Accidental Data Leak Detection, Secure Coding, and Payment Card Ecosystem (Part 2)

Danfeng (Daphne) Yao’
Professor of Computer Science
Virginia Tech

Brown EMCS Oct 2019
Acknowledgments

Stefan Nagy (VT)
Xiaokui Shu (IBM Research)
Sazzadur Rahaman (VT)
Fang Liu (Palo Alto Networks)
Jing Zhang (AMD)

Na Meng (VT)
Elisa Bertino (Purdue)
Gang Wang (UIUC)
Ali Butt (VT)
Wu Feng (VT)
Smart virus X:

1. if Scanner says no
2. then infect;
3. else do nothing;

AV Scanner Thinks

- X is a virus
  - Contradicts
- X is not a virus
  - Contradicts

Actual Behavior of X

- X chooses not to infect
- X chooses to infect

[Fred Cohen, J. of Virology 1987]
Absolute security is impossible. But people make money on security all the time.

Why?
Real quotes from the StackOverflow forum

“Adding \texttt{csrf().disable()} solved the issue!!! I have no idea why it was enabled by default”

“adding -D\texttt{trust\_all\_cert}=true to VM arguments”

“I want my client to accept any certificate (because I'm only ever pointing to one server)”

[Meng, Yao, et al. ICSE 2018] We examined 497 Java and security related StackOverflow Posts
Writing secure code is tough

```java
// Create a trust manager that does not validate certificate chains
TrustManager[] trustAllCerts = new TrustManager[] {
    new X509TrustManager() {
        public java.security.cert.X509Certificate[]
            getAcceptedIssuers() { return null; }
    },
    public void checkClientTrusted(...) {}
    public void checkServerTrusted(...) {} });

// Install the all-trusting trust manager
try {
    SSLContext sc = SSLContext.getInstance("SSL");
    sc.init(null, trustAllCerts, new java.security.
        SecureRandom() );
    HttpsURLConnection.setDefaultSSLSSLSocketFactory(sc.
        getSocketFactory() );
} catch (Exception e) {} 
```

[Meng, Yao, et al. ICSE 2018]
How Much Influence Does StackOverflow Have?

<table>
<thead>
<tr>
<th>Insecure Posts</th>
<th>Total Views</th>
<th>No. of Posts</th>
<th>Min Views</th>
<th>Max Views</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabling CSRF Protection*</td>
<td>39,863</td>
<td>5</td>
<td>261</td>
<td>28,183</td>
<td>7,258</td>
</tr>
<tr>
<td>Trust All Certs</td>
<td>491,567</td>
<td>9</td>
<td>95</td>
<td>391,464</td>
<td>58,594</td>
</tr>
<tr>
<td>Obsolete Hash</td>
<td>91,492</td>
<td>3</td>
<td>1,897</td>
<td>86,070</td>
<td>30,497</td>
</tr>
<tr>
<td>Total Views</td>
<td>622,922</td>
<td>17</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

StackOverflow posts that make insecure suggestions have a large influence on developers.

* In Java Spring Security for web applications
Social Dynamics on Stackoverflow

“Do NOT EVER trust all certificates. That is very dangerous.”

“the "accepted answer" is wrong and INDEED it is DANGEROUS. Others who blindly copy that code should know this.”

“once you have sufficient reputation you will be able to comment”

“If you don't have enough rep to comment, ... then participate ... until you have enough rep.”

https://stackoverflow.com/questions/10594000/when-i-try-to-convert-a-string-with-certificate-exception-is-raised
Deployment-quality Accuracy and Scalability

Maximum & minimum LoC: 2,571K (Hadoop), 1.1K (Commons Crypto); and average LoC: 402K
Security Issues we found in professionally developed Apache software projects
How to measure the quality of PCI scanners?

Can We Measure the Strength of PCI Enforcement?

Our BuggyCart Testbed embeds 35 vulnerabilities (to open source)

- Network security (14 test cases)
- System security (7 test cases)
- Web Application security (8 test cases)
- Secure storage (6 test cases) – cannot be detected by external scans
Our BuggyCart Testbed and Commercial PCI Scanners Selected

<table>
<thead>
<tr>
<th>PCI Scanners</th>
<th>Price</th>
<th>Spent Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanner 1</td>
<td>$2,995/Year</td>
<td>$0 (Trial)</td>
</tr>
<tr>
<td>Scanner 2</td>
<td>$2,190/Year</td>
<td>$0 (Trial)</td>
</tr>
<tr>
<td>Scanner 3</td>
<td>$67/Month</td>
<td>$335</td>
</tr>
<tr>
<td>Scanner 4</td>
<td>$495/Year</td>
<td>$495</td>
</tr>
<tr>
<td>Scanner 5</td>
<td>$250/Year</td>
<td>$250</td>
</tr>
<tr>
<td>Scanner 6</td>
<td>$59/Quarter</td>
<td>$118</td>
</tr>
<tr>
<td>Scanner 7</td>
<td>Unknown</td>
<td>N/A</td>
</tr>
<tr>
<td>Scanner 8</td>
<td>$350/Year</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>-</td>
<td><strong>$1198</strong></td>
</tr>
</tbody>
</table>
Web Security Test Cases Are Particularly Weak

<table>
<thead>
<tr>
<th>Req.</th>
<th>Test Cases</th>
<th>Vul. Location</th>
<th>Is Within ASV Scope?</th>
<th>Scanner 1</th>
<th>Scanner 2</th>
<th>Scanner 3</th>
<th>Scanner 4</th>
<th>Scanner 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Baseline</td>
<td>Certified</td>
<td>Baseline</td>
<td>Certified</td>
<td>Baseline</td>
</tr>
<tr>
<td>26.</td>
<td>Sql Inject in Admin Login</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>27.</td>
<td>Sql Inject in Customer Login</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>28.</td>
<td>Disable password retry restriction</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>29.</td>
<td>Allow passwords with len &lt;8</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>30.</td>
<td>Javascript source integrity check</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>✗</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>31.</td>
<td>Don’t hide program crashes</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>32.</td>
<td>Implant XSS</td>
<td>Webapp</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>33.</td>
<td>Implant CSRF</td>
<td>Webapp</td>
<td>Yes</td>
<td>✮</td>
<td>✗</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
## Worrisome PCI scanners security – Summary of Testbed Results

<table>
<thead>
<tr>
<th></th>
<th>Scanner 1</th>
<th>Scanner 2</th>
<th>Scanner 3</th>
<th>Scanner 4</th>
<th>Scanner 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Vul. Detected</td>
<td>21</td>
<td>16</td>
<td>17</td>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>(29 Total*)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Vul. Remaining in</td>
<td>7</td>
<td>15</td>
<td>18</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>Certified Ver.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Vul. detected,</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>but did not fix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*All 29 vulnerabilities violate the PCI’s data security specifications and are required by the specifications to be removed.*
Assessed 1203 e-commerce sites with our PCICheckerLite tool.

<table>
<thead>
<tr>
<th>Category (810)</th>
<th>#Vul. Websites</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>At least 1</td>
</tr>
<tr>
<td>Business</td>
<td>113</td>
</tr>
<tr>
<td>Shopping</td>
<td>143</td>
</tr>
<tr>
<td>Arts</td>
<td>76</td>
</tr>
<tr>
<td>Adults</td>
<td>65</td>
</tr>
<tr>
<td>Recreation</td>
<td>75</td>
</tr>
<tr>
<td>Computer</td>
<td>56</td>
</tr>
<tr>
<td>Games</td>
<td>42</td>
</tr>
<tr>
<td>Health</td>
<td>55</td>
</tr>
<tr>
<td>Home</td>
<td>93</td>
</tr>
<tr>
<td>Kids &amp; Teens</td>
<td>36</td>
</tr>
<tr>
<td>Total (1,203)</td>
<td>1,135 (94%)</td>
</tr>
</tbody>
</table>

Using HTTP to transmit Signup form.

Self-signed certificate.

Wrong hostname.
Key PCI Takeaways

5 out of 6 PCI scanners certify vulnerable merchant sites

94% websites (out of 1,203) Not PCI compliant

[Rahaman, Wang, Yao. ACM CCS 2019]
Data Leak Detection as an Add-on Service by Cloud Providers to Prevent Data Exposure
Clients do not want the cloud/detection providers to learn about the sensitive information. How?
The Basic Set-Intersection Approach
Adding some twists to the set intersection based detection approach

N-gram generation & Rabin fingerprints

Sensitive Data: between them.<ref>RFC 1812

N-grams

[“bet”, “etw”, “twe”, “wee”, “een”, “en”, “nt”, “th”, “the”, “htm”…]

Rabin fingerprints

{7872, 3408, 2410, 1961, 4238, 2383, 2158, 8131, 3271, 1833…}

Intersection

{… 7872, 3408, 2410, 1961, 4238, 3271, 1833…}

Rabin fingerprints

{…7872, 3408, 2410, 1961, 4238, 4340, 1666, 1008, 3271, 1833…}

N-grams

[“bet”, “etw”, “twe”, “wee”, “een”, “en+”, “nt”, “+th”, “the”, “htm”…]

Content

The difference between+them.%26lt%3Br%26gt%3BRFC+1812

[Shu and Yao. SecureComm ‘12] [Shu, Yao, and Bertino. IEEE TIFS ’15]
A Twist -- Fuzzy Fingerprints

[Shu, Yao, and Bertino. *IEEE TIFS ’15*]

*Top 25 most downloaded article of IEEE Signal Processing Society in 2018*
Another work: Detection of transformed accidental data leak?

Auto-formatting (WordPress)

The application layer contains the higher-level protocols used by most applications for network communication. Examples of application layer protocols include the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP). Data encoded according to application layer protocols are then encapsulated into one or (occasionally) more transport layer protocols (such as TCP or UDP), which in turn use lower layer protocols to effect actual data transfer.

Partial source code leak

```python
def encode(msg, pubkey, verbose=False):
    chunksize = int(log2(pubkey.modulus, 256))
    outchunk = chunksize + 1
    outfmt = '%%0%dx' % (outchunk * 2)
    bmsg = msg if isinstance(msg, binary_type) else msg
    result = []

    for start in range(func(0, len(bmsg), chunksize)):
        chunk = bmsg[start:start + chunksize]
        chunksize = len(chunk)
        encoded = hexlify(chunk, 16)
        coded = pow(encoded, *pubkey)
        bencoded = unhexlify((outfmt % coded).encode())
        if verbose:
            print('encoded: ', chunksize, chunk, plain, bencoded)
        result.append(bencoded)

    return b''.join(result).rstrip(b'\x00').decode('utf-8')
```

```python
def _delitem(self, item):
    self.heap = [(v, k) for v, k in self.heap if k != item]
    chunk = b'\x00' * (chunksize - len(chunk))
    heapq.heapify(self.heap)
    self.heap = [(v, k) for v, k in self.heap if k != item]
    self.heap = [(v, k) for v, k in self.heap if k != item]
    return smallest
```

[Shu, et al. IEEE TIFS 2016]
Transformed data leak –
Our sequence-alignment based detection

Sequence 1

Sequence 2

[Shu, et al. IEEE TIFS 2016]
Also invented a smart sampling algorithm

2 identical input streams:

\[
1, 9, 4, 5, 3, 5, 9, 7, 6, 6, 3, 3, 7, 1 \\
1, 9, 4, 5, 3, 5, 9, 7, 6, 6, 3, 3, 7, 1
\]

Output of random sampling:

\[
1, -, 4, -, 3, 5, -, 7, -, 6, -, -, 7, 1 \\
-, 9, -, 5, -, 5, -, 7, -, 6, 3, -, -, -, 1
\]

Output of our comparable sampling:

\[
1, -, 4, -, 3, 5, -, -, -, 3, 3, -, 1 \\
1, -, 4, -, 3, 5, -, -, -, 3, 3, -, 1
\]

If \(x\) is a substring of \(y\), then \(x'\) (the sample of \(x\)) is a substring of \(y'\) (the sample of \(y\)).

[Shu, et al. IEEE TIFS 2016]
Transformed leak stands out in the alignment-based detection

Enron dataset (2.6GB): 150 users, 517,424 emails. 3-grams.
GPU acceleration of AlignDLD

Testing Platforms

<table>
<thead>
<tr>
<th></th>
<th># of Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td></td>
</tr>
<tr>
<td>Intel Core i5 2400, Sandy-Bridge microarchitecture</td>
<td>4</td>
</tr>
<tr>
<td><strong>GPU (single)</strong></td>
<td></td>
</tr>
<tr>
<td>Nvidia Tesla C2050, Fermi architecture</td>
<td>448</td>
</tr>
</tbody>
</table>
Hadoop (distributed hashtable) implementation of the set intersection based detection

225 Mbps peak throughput

Amazon EC2: 24 nodes each with a c3.2xlarge instance (8 CPUs and 15 GB RAM)

Local: 24 nodes each with 2 quad-core 2.8 GHz Xeon processors and 8 GB RAM

37 GB Enron Email Corpus as content

[Liu, et al. ACM CODASPY 2015]
What executives should do?
Questions?
Fuzzy fingerprints and the detection protocol

**Our Approach**

1. Preprocess and prepare fuzzy fingerprints
2. Release fingerprints
3. Monitor outbound network traffic
4. Detect
5. Report all data-leak alerts
6. Postprocess and identify true leak instances

**PPDLD**

1) Pre-process
2) Release
3) Monitor
4) Detect
5) Report
6) Post-process

[Shu, Yao, and Bertino. *IEEE TIFS ‘15*]