Engineering Cyber-Deceptive Software

Kevin Hamlen (UTD) and Frederico Araujo (IBM)

Supported in part by ONR N00014-14-1-0030, AFOSR FA9550-14-1-0173, NSF CAREER 1054629, and NSA H98230-15-1-0271
Any opinions, recommendations, or conclusions are those of the authors, and not necessarily of ONR, AFOSR, NSF, and NSA
The threat landscape
Scalability Problems in Cyber Defense

The chart shows a significant increase in reported federal cybersecurity incidents from 2007 to 2017. The number of incidents has risen from around 5 in 2007 to over 150,000 in 2017. Concurrently, federal cybersecurity spending has also increased, with a noticeable spike from 2014 onwards. The chart indicates a strong correlation between the rise in incidents and the increase in spending.
Conventional cyber-defense mechanisms

• **Attacker-defender asymmetry**
  – advantage on the side of the attackers

• **Predictability** of current defenses
  – attackers rely on *expected* responses to move their attack goals forward

• **Superficial** attack data collection
  – stop most attackers too early without deterring advanced attackers
The Cyber Kill Chain [credit: Netskope’19]

1. Recon
2. Weaponize
3. Delivery
4. Exploit
5. Install
6. Callback
7. Persist

a missed opportunity for thwarting attacks?
1. Cyber-deception
   • Leveling the asymmetry through cyber-deception

2. Honey-patching
   • A new software cyber-deception technology

3. Process image secret redaction
   • Process image secret redaction through taint analysis
Cyber-deception defense

computer-security deception —

1: *noun*. "Planned actions taken to mislead attackers and to thereby cause them to take (or not take) specific actions that aid computer security defenses." **J. J. Yuill, 2006**

"All warfare is based on deception."
— **Sun Tzu, The Art of War**


"The Cuckoo's Egg", Clifford Stoll, 1989
The deception stack

Pingree, L., Gartner, July 2015

- Network
  - IP Address hopping
  - Deceptive Firewalls
  - Deceptive appliances

- Endpoint
  - Protocol emulation
  - Distributed decoys
  - Honeypots

- Application
  - ?

- Data
  - Honey-tokens
  - Decoy files
  - Baits & beacons

Deceit Difficulty
The deception stack

Pingree, L., Gartner, July 2015

IP Address hopping
Deceptive Firewalls
Deceptive appliances

Protocol emulation
Distributed decoys
Honeypots

Honeytokens
Decoy files
Baits & beacons
Science of *deception-facilitating* software engineering

"Introduces a **language-level** methodology for arming live, **legacy** server software with **deceptive** attack-response capabilities."
Advantages of cyber-deception defense

- Can change the **economics** of hacking
  - more difficult, time consuming, cost prohibitive
  - pushes Nash equilibrium towards defender (Carroll & Grosu, 2011)
- Increased attacker **risk** and **uncertainty**
  - attackers cannot rely on what they learn
- Attacker **disinformation**
  - fake successes, responses, files and assets to exploit
- Gain important **information** about the adversary
  - monitoring of attacker actions and honey-data misuse
Background: honeypots

**Definition:** a computer security mechanism set to entice, detect, and monitor attempts at unauthorized use of information systems.

- **Prior research:**
  - Virtual Honeypots, Shadow Honeypots, Hybrid Honeypots, ...

- **Honeypots tradeoffs:**

  - **Low-Interaction:** simulates some parts of an OS (e.g., Network Stack)
    - **Pros:** low-cost, easy to deploy and maintain, safe
    - **Cons:** gathers poor-quality attacker data, easily detectable

  - **High-Interaction:** provides a real system attackers can interact with
    - **Pros:** gathers high-quality attacker data, with further levels of interaction
    - **Cons:** prohibitively hard to build and to maintain, high-cost
A new software cyber-deception technology

HONEY-PATCHING
Attacker Methodology: Probing

web server (patched)

probe payload

web server (unpatched)

attacker

Targets
74.125.205.231
74.125.245.177
74.125.235.122
74.125.217.103
...
Attacker Methodology: Probing

MAJOR SITES AFFECTED BY HEARTBLEED
THE PASSWORDS YOU SHOULD CHANGE AND THE PERSONAL INFORMATION AT STAKE

Vulnerable to Heartbleed?
Yes
No

Should you change your password?
Yes
No

Social Media
Email
Financial Institutions
Other Popular Sites

Key
Revealing Information Including Names, Social Security numbers, personal contact details and other personal information.
Sensitive information including credit card numbers, account statements, tax info and security information.
Sensitive information including credit card numbers, account statements, tax info and security information.
Information is not shared with others.

Business information including proprietary documents, operational procedures, legal info, customer rights, and customer information.

Shield that don’t use OpenSSL.

Facebook
Twitter
Instagram
Pinterest
Google+
YouTube
Amazon
Flickr
LinkedIn
Windows
Slideshow

Google
Yahoo!
AOL
News

Goldman Sachs
Citigroup
Chase
TD Bank

Dropbox
MailChimp

Amazon Web Services
OpenSSL

Generally, financial institutions don’t use OpenSSL, and even “that one time” you thought you had been affected by Heartbleed...
Cyber-deception: Honey-patching

- **Targets**
  - 74.125.205.231
  - 74.125.245.177
  - 74.125.215.214
  - 74.125.235.122
  - 74.125.217.103
  - ...

- **Red Herring**
  - decoy clone (honey-patched)
  - "fake credit card #"
Honey-patch —

1: *noun*. A software security patch that **closes a vulnerability**, but makes attempted exploits of the vulnerability **appear successful**.

2: *verb*. To **apply** a honey-patch to software (often while mischievously cackling).
Advantages

• Frustrate attacker vulnerability-probing
  – mask patching lapses
  – increase attacker risk

• Collect preparatory counterreconnaissance against directed attacks
  – Honeypot lives inside the live server, not as a separate decoy machine

• Unique opportunities for attacker disinformation and misdirection
  – Keep attackers “on the hook” longer
  – “Leak” arbitrary (fake) secrets
  – Fool attackers into disclosing their “real” payloads
Challenges

• Efficiency
  – minimal *overhead* for legitimate users
  – perform well enough for attackers that attack failures are not *advertised*
  – decoy deployment must be *fast*

• Reliability
  – no perceptible *disruption* in the target application
    → migration of attacker sessions over decoys must happen “live”
  – *established connections* must not be broken
  – provide same security of vendor-supplied patches

• Secrets redaction
  – “*in-memory*” and filesystem secrets must not be leaked to decoys

• Generality
  – should work for many server types

• Maintainability
  – modular and generic, need only a superficial understanding to be deployed
  – easiness of reformulating vendor-provided patches into honey-patches
Approach

• Recent advances in *process migration* for HPC
  – “checkpoint-restart”
    *act of transferring a running process between two nodes by dumping its state on the source and resuming its execution on the destination*
  – “key” to make our idea workable
  – **CRIU** (Checkpoint/Restore In Userspace) – http://criu.org

• OS-Level virtualization through LXC (*Linux Containers*)
  – made it possible to create lightweight decoy servers on-demand
  – lightweight sandboxing for attacker session isolation
  – **overlayfs** for fast provisioning of decoys

• Proof of feasibility
  – Apache HTTP (~2.2M SLOC)
  – Nginx
  – Lighttpd

### Market share: top million busiest sites

<table>
<thead>
<tr>
<th>Developer</th>
<th>Jan. 2014</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>554,533</td>
<td>55.45%</td>
</tr>
<tr>
<td>nginx</td>
<td>159,079</td>
<td>15.91%</td>
</tr>
<tr>
<td>Microsoft</td>
<td>126,568</td>
<td>12.66%</td>
</tr>
<tr>
<td>Google</td>
<td>30,370</td>
<td>3.04%</td>
</tr>
</tbody>
</table>

Source: http://news.netcraft.com
Framework

```
patch
1 + if (attack detected)
2 + reject;
```

```
honey-patch
1 + if (attack detected)
2 + fork to decoy;
```
Outline

1. Overview
2. Challenges & Approach
3. Architecture
4. Case Study: Heartbleed
5. Evaluation
6. Conclusions
Attack Walkthrough

1. GET /malicious
2. GET /malicious
3. fork
4. checkpoint
5. restore
6. HTTP response
7. HTTP response

Legend:
- request message
- response message
- CR operation

From Patches to Honey-Patches: Lightweight Attacker Misdirection, Deception, and Disinformation
Honey-patching mechanism:
• Implemented as a tiny C library (~270 SLOC)

Listing 1: hp_fork function

```c
void hp_fork()
{
  read_context(); // read context (target/decoy)
  if (decoy) return; // if in decoy, do nothing
  register_handler(); // register signal handler
  request_fork(); // fork session to decoy
  wait(); // wait until fork process has finished
  save_context(); // save context and resume
}
```
LXC Controller

- Manages a pool of Linux containers
- Exposes two operations: **acquire** and **release**
- **overlayfs** for efficient container creation
- ~190 SLOC in Node.js – fully asynchronous
CR-Service

- Process-CR daemon running on containers
  - Extends **CRIU** with
    - Memory redaction at checkpoint
    - Transparent connection relocation at restore
- Implements a remote *façade* for CR operations
  - RPC protocol based on Protocol Buffers over IPC sockets
- Filesystem sanitization & attacker session signaling
- ~710 SLOC mostly in C

CR-Controller

- Implemented as an external module to the proxy
- ~450 SLOC mostly in C++
Memory Redaction

GET / HTTP/1.1 /browse/doc1.html
en_US xyz-198 8229788/6160/11/
Accept-Encoding: gzip, deflate, sdch...
Accept-Language: en-US, en; q=0.8...
Cookie: app.token= BACC-76GF-ABS3-ZOV2
SESSIONID=2321CFA5DA771A284D13DD67798A
GET / HTTP/1.1 /browse/doc1.html
Cookie: app.token= BACC-76GF-ABS3-ZOV2

--.- collect process group info
build process tree
collect state information
performs memory redaction using $pid and $tid
write process tree image files to disk
Connection Relocation

- Fast, *transparent* migration of attack sessions to decoys
- TCP connection **repair** option

```
tsk ← create new connection ( )

<p>| tsk = create_socket (new bounds) |
| bind(tsk)                        |</p>
<table>
<thead>
<tr>
<th>connect(tsk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>close silently (tsk)</td>
</tr>
</tbody>
</table>

| enter_repair_mode(tsk)           |
| close(tsk)                       |

<table>
<thead>
<tr>
<th>transfer state (tsk, sk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind(sk)</td>
</tr>
<tr>
<td>connect(sk)</td>
</tr>
<tr>
<td>transfer_opts(tsk, sk)</td>
</tr>
<tr>
<td>transfer_queues(tsk, sk)</td>
</tr>
</tbody>
</table>
```

Reverse Proxy

Target
Decoy
Outline

1. Overview
2. Challenges & Approach
3. Architecture
4. Case Study: Heartbleed
5. Evaluation
6. Conclusions
Heartbleed

HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "POTATO" (6 LETTERS).

User Meg wants these 6 letters: POTATO.
User 12a wants pages about "irl games". Unlocking secure records with master key 5130985753439.
Secure connection using key "4538538374224".

User Meg wants these 6 letters: POTATO.

Source: http://xkcd.com/1354/
Heartbleed

HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).

User Meg wants these 500 letters: HAT.

Lucas requests the "missed connections" page. Eve (administrator) wants to set server’s master key to "14835038534". Isabel wants pages about snakes but not too long. User Karen wants to change account password to "passw0rd". Here, hacker requests pass!

Source: http://xkcd.com/1354/
Heartbleed

Listing 2: Abbreviated patch for Heartbleed

1    ...
2    + if (1 + 2 + payload + 16 > s->s3->rrec.length)
3    + return 0; /* silently discard */
4    ...

Listing 3: Honey-patch for Heartbleed

1    ...
2    + if (1 + 2 + payload + 16 > s->s3->rrec.length) {
3    +  hp_fork();  /* attack detected */
4    +  hp_skip(return 0); /* skip remediation in decoy */
5    + }
6    ...

From Patches to Honey-Patches: Lightweight Attacker Misdirection, Deception, and Disinformation
Analysis of Apache security vulnerabilities

Honey-patchable patches = ~65% (49 out of 75 patches)
Not Easily Honey-Patchable Patch

Listing 4: Abbreviated patch for CVE-2013-1862

```
logline = apr_psprintf(r->pool, ..., 
... 
- ap_get_server_name(r),
+ ap_escape_logitem(r->pool, ...(r)),
... 
```
Outline

1. Overview
2. Challenges & Approach
3. Architecture
4. Case Study: Heartbleed
5. Evaluation
6. Conclusions
Performance

Stress test illustrating request throughput for a 3-node, load-balanced setup

Average RTT = 4.25 ms
Average Throughput = 240 req/sec
Malicious HTTP request round-trip times for different web servers

<table>
<thead>
<tr>
<th>Web Server</th>
<th>Median Round-Trip Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache HTTP</td>
<td>0.138</td>
</tr>
<tr>
<td>Lighttpd</td>
<td>0.092</td>
</tr>
<tr>
<td>Nginx</td>
<td>0.095</td>
</tr>
<tr>
<td>Nginx (with redaction)</td>
<td>0.139</td>
</tr>
<tr>
<td>Nginx (no redaction)</td>
<td>0.157</td>
</tr>
</tbody>
</table>
Effect of concurrent attacks on legitimate HTTP request round-trip time on a single-node VM
Performance

Effect of payload size on malicious HTTP request round-trip time

![Graph showing the effect of payload size on malicious HTTP request round-trip time. The graph compares the median round-trip time with and without redaction for different payload sizes (KB).]
Generality

Analysis of Apache security vulnerabilities

Honey-patchable patches = ~65% (49 out of 75 patches)
# Honey-Patched Vulnerabilities

Table 1: Honey-patched security vulnerabilities

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
<th>CVE-ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>2.2.21</td>
<td>CVE-2011-3368</td>
<td>Improper URL Validation</td>
</tr>
<tr>
<td></td>
<td>2.2.9</td>
<td>CVE-2010-2791</td>
<td>Improper timeouts of keep-alive connections</td>
</tr>
<tr>
<td></td>
<td>2.2.15</td>
<td>CVE-2010-1452</td>
<td>Bad request handling</td>
</tr>
<tr>
<td></td>
<td>2.2.11</td>
<td>CVE-2009-1890</td>
<td>Request content length out of bounds</td>
</tr>
<tr>
<td>OpenSSL</td>
<td>1.0.1f</td>
<td>CVE-2014-0160</td>
<td>Buffer over-read in heartbeat extension</td>
</tr>
<tr>
<td>Bash</td>
<td>4.3</td>
<td>CVE-2014-6271</td>
<td>Improper parsing of environment variables</td>
</tr>
</tbody>
</table>

**Heartbleed!**

**Shellshock!**
Use case: Heartbleed

**The Economic Times**

**Software**

*New technique Red Herring fights 'Heartbleed' virus*

PTI Apr 15, 2014, 05:17PM IST

WASHINGTON: US cybersecurity researchers have developed a technique that fights the 'Heartbleed', and detects and entrap hackers who might be using it to steal sensitive data.

The Heartbleed bug, which became public last week, has set alarm bells ringing across the globe, including in India, for fear of exposing millions of passwords, credit card numbers and other sensitive information to hackers.
From Patches to Honey-Patches: Lightweight Attacker Misdirection, Deception, and Disinformation
Deception ≠ Obscurity/Obfuscation

Security by Obscurity/Obfuscation
- Attacker **uncertainty**
- Obstruct attacks
- Secrets **ill-defined**
  - e.g., impede reverse-engineer “understanding”
- **Ad hoc** secret protection

Deceptive Cyber-defense
- Attacker **false certainty**
- Refine defenses
- Formally defined secrets
  - e.g., indistinguishable overhead distribution
- **Quantifiable/provable** secret protection
Conclusions

• **Traditional patching**
  – Fix vulnerability by simply **rejecting** malicious requests
  – Missed **opportunity** for defenders!

• **Honey-patching**
  – source-level patching strategy that **raises attacker risk** and **uncertainty**
  – can be realized for **large-scale, performance-critical** software with **minimal overheads** to users

• Adoption could significantly **impede vulnerability probing**

• Offer defenders a new, potent tool for **attacker deception**