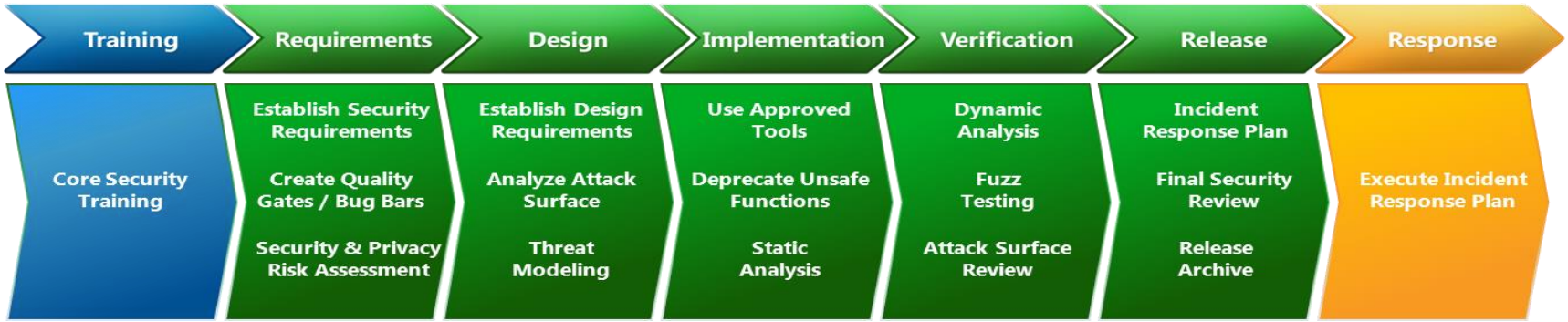
- Jared DeMott
  - Security Researcher
    - E.g. life long learner
      - Fuzzing
      - Code auditing
      - Reversing
      - Exploitation
  - Author
    - Fuzzing book and various articles
  - Speaker
    - Here, and lots of other venues
  - Trainer
    - Check out my full two day Class
      - Next at BlackHat USA
  - Friend
    - Drop me a line



Secure Development Lifecycle

New VS better than old

Fuzzing



Src code checking in build

Manual Review and Pentest

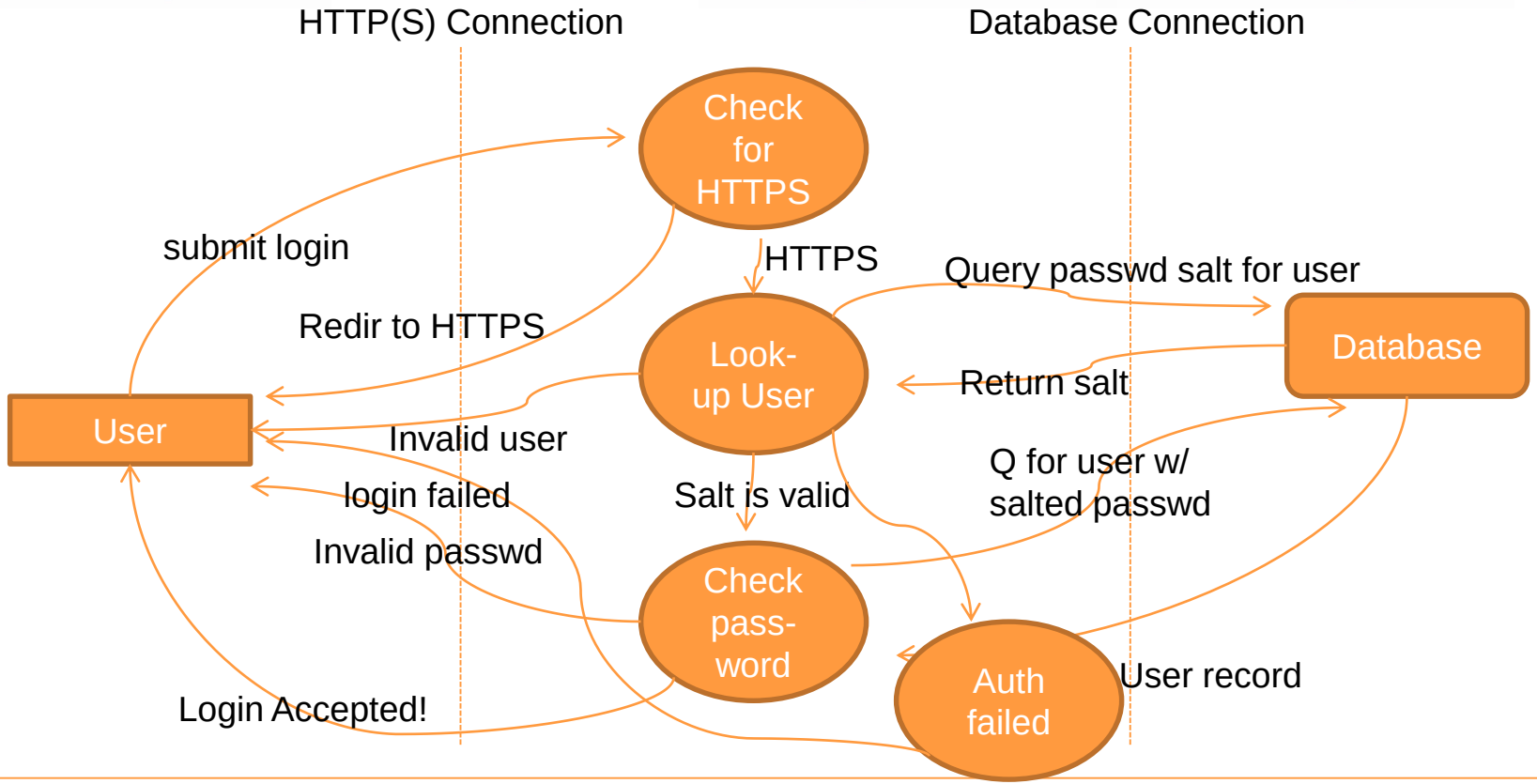
Push Security to the Left

- Before you code!
- Historically: an over focus on Testing
  - Under focus on Threat Modeling
  - Getting Devs, Testers, and Operational folks together
    - Especially for todays cloud applications

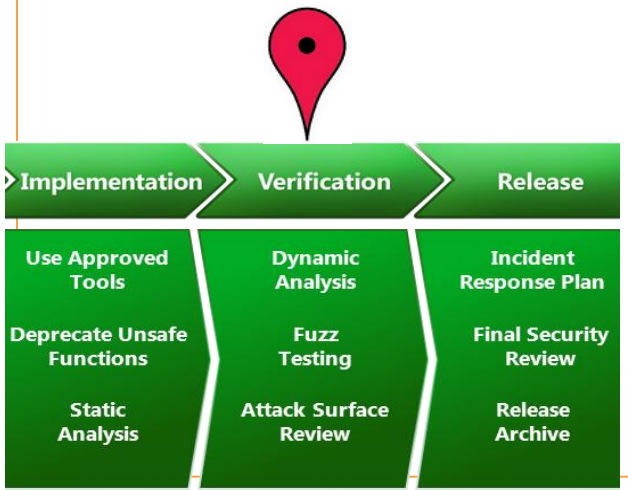
Threat Modeling Software

- Risk based threat models
  - Apps of LOW, MED, HIGH require different amounts of assurance
    - As an example, LOW apps might be the cafeteria menu.
      - The use of static analysis may be enough
    - MED applications, perhaps B2B web apps, require static and dynamic analysis
    - HIGH, consumer desktop products, might require all the prior, plus a more expensive pentest and manual analysis.
- Threat models help determine what we are testing for
  - Formal tools available but not widely used
    - http://www.microsoft.com/security/sdl/adopt/threatmodeling.aspx

Design Review via DFD

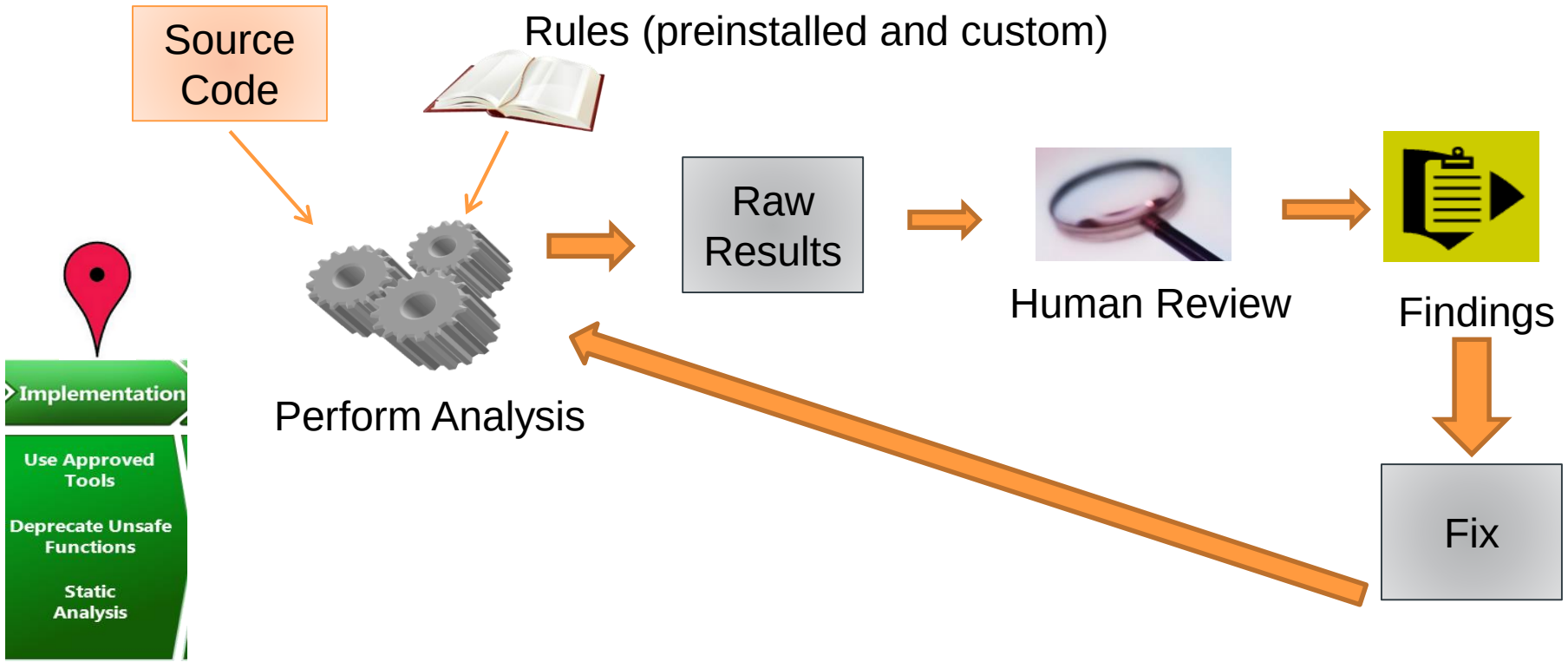


- Do all three if budget allows and threat dictates
  - Static
  - Dynamic
  - Manual



	Static	Dynamic	Manual
Scans all code for known buggy patterns	X		
Hammers attack surface using heuristics to find bugs		X	
Finds tricky design flaws and implementation bugs			X
Lower cost	X	X	
Med cost	X	X	
Higher cost			X
Miss Bugs	Yes	Yes	Yes
False Positive	Yes	Not usually	Maybe

Functional View of Static Analysis

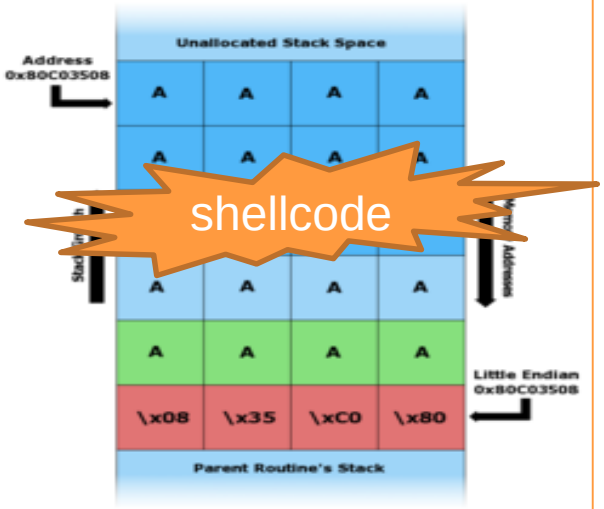


Quickly Finds Bugs for which a Known Pattern Exists

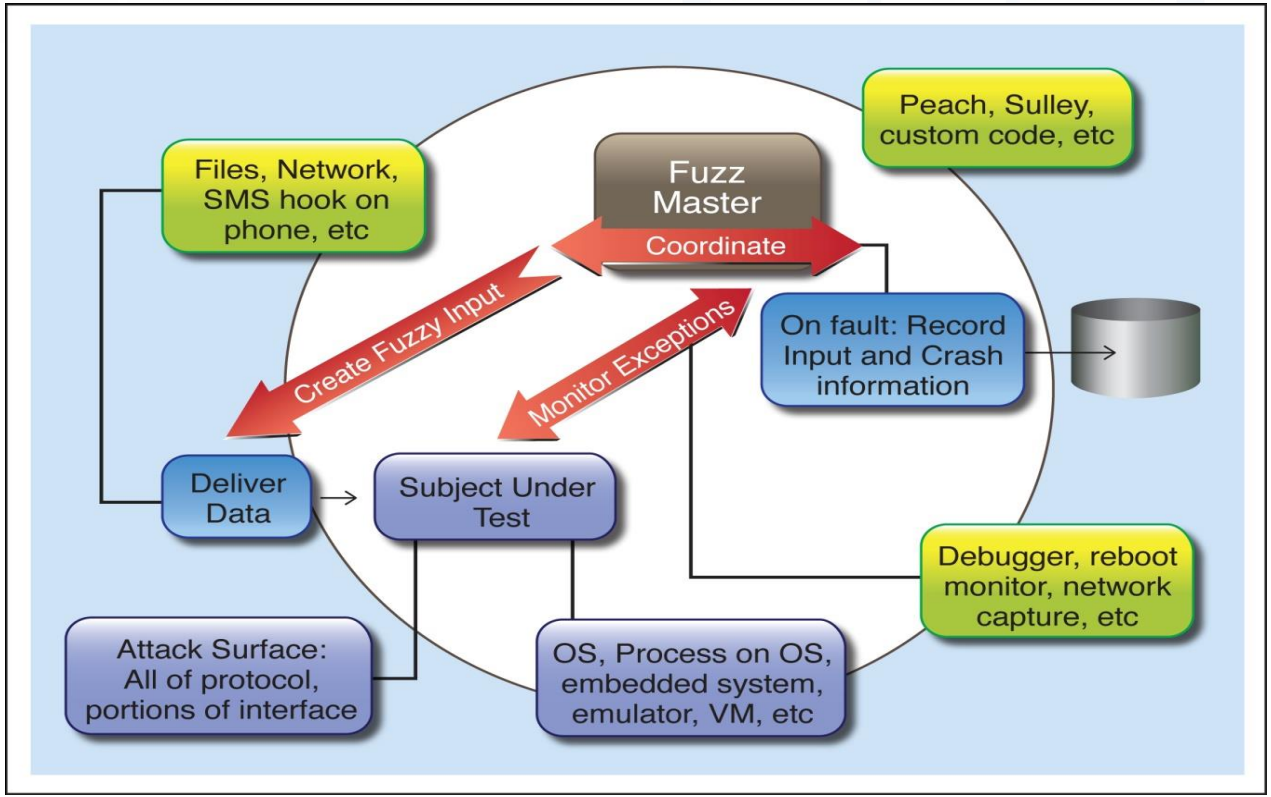
- Buffer Overflow
  - Untrusted data written outside of some data structure
  - Allowing the attacker to hijack code execution

```
--------------------------------
        char buf[1024];
        sprintf(buf, “%s@%s”, name, domain);
--------------------------------
        char buf[100];
        for(int i=0; i<=100; i++)
                buf[i]=i;
--------------------------------
        char * buf = malloc(100);
        strncpy(buf, argv[1], strlen(argv[1]));
--------------------------------
        printf(argv[1]);
```

Banned.h



Fuzzing



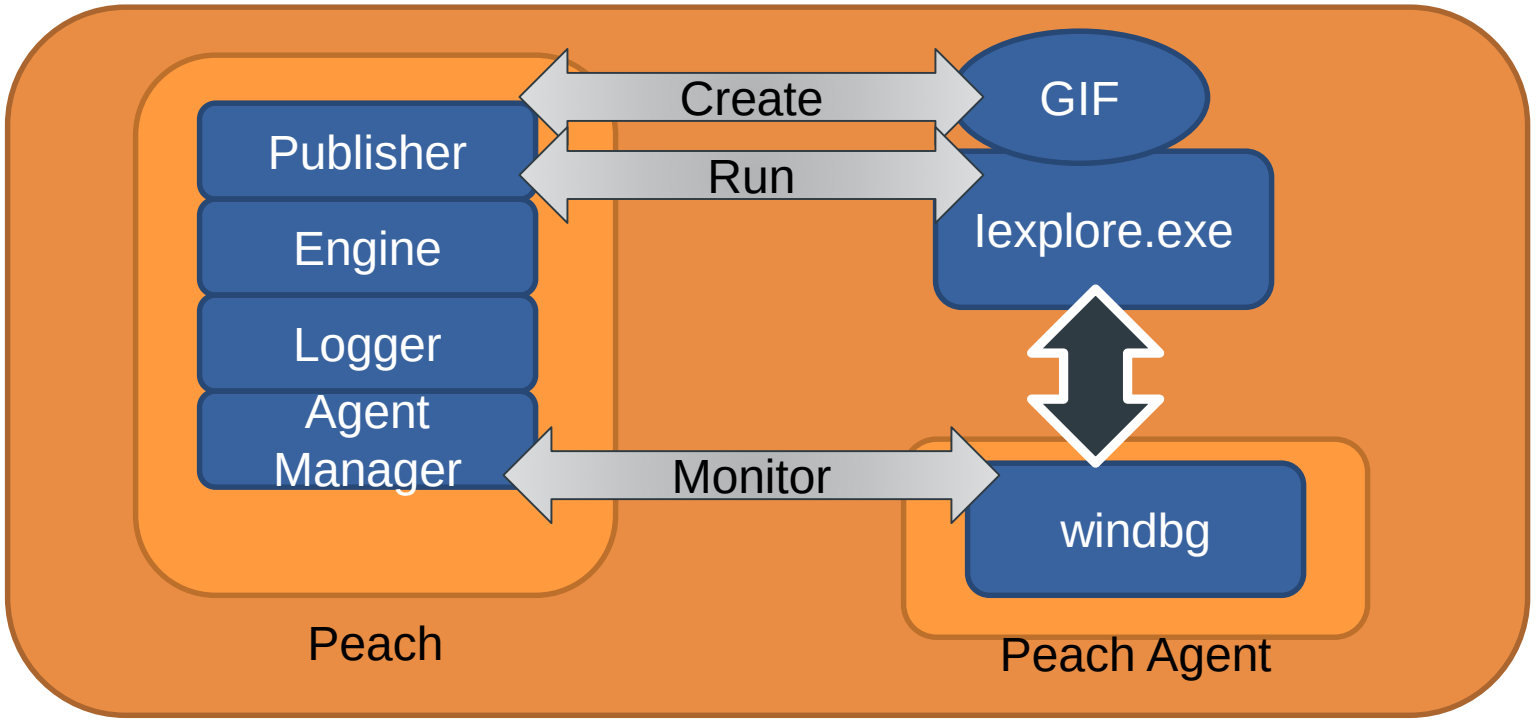
Verification

Dynamic Analysis

Fuzz Testing

Attack Surface Review

File Fuzzing Demo with Peach



Manual Code Review

- Look for hard to spot implementation bugs and architectural flaws

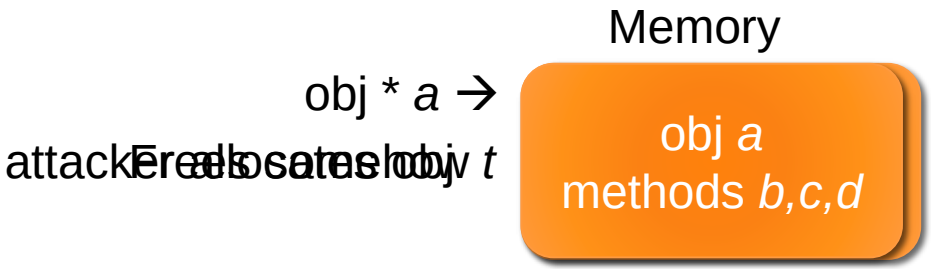
- Null ptrs
- Typos on variables
- Forgotten default switch case
- Uninitialized memory
- Incorrect pointer usage
- Returning locally scoped variable
- Exception handling mistakes
  - Out of state alloc/free

- TOCTOU race conditions
  - Applies to files, shared memory, etc.
  - Concurrency issues
- Unchecked return values
- Out of date compilers
- Ignored warnings
- Failure to opt into protections
- Old STL

Example 1: Use-after-Free (UAF)

- Common in browsers
  - Because JavaScript events can delete an object at unexpected times, while back in the C++ of the browser – the object is about to get used again
    - And this bug can occur in other types of applications as well of course
  - Chrome
    - Probably has the best sandbox, but look out for kernel exploits, and sandbox escapes
    - The usual bugs as well, but less of them
  - Safari
    - Webkit…Google just forked to their blink… not thinking that will help Apples security posture
  - Internet Explorer
    - Plenty of UAF examples in metasploit
  - Firefox
    - Bugzilla is helpful for finding new bugs to explore
  - Opera
    - Security through obscurity? Seriously, don't use it
      - RWX in mem, bugs galore, etc. bad news

Use-after-Free



1. *a→b*() is called by application(e.g. original obj used after freed)
2. But expected virtual pointer is not present
3. Instead program dereferences attacker controlled data (func ptr)
4. Which may allow any of the three primitives: R/W/X

Use-After-Free Remote Code Executions

- Examples:
  - Chrome CVE-2013-2871
  - Firefox CVE-2013-1704
  - Internet Explorer CVE-2013-1311
  - Safari CVE-2011-3443
  - Opera SVG CVE-2013-1638

Webkit UAF: Prior Chrome Bug

setOuterText in HTMLElement.cpp

```
// FIXME: This creates a single text node even when the text has CR and LF
// characters in it. Instead it should create <br> elements.
RefPtr<Text> t = Text::create(document(), text);
ec = 0;
parent->replaceChild(t, this, ec);
if (ec)
    return;

// Is previous node a text node? If so, merge into it.
Node* prev = t->previousSibling();
if (prev && prev->isTextNode()) {
    Text* textPrev = static_cast<Text*>(prev);
    textPrev->appendData(t->data(), ec);
    if (ec)
        return;
    t->remove(ec);
    if (ec)
        return;
    t = textPrev;
}

// Is next node a text node? If so, merge it in.
Node* next = t->nextSibling();
if (next && next->isTextNode()) {
    Text* textNext = static_cast<Text*>(next);
    t->appendData(textNext->data(), ec);  //can remove what textNext points at, since not ref pointers.  look for raw ptrs as pattern
    if (ec)
        return;
    textNext->remove(ec);
    if (ec)
        return;
}
```

Non-ref ptr defined

Uh oh. Possible UAF

UAF: Example -- Fixed

```
static void mergeWithNextTextNode(PassRefPtr<Node> node, ExceptionCode& ec)
{
    ASSERT(node && node->isTextNode());
    Node* next = node->nextSibling();
    if (!next || !next->isTextNode())
        return;

    RefPtr<Text> textNode = static_cast<Text*>(node.get());
    RefPtr<Text> textNext = static_cast<Text*>(next);
    textNode->appendData(textNext->data(), ec);
    if (ec)
        return;
    if (textNext->parentNode()) // Might have been removed by mutation event.
        textNext->remove(ec);
}

void HTMLElement::setOuterHTML(const String& html, ExceptionCode& ec)
{
    Node* p = parentNode();
    if (!p || !p->isHTMLElement()) {
        ec = NO_MODIFICATION_ALLOWED_ERR;
        return;
    }
    RefPtr<HTMLElement> parent = toHTMLElement(p);
    RefPtr<Node> prev = previousSibling();
    RefPtr<Node> next = nextSibling();

    RefPtr<DocumentFragment> fragment = createFragmentFromSource(html, parent.get(), ec);
    if (ec)
        return;

    parent->replaceChild(fragment.release(), this, ec);
    RefPtr<Node> node = next ? next->previousSibling() : 0;
    if (!ec && node && node->isTextNode())
        mergeWithNextTextNode(node.release(), ec);

    if (!ec && prev && prev->isTextNode())
        mergeWithNextTextNode(prev.release(), ec);
}
```

Now uses reference pointer

Example 2: Double Fetch

Time-of-check to time-of-use race condition

An exemplary bug in a syscall handler

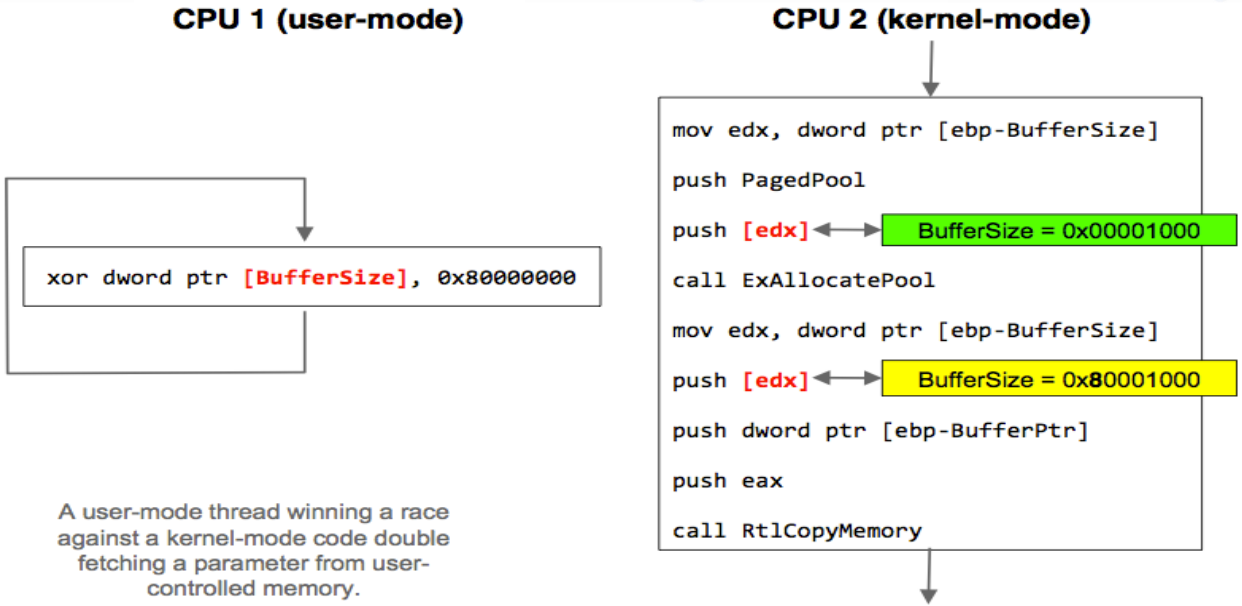
```
PDWORD BufferSize = /* controlled user-mode address */;
PBYTE BufferPtr = /* controlled user-mode address */;
PBYTE LocalBuffer;

LocalBuffer = ExAllocatePool(PagedPool, *BufferSize);
if (LocalBuffer != NULL) {
  RtlCopyMemory(LocalBuffer, BufferPtr, *BufferSize);
} else {
  // bail out
}
```

Once OK

Again bad!

Double Fetch



A user-mode thread winning a race against a kernel-mode code double fetching a parameter from user-controlled memory.

http://vexillium.org/dl.php?syscan_slides.pdf

Another Double Fetch, with Fix

Fetch twice… bad

```
__try {
  ProbeForWrite(*UserPtr, sizeof(STRUCTURE), 1);
  (*UserPtr)->Field = 0;
} except {
  return GetExceptionCode();
}
```

vs.

Fetch Once Good

```
PSTRUCTURE Pointer;
__try {
  Pointer = *UserPtr;

  ProbeForWrite(Pointer, sizeof(STRUCTURE), 1);
  Pointer->Field = 0;
} except {
  return GetExceptionCode();
}
```

Patch

- Do you have a secure@company.com?
- Who will respond to it?
- How quickly do you commit to fixing bugs for customers?
  - Likely depends on realities
    - Severity of bug
    - Ease of repair
    - etc

Trends

- SDL has “caught on”
  - At least in bigger organizations
  - Thus, well made software has less “lame bugs”
- But…. software is still getting more complex
  - Newer types of interesting bugs being found
  - 3rd party libraries
    - If you were going to try and pwn Safari
      - audit closed source html parser?
        - No.
      - Or grep open source webkit for “FIXME”?
        - YES!
- Better analysis on why bugs were missed
  - Lot of discussion around why tools/techniques missed heartbleed

Q&A will happen at the very end

http://labs.bromium.com

@bromium; @jareddemott