YAHOO!

Offense At Scale

PRESENTED BY Chris Rohlf | May 30th, 2015

whoami

- I lead the pentesting team at Yahoo!
- Founded Leaf Security Research, a boutique security consultancy
- Previously {Principal Consultant Matasano Security, DoD}
- Black Hat review board member and speaker (2009, 2011, 2013)
- Security researcher and tool/exploit developer

Introduction

This keynote

Introduction

What is a system in 2015?

Introduction

What is scale?

Disposable Environments

"Program against your datacenter like it's a single pool of resources"

MESOS

Container, Unikernel, Picoprocess, Hypervisors

openstack™

Xen™

docker

CoreOS

Offense and Scale

- We can break down offense into three areas
  - Vulnerability Research, Pentesting, Red Teaming

- How does the offense view architectures designed to scale?
  - The bottom of the stack moves slow
  - The top of the stack changes fast
  - Reliability is a transitive property
  - We can’t compete without automation

Offense and Scale

- The massive growth of infrastructures has resulted in the unexpected

- We rely too much on the 1 in a million defense
  - “In our business, one in a million is next Tuesday”
  - http://blogs.msdn.com/b/larryosterman/archive/2004/03/30/104165.aspx

- “Mean time between failure doesn’t scale” - @iamreallyfrank

Offense and Scale: Scenario 1

- You want to gain access to as many email accounts as possible

- Steal/buy a password list, take the most used password and try it one time against 1 billion email accounts
  - Low risk of massive account lockouts
  - Almost guaranteed to have a decent success rate

Offense and Scale: Scenario 2

- A forking HTTPD with an exploitable heap overflow
  - x64 Linux ASLR Heap has 28 bits of entropy
    - 268,435,456 possible base address values (2^28)
    - 268,435,456 / 200,000 httpd daemons = 1342 requests
    - 268,435,456 / 300,000 httpd daemons = 894 requests
    - 268,435,456 / 400,000 httpd daemons = 671 requests
    - 268,435,456 / 500,000 httpd daemons = 536 requests
  - Hint: 1342 requests per second was impressive in mid-2000s
  - The birthday problem applies to ASLR
  - Can you analyze a terabyte of logs and respond before the entire file system is exfiltrated?

Offense and Scale

- *“Heartbleed Team, National Security Agency. Their efforts resulted in a means to stop attempts to exploit a security vulnerability across the department’s **global network of more than eight million computing devices**.”*
    - http://www.defense.gov/releases/release.aspx?releaseid=17062

- *“**The exercise took place on a specially-constructed closed network designed to simulate the DoD and allied information networks and adversary networks.** The event also featured an expert opposing force, which takes on the role of the adversary, using a range of tactics and weapons to provide a realistic training environment.”*
    - http://www.defense.gov/news/newsarticle.aspx?id=123621
    - How can you simulate the real world scale (e.g. internet) these people operate in?

- *“The United States Department of Defense (DoD) Cyber Range is a **realistic simulation** and modeling network environment”*
    - http://www.tintri.com/customers/department-defense-cyber-range

Vulnerability Research at Scale

- Targeting
    - Ubiquitous software is a good start

- Data driven analysis
    - Where do you spend your limited resources and time?
    - Continuously consume and analyze vulnerability data

- Fuzzing
    - Containers can increase scale for certain applications
    - yFuzz - Automated fuzzing at scale with Docker

- Automation
    - Too much of our work is still a manual process

Offensive tools are not keeping up

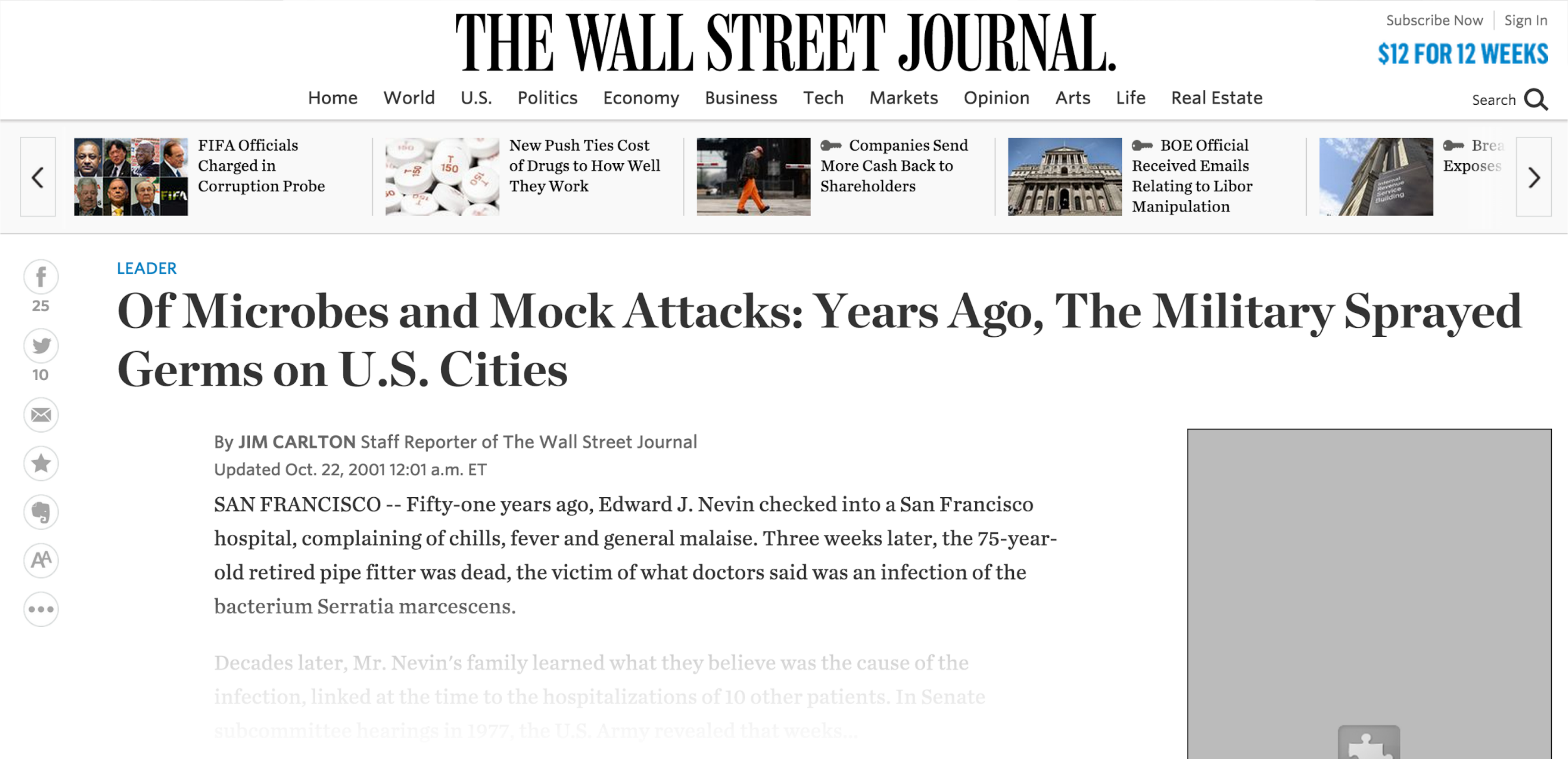
- Most public tools are not built to scale
  - No concept of one-to-many
  - Often built for and tested against 1 to 5 systems
  - Never able to handle more than STDOUT logging
  - Rely too heavily on a single language (Python)
  - Worms written in 2001 are more capable of lateral movement than most pentesters

- Majority of focus is on client side

Pentesting At Scale

- Vulnerability research has paid off, you have 0day to deploy to thousands of systems
    - Revisiting your tools is the next logical step

- Scenario: You have XXE 0day on a popular web framework and the network you are targeting has it widely deployed
    - Your favorite tool *abandoned_web_scanner.py* will take 4 days to complete
        - It has to perform a DNS lookup for each host and has no async resolver
        - The tool has no concept of threads, and even if it did Python has a GIL
    - My solution at Yahoo was to wrap libcurl with a thread pool library in C++
        - 10x as fast as the earlier Ruby implementation
        - Took no more than a week to develop
        - Can be extended to create many other web scanners


THE WALL STREET JOURNAL.
Subscribe Now
Sign In
$12 FOR 12 WEEKS
Home
World
U.S.
Politics
Economy
Business
Tech
Markets
Opinion
Arts
Life
Real Estate
Search
FIFA Officials Charged in Corruption Probe
New Push Ties Cost of Drugs to How Well They Work
Companies Send More Cash Back to Shareholders
BOE Official Received Emails Relating to Libor Manipulation
Brea
Exposes
25
10

LEADER

Of Microbes and Mock Attacks: Years Ago, The Military Sprayed Germs on U.S. Cities

By **JIM CARLTON** Staff Reporter of The Wall Street Journal
Updated Oct. 22, 2001 12:01 a.m. ET

SAN FRANCISCO -- Fifty-one years ago, Edward J. Nevin checked into a San Francisco hospital, complaining of chills, fever and general malaise. Three weeks later, the 75-year-old retired pipe fitter was dead, the victim of what doctors said was an infection of the bacterium Serratia marcescens.

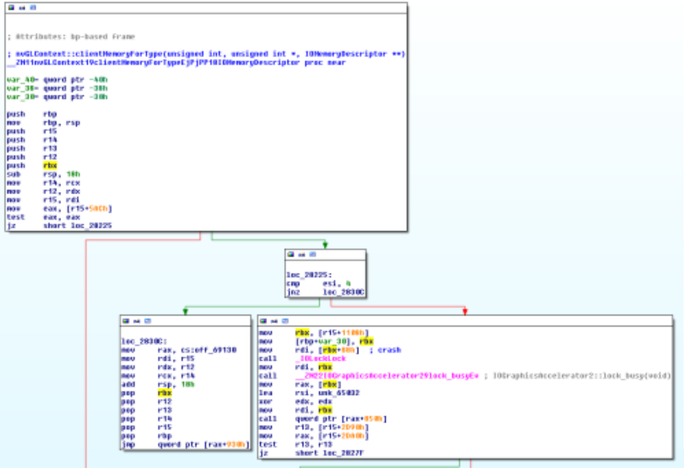
Decades later, Mr. Nevin's family learned what they believe was the cause of the infection, linked at the time to the hospitalizations of 10 other patients. In Senate subcommittee hearings in 1977, the U.S. Army revealed that weeks...

YAHOO!

Discovering attack surface

Physical

Code



```
; Attributes: bp-based frame

; nvGLContext::clientMemoryForType(unsigned int, unsigned int *, IOMemoryDescriptor **)
__ZN11nvGLContext19clientMemoryForTypeEjPjPP18IOMemoryDescriptor proc near

var_40= qword ptr -40h
var_38= qword ptr -38h
var_30= qword ptr -30h

push    rbp
mov     rbp, rsp
push    r15
push    r14
push    r13
push    r12
push    rbx
sub     rsp, 18h
mov     r14, rcx
mov     r12, rdx
mov     r15, rdi
mov     eax, [r15+5A8h]
test    eax, eax
jz      short loc_28225
```

```
loc_28225:
cmp     esi, 4
jnz     loc_2830C
```

```
loc_2830C:
mov     rax, cs:off_69130
mov     rdi, r15
mov     rdx, r12
mov     rcx, r14
add     rsp, 18h
pop     rbx
pop     r12
pop     r13
pop     r14
pop     r15
pop     rbp
jmp     qword ptr [rax+930h]
```

```
mov     rbx, [r15+1180h]
mov     [rbp+var_30], rbx
mov     rdi, [rbx+88h]  ; crash
call    _IOLockLock
mov     rdi, rbx
call    __ZN22IOGraphicsAccelerator29lock_busyEv ; IOGraphicsAccelerator2::lock_busy(void)
mov     rax, [rbx]
lea     rsi, unk_65032
xor     edx, edx
mov     rdi, rbx
call    qword ptr [rax+850h]
mov     r13, [r15+2D90h]
mov     rax, [r15+2DA0h]
test    r13, r13
jz      short loc_2827F
```

Network/Data

Sources: NYC MTA, http://www.martingrandjean.ch

Red Teaming At Scale

- An effective red team (and defenders) studies the real world tools, techniques and procedures from past attack campaigns
  - What about your attackers methods are different because of the size and scale of your operation?
  - Did their method of exfiltration take advantage of speed or could you have beaten them had you known what to look for?
  - Did the growth of the haystack make it harder to find the needle?
  - Does the maturity of your operation lend itself to faster needle searching?

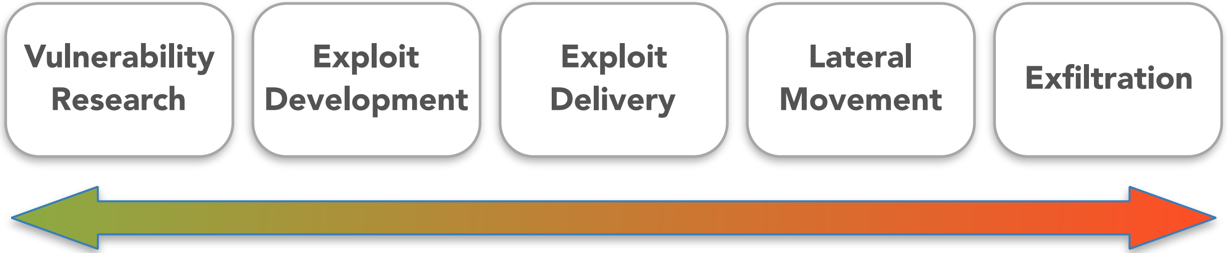
Attack Chains

- Scale can afford you the option of multiple paths to a target
  - This is an opportunity to avoid overlapping IOCs and toolchains

- Crash and/or evasion of security products
  - If defenders are using speed to their advantage in log collection and analysis then identify and exploit the slowest part of that process
  - NIDS are easily bypassed and taken down via memory consumption, NULL pointer dereferences and volume of noise
  - Many of these technologies fail open by design

- Continuous Integration / Continuous Delivery
  - Built in lateral movement

Automation



Vulnerability Research

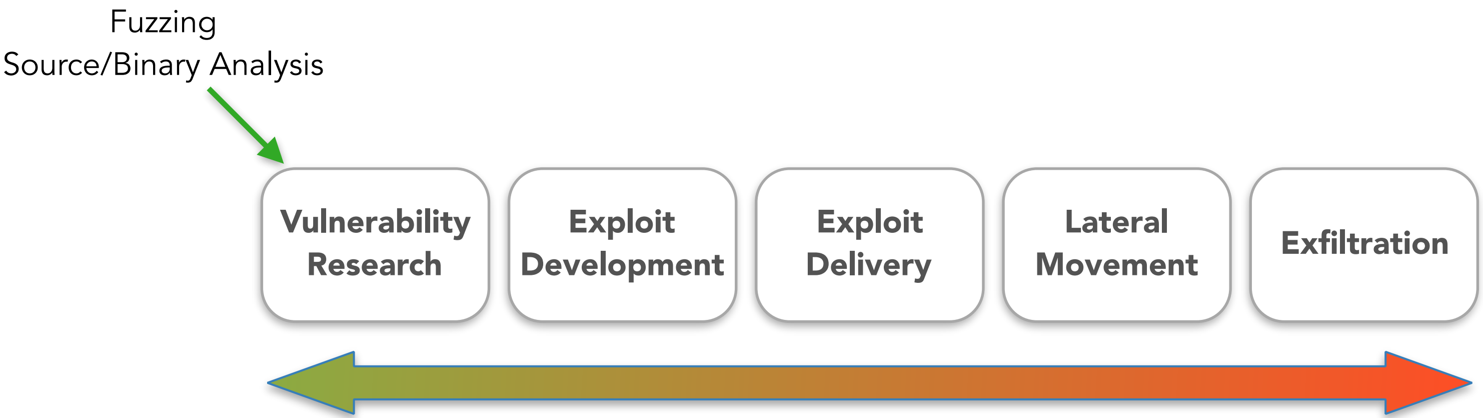
Exploit Development

Exploit Delivery

Lateral Movement

Exfiltration

Automation



Fuzzing
Source/Binary Analysis

Vulnerability Research

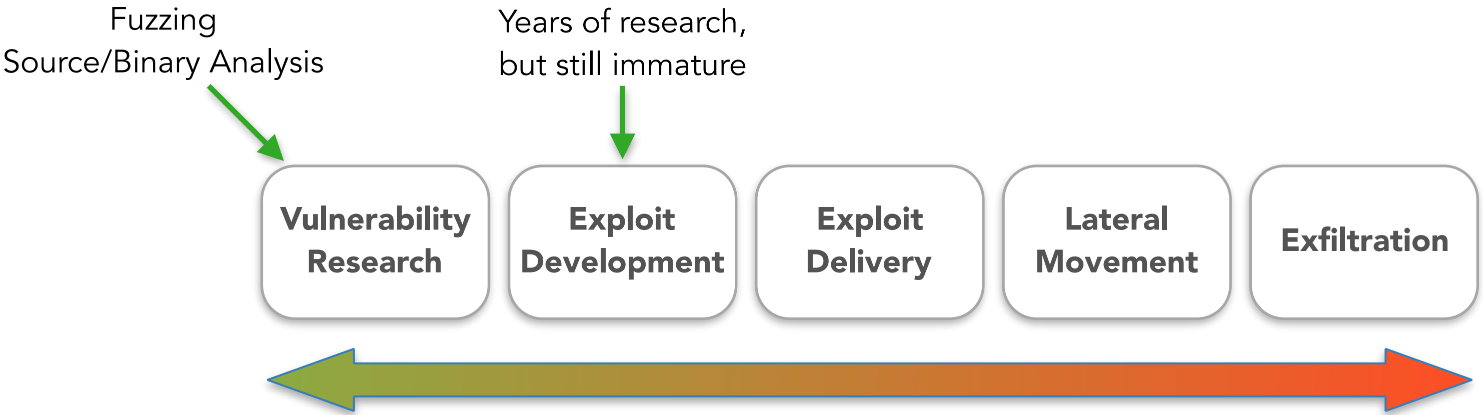
Exploit Development

Exploit Delivery

Lateral Movement

Exfiltration

Automation

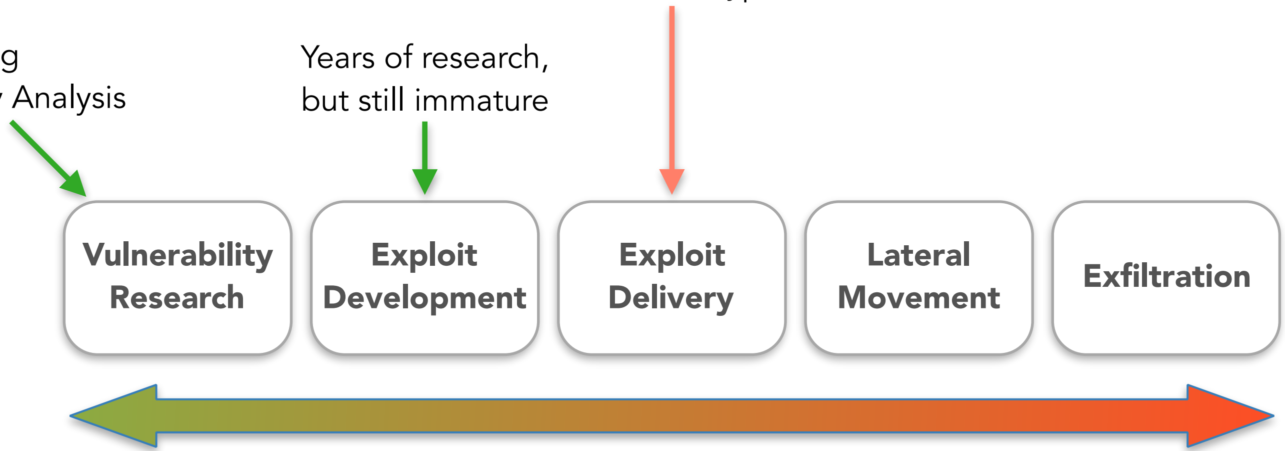

Fuzzing
Source/Binary Analysis
Years of research,
but still immature
Vulnerability Research
Exploit Development
Exploit Delivery
Lateral Movement
Exfiltration

Automation

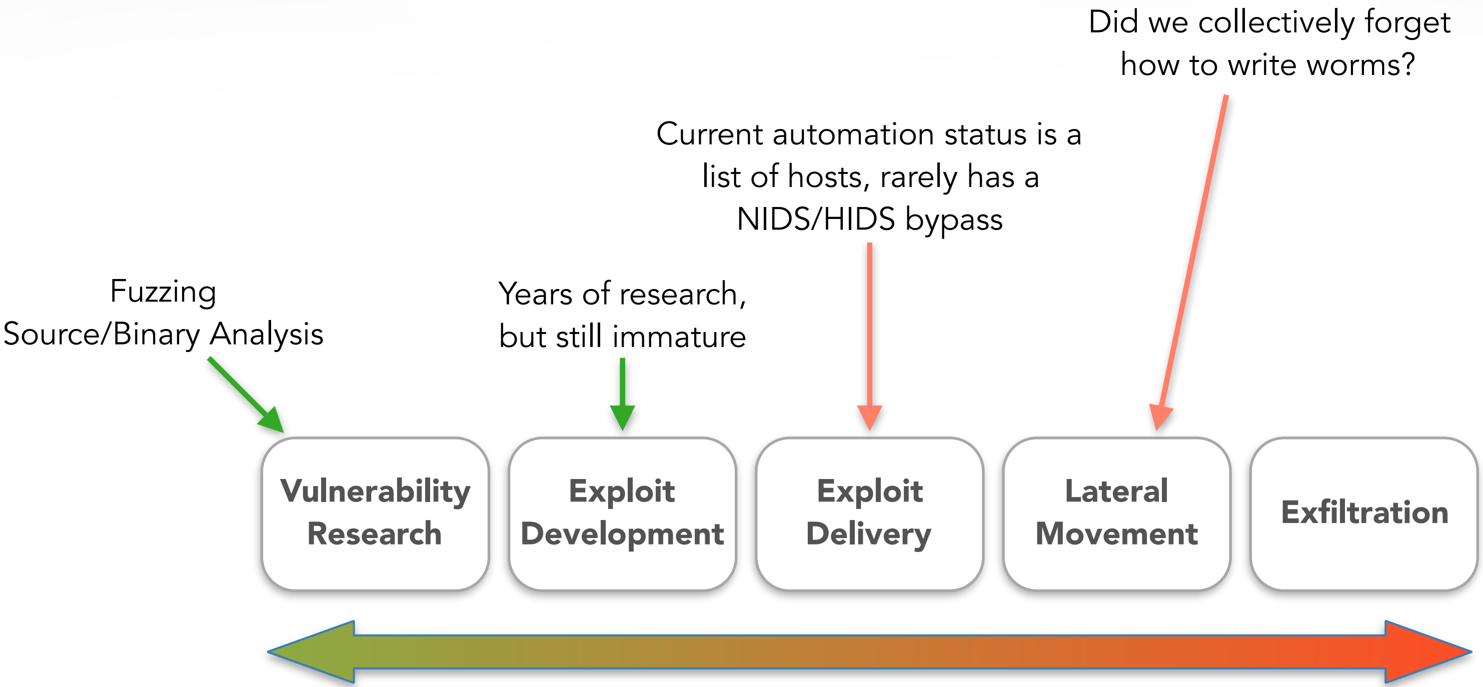
Current automation status is a list of hosts, rarely has a NIDS/HIDS bypass

Fuzzing
Source/Binary Analysis

Years of research, but still immature



Automation



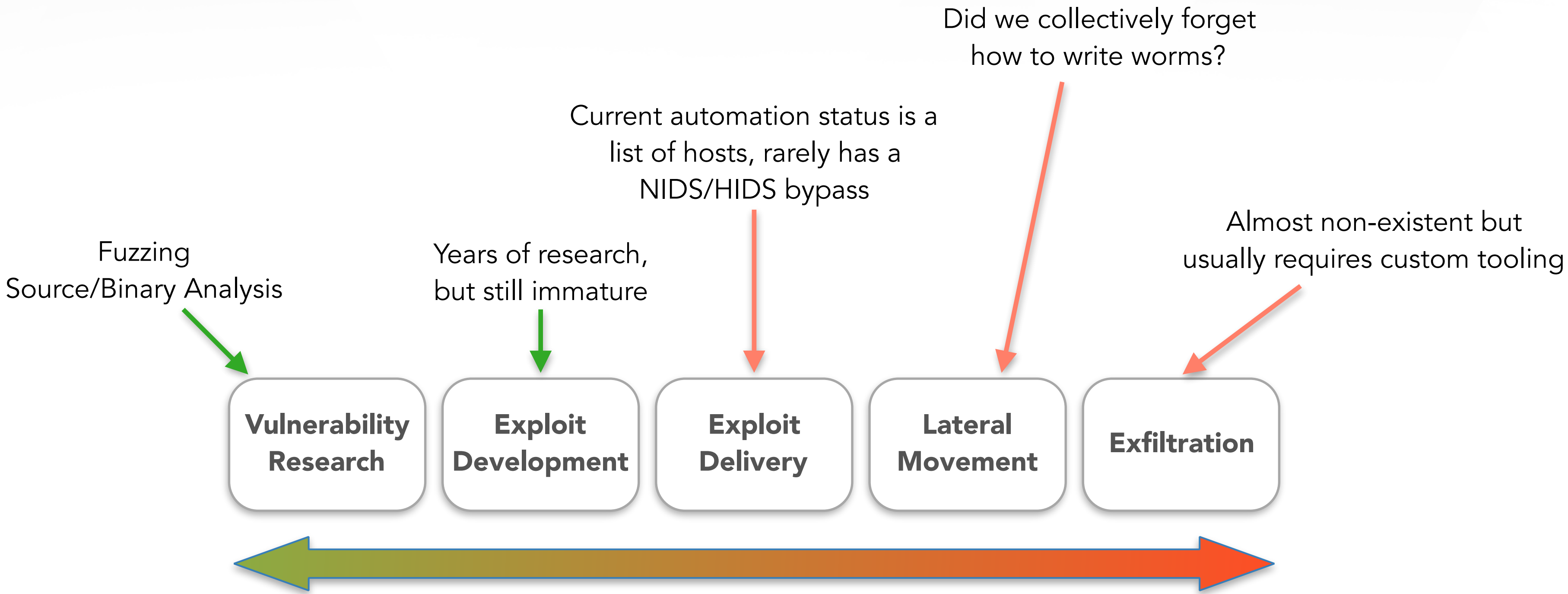
Fuzzing
Source/Binary Analysis

Years of research,
but still immature

Current automation status is a
list of hosts, rarely has a
NIDS/HIDS bypass

Did we collectively forget
how to write worms?

Automation



Did we collectively forget how to write worms?

Current automation status is a list of hosts, rarely has a NIDS/HIDS bypass

Almost non-existent but usually requires custom tooling

Fuzzing
Source/Binary Analysis

Years of research, but still immature

Vulnerability Research

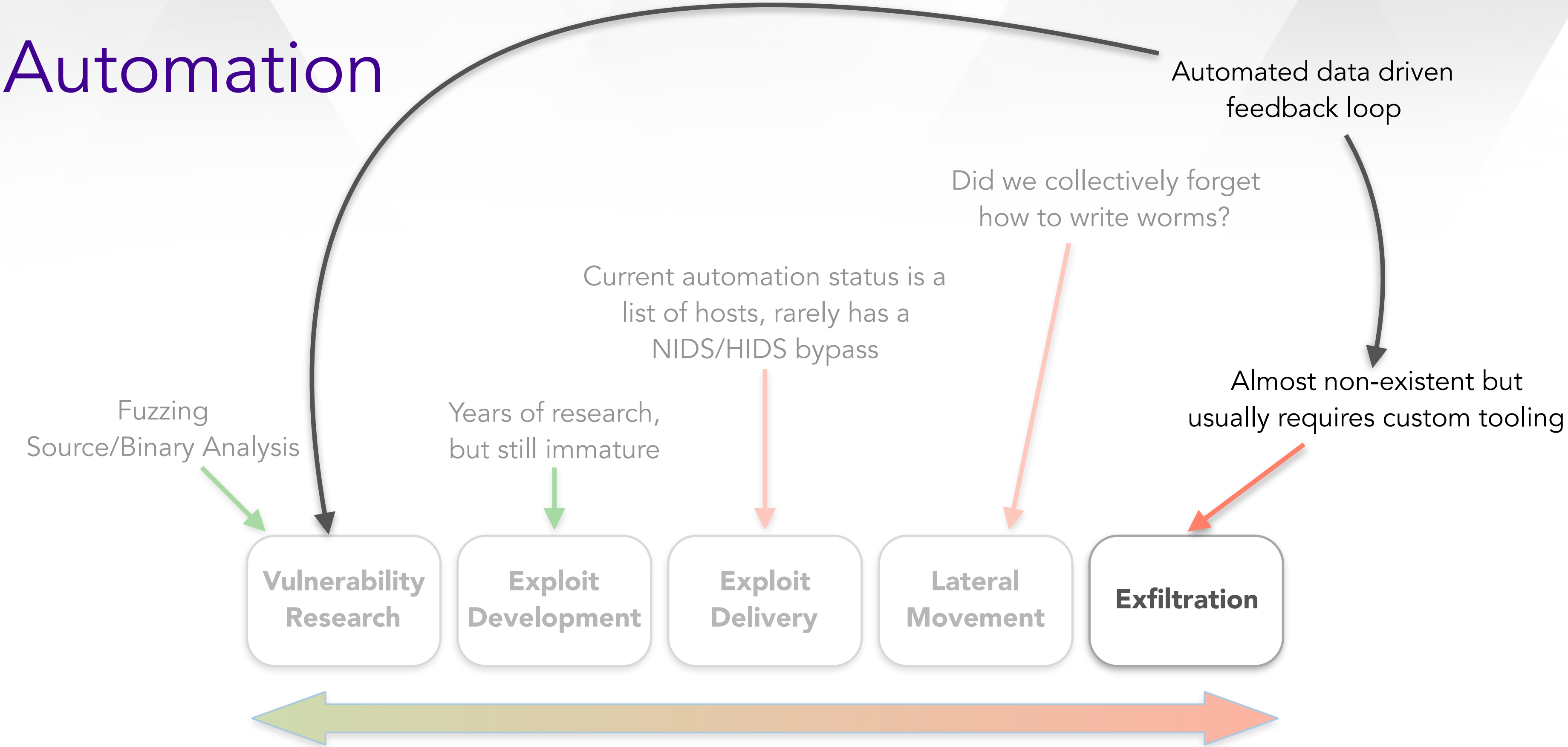
Exploit Development

Exploit Delivery

Lateral Movement

Exfiltration

Automation



Automated data driven feedback loop

Did we collectively forget how to write worms?

Current automation status is a list of hosts, rarely has a NIDS/HIDS bypass

Almost non-existent but usually requires custom tooling

Fuzzing
Source/Binary Analysis

Years of research, but still immature

YAHOO!

Post Compromise

- The ‘Low and Slow’ approach is out
- Exfiltration will have to meet or exceed speed of log collection/analysis
- There is no HIDS with a knowledge of containers
  - Containers have a shared kernel
  - Until Docker meets SECCOMP-BPF, the bar for compromising another container is lower than a hypervisor
  - HIDS will need some time to catch up

The Future Of Disposable Environments

- Hypervisors, Containers, Unikernels, Picoprocesses
  - Scenario: It’s 2020 and every web server request is handled by a picoprocess on demand that is terminated upon TCP FIN
    - Exfiltration must be automated, over TLS and faster than log analysis can detect it
    - Can your current toolset do this?
    - Can your current toolset catch this?
- Vulnerabilities that allow an attacker to escape these environments are valuable
  - Row Hammer
  - Hypervisor memory corruption (http://xenbits.xen.org/xsa)

Advice for defenders

- Hardened targets with dense bugs are still more secure than targets with no hardening and sparse bugs (e.g. Chrome vs Bash)

- A moving target is always harder to exploit
  - Reduces lifetime and increases cost of an exploit
  - CI/CD can be leveraged to achieve this

- Security staff levels
  - %1 of total headcount
  - Security staff that focuses on enterprise systems should be measured against system count
  - Security staff focused on product should be measured by application complexity

Advice for defenders

- Attacker asymmetry, like everything else, grows linearly with scale

- Scale has a way of magnifying even the smallest security problems
  - “1 in a million is next Tuesday”

- Work with what you’ve got and put pressure on your vendors to deliver secure software

- The future of defending scale is *O(n)* where *n* is a list of systems
  - Is your *detect_respond()* faster than *identify_pwn()* ?

The End

Scale?^H^H^H^H Questions?

YAHOO!