DEFCON Quals

• **May 15/16/17**
  – Starts 8pm Friday, May 15th
  – Sage 3101 Friday, Sage 4101 Saturday/Sunday

• **Extra Credit**
  – Letter grade raise on a Lab
  – **OR** +10% on the final project

• **To get the extra credit**
  – Solve **one** challenge (**that’s not a sanity check**)
  – **OR** Play 10 hours on Saturday and/or Sunday
Lecture Overview

- **Security**
  - Security Today
  - Security Tomorrow

- **Exploitation**
  - Exploitation Today
  - Exploitation Tomorrow
CVE Statistics – May 2015

http://www.cvedetails.com/browse-by-date.php
Security Trends

• As you know, security and mitigation technologies are no doubt getting better

– Why the spike in 2014?
CVE Statistics – May 2015

Vulnerabilities By Year

http://www.cvedetails.com/browse-by-date.php
As you know, security and mitigation technologies are no doubt getting better.

- Why the spike in 2014?

Possibly a result of the Snowden revelations.

- The fallout raised global awareness and interest in security/privacy. ‘Cyber’ in the news ever since.
Unsustainable Complexity

- Exploits are getting more and more complex
  - More bugs
  - More time
  - More money
Unsustainable Complexity

Exploit Complexity

$ 2015

Years

$ 20??

Exploit Complexity

$
Unsustainable Complexity

- Exploits are getting more and more complex
  - More bugs
  - More time
  - More money

- At what point do hobbyists have to draw the line? Companies? Contractors? Nation States?
Unsustainable Complexity

Exploit Complexity

Nation states

Sec firms

The hobbyist

$ 2015 Years 20??
The Security Mindset

- Systems and applications will never be perfectly secure. Period.
The Security Mindset

- Systems and applications will never be perfectly secure. Period.

- They just have to be hard enough to break that nobody can afford it anymore.
The Weakest Link - Humans

https://xkcd.com/538/
Lecture Overview

- Security
  - Security Today
  - Security Tomorrow

- Exploitation
  - Exploitation Today
  - Exploitation Tomorrow

MBE - 05/12/2015
Future of Security & Exploitation
The Future of Security

• The entry bar for **binary exploitation** is rising faster and faster
  – It’s starting to outpace individuals and hobbyists interest, ability, and dedication to enter the field
Unsustainable Complexity

Exploit Complexity

$\ldots$

Years

$\ldots$

2015

2020?

20??

the hobbyist
The Future of Security

- Memory corruption based exploits will no longer be feasible to produce for the average desktop or server.
The Future of Security

• Memory corruption based exploits will no longer be feasible to produce for the average desktop or server
  – In the immediate 10-20 years (?)
• Embedded devices are further behind
The Future of Security

• Implementation & logic flaws will probably always exist
  – You can’t really fix stupid
The Future of Security

• Implementation & logic flaws will probably always exist
  – You can’t really fix stupid

• What we will see and discover more of:
  – Sponsored backdoors, ‘cheating’
  – Hardware backdoors, flaws, supply chain trust
  – Crypto backdoors, subtle design flaws
Lecture Overview

- Security
  - Security Today
  - Security Tomorrow

- Exploitation
  - Exploitation Today
  - Exploitation Tomorrow
This Course

• You spent **hours** looking for bugs

• You spent **hours** reversing in IDA

• You spent **hours** debugging with GDB

• You spent **hours** writing python
This Course

• You spent hours looking for bugs

• You spent **hours** reversing in IDA

• You spent **hours** debugging with GDB

• You spent **hours** writing python
Bug Hunting

- Looking for bugs with or without source is the most time consuming part of the process.
Bug Hunting

- **Looking for bugs** with or without source is the most *time consuming* part of the process

- **How can we find these bugs faster?**

```assembly
push edi
call sub_319623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
call sub_3146A
test eax, eax
jz short loc_31306D
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push esi
push [ebp+arg_4]
push edi
call sub_319623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066:
; CODE XREF: sub_312FD0
; Sub_312FD0+51
push 0Dh
call sub_31411B

loc_31306D:
; CODE XREF: sub_312FD0
; Sub_312FD0+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

; ----------------------------------------

loc_31307D:
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h

loc_31308C:
mov [ebp+var_4], eax
```

MBE - 05/12/2015
Bug Hunting

• Looking for bugs with or without source is the most time consuming part of the process

• How can we find these bugs faster?
  – Automation
Static Code Analyzers

• Source code analyzers can help find bugs statically, but they can also miss a lot—Very hard to detect many real UAF’s statically
Static Code Analyzers

• Source code analyzers can help find bugs statically, but they can also miss a lot
  – Very hard to detect many real UAF’s statically

• Coverity is popular with the kids nowadays
  – integrates straight with GitHub
Static Code Analyzers

• Source code analyzers can help find bugs statically, but they can also miss a lot
  – Very hard to detect many real UAF’s statically

• Coverity is popular with the kids nowadays
  – integrates straight with GitHub

• Tons of good options for C/C++ Code
  – http://spinroot.com/static/
Fuzzing

- Fuzzing – The act of mangling data and throwing it at a target application to see if it mishandles it in some fashion

- Fuzzing has probably been the source of over 95% of the bugs from the past 10 years – The fuzzing era is starting to wind down
Fuzzing

• Remember these labs?
  – 7C
  – 7A
  – 9C
  – 9A
  – ...

• Since the scope of the labs is so small, it would have been easy to fuzz them
American Fuzzy Lop (AFL)

- A ‘security-oriented’ fuzzer that inserts and utilizes instrumentation that it inserts at compile time
  - Requires source code to be super effective
# American Fuzzy Lop (AFL)

## American Fuzzy Lop 1.74b (readelf)

<table>
<thead>
<tr>
<th>Process Timing</th>
<th>Overall Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>run time</strong></td>
<td>cycles done: 0</td>
</tr>
<tr>
<td><strong>last new path</strong></td>
<td>total paths: 812</td>
</tr>
<tr>
<td><strong>last uniq crash</strong></td>
<td>uniq crashes: 8</td>
</tr>
<tr>
<td><strong>last uniq hang</strong></td>
<td>uniq hangs: 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle Progress</th>
<th>Map Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>now processing</strong></td>
<td>map density: 3158 (4.82%)</td>
</tr>
<tr>
<td><strong>paths timed out</strong></td>
<td>count density: 2.56 bits/tuple</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage Progress</th>
<th>Findings in Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>now trying</strong></td>
<td>favored paths: 1 (0.12%)</td>
</tr>
<tr>
<td><strong>stage execs</strong></td>
<td>new edges on: 318 (39.16%)</td>
</tr>
<tr>
<td><strong>total execs</strong></td>
<td>total crashes: 63 (8 unique)</td>
</tr>
<tr>
<td><strong>exec speed</strong></td>
<td>total hangs: 191 (10 unique)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fuzzing Strategy Yields</th>
<th>Path Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bit flips</strong></td>
<td>levels: 2</td>
</tr>
<tr>
<td><strong>byte flips</strong></td>
<td>pending: 812</td>
</tr>
<tr>
<td><strong>arithmetic</strong></td>
<td>pend fav: 1</td>
</tr>
<tr>
<td><strong>known ints</strong></td>
<td>own finds: 811</td>
</tr>
<tr>
<td><strong>dictionary</strong></td>
<td>imported: n/a</td>
</tr>
<tr>
<td><strong>havoc</strong></td>
<td>variable: 0</td>
</tr>
<tr>
<td><strong>trim</strong></td>
<td>[cpu: 15%]</td>
</tr>
</tbody>
</table>

MBE - 05/12/2015

Future of Security & Exploitation
American Fuzzy Lop (AFL)

- A ‘security-oriented’ fuzzer that inserts and utilizes instrumentation that it inserts at compile time
  - Requires target source code to be super effective

- Great for file format fuzzing!
  - Generally not that useful for CTF fuzzing :/

- http://lcamtuf.coredump.cx/afl/
As the bugs get more refined and complex, fuzzing will only take us so far.
Fundamentals of Modern Bugs

• As the bugs get more refined and complex, fuzzing will only take us so far.

• Many modern bugs have to be ‘forced’ by requiring very specific conditions — like some sort of crazy edge cases.
A ‘timeless debugger’ – By GeoHot

- Observe a binary at any point of its execution state for a given input
- You can move forwards and backwards in time
QIRA

• A ‘timeless debugger’ – By GeoHot
  – Observe a binary at any point of its execution state for a given input
  – You can move forwards and backwards in time

• Super basic taint sort of functionality
  – Helps visualize r/w of specific memory addresses

• http://qira.me/
PANDA

• An ‘open-source Platform for Architecture-Neutral Dynamic Analysis’ – By MITLL
MBE - 05/12/2015

Future of Security & Exploitation

PANDA

Brendan Dolan-Gavitt
Columbia University

Josh Hodosh, Patrick Hufn, Tim Leek, and
Ryan Whelan
MIT Lincoln Lab

loc_31307D:
call sub_3140F3
and eax, 0FFFFh
or eax, 80000000h

loc_31308C:
mov [ebp+var_4], eax

; CODE XREF: sub_312FD8
; sub_312FD8+49

; CODE XREF: sub_312FD8
; sub_312FD8+51

; CODE XREF: sub_312FD8
; sub_312FD8+42
PANDA

• An ‘open-source Platform for Architecture-Neutral Dynamic Analysis’ – By MITLL

• Built on top of QEMU, allows instrumentation, analysis, and replay of an entire system
PANDA Model

Record Whole System Execution → Write Analysis Plugins → Run Replay and Analyze → RE Understanding

MBE - 05/12/2015  Future of Security & Exploitation
PANDA

• An ‘open-source Platform for Architecture-Neutral Dynamic Analysis’ – By MITLL

• Built on top of QEMU, allows instrumentation, analysis, and replay of an entire system

• Awesome plugin infrastructure
  – Utilizes LLVM Intermediate Representation to make one size fits all (CPU’s) analysis plugins

• https://github.com/moyix/panda
Advanced Concepts Today

• **Taint Analysis**
  - Tracing the impact of user input throughout the binary, and how it influences execution
  - PANDA, QIRA

• **Symbolic Execution + SAT/SMT Solving**
  - Proving that specific conditions can exist in execution to manifest difficult bugs
  - Z3, SMT-LIB

• **Machine Learning**
Lecture Overview

• Security
  – Security Today
  – Security Tomorrow

• Exploitation
  – Exploitation Today
  – Exploitation Tomorrow
DARPA’s Cyber Grand Challenge
DARPA’s Cyber Grand Challenge

https://www.youtube.com/watch?v=OVV_k73z3E0
About CGC

- A challenge set forth by DARPA
- Can we develop a **completely autonomous** system that is capable of...
  - finding vulnerabilities *(whitebox and blackbox)*
  - patching said vulnerabilities
  - writing exploits for said vulnerabilities
Some CGC Competitors

TRAIL of BITS

Raytheon

SRI International

Lincoln Laboratory
Massachusetts Institute of Technology

Secure

Berkeley
University of California
The ‘Cyber Reasoning Systems’ being developed by CGC competitors are quickly pushing the envelope of bug discovery and exploitation.
Exploitation of Tomorrow

• The ‘Cyber Reasoning Systems’ being developed by CGC competitors are quickly pushing the envelope of bug discovery and exploitation

• The technology behind them is likely to be some smart fuzzers guided by taint analysis, constraint solvers, and more