To Be Software Developers' Friends: Tool Development for Cryptographic Coding

Daphne Yao 姚丹凤
Professor
Virginia Tech

http://yaogroup.cs.vt.edu/
Acknowledgements

Sazzadur Rahaman  
Assistant Professor  
University of Arizona

Ya Xiao, Miles Frantz  
Sharmin Afrose, Na Meng  
Virginia Tech

Yang Zhao  
Cristina Cifuentes  
Oracle Lab Australia

Barton Miller  
UW-Madison

Inscrypt 2020
Software is everywhere

Ford GT has over 10 million lines of code

F-22 Raptor has 2 million lines of code

Boeing 787 Dreamliner has 7 million lines of code

Ford pickup truck F-150 has 150 million lines of code

Developers’ code is getting closer and closer to your body

Virginia State’s contact tracing app using Google/Apple’s exposure notification library
We need both -- developer training & using tools

Top 10 secure coding rules

1. Validate input. Validate input from all untrusted data sources.
2. Heed compiler warnings [and other warnings].
3. Architect and design for security policies.
4. Keep it simple.
5. Default deny.
6. Adhere to the principle of least privilege.
7. Sanitize data sent to other systems.
8. Practice defense in depth.
9. Use effective quality assurance techniques.
10. Adopt a secure coding standard.

https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices
Microsoft secure development lifecycle (SDL)

Developers need TOOLS and more TOOLS

Who would not want to write secure code?

- Budget
- Time
- False positives
- Resources
Deployment

GAP

theory

practice
Cross-Site Request Forgeries: Exploitation and Prevention

William Zeller* and Edward W. Felten*†
*Department of Computer Science
†Center for Information Technology Policy
†Woodrow Wilson School of Public and International Affairs
Princeton University
{wzeller,felten}@cs.princeton.edu

Revision 10/15/2008: Noted that the New York Times has fixed the vulnerability described below. Also clarified that our server-side CSRF protection recommendations do

Cross-Site Request Forgery (CSRF) attacks occur when a

[PDF] Robust Defenses for Cross-Site Request Forgery - Stanford Security Lab
by A Barth - 2008 - Cited by 456 - Related articles
Collin Jackson. Stanford ... Cross-Site Request Forgery (CSRF) is a widely exploited web site ... the header can be used today as a reliable CSRF defense.
What is cross-site request forgery (CSRF) attack?

1. Victim has a valid session with bank.com
2. Victim visited a malicious form
3. Victim tricked into submitting form
4. Browser automatically attaches session-id
5. Money transferred to attacker 🤔

From C. Jackson
Developers need help

“Adding csrf().disable() solved the issue!!! I have no idea why it was enabled by default” – a StackOverflow post

"Adding csrf().disable() solved the issue!!! I have no idea why it was enabled by default"

"adding -Dtrust_all_cert=true to VM arguments"

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

// 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.setDefaultSSLSocketFactory(sc.
    getSocketFactory());
} catch (Exception e) {"
Influencers -- 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><strong>Total Views</strong></td>
<td><strong>622,922</strong></td>
<td><strong>17</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

As of August 2017

Insecure StackOverflow posts seem to have a large influence on developers 😐


* In Java Spring Security for web applications
Cyberbully on StackOverflow developers forum

User: skanga [0]

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

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

User: MarsAtomic [6,287]

“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
The media drives a wedge between software developers and security researchers 😞
The truth is –

Developers need help to write secure crypto code
A simple vulnerability example

Constant keys defined & used in the same method (intra-procedural)

Insecure

```java
String defaultKey = "Inscrypt";
byte[] keyBytes = defaultKey.getBytes();
keyBytes = Arrays.copyOf(keyBytes, 16);
SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES");
```

Secure

```java
SecureRandom random = new SecureRandom();
String defaultKey = String.valueOf(random.ints());
byte[] keyBytes = defaultKey.getBytes();
keyBytes = Arrays.copyOf(keyBytes, 16);
SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES");
```
Need to recognize more complex vulnerability patterns

Multi-class/method data-flow

```java
public class MultipleClass1 {
    public void method1 (String passedAlgo) {
        MultipleClass2 mc = new MultipleClass2 ();
        mc.method2 (passedAlgo);
    }
}

public class MultipleClass2 {
    public void method2 (String cryptoAlgo) {
        Cipher cipher = Cipher.getInstance (cryptoAlgo);
    }
}
```

Lack of hostname verification (TLS)

```java
public class SecDevTM implements X509TrustManager {
    private X509TrustManager defaultTM;
    ...
    @Override
    public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {
        try{
            defaultTM.checkServerTrusted(chain, authType);
        } catch(CertificateException e){
            Log.w("checkServerTrusted", e.toString());
        }
    }
}
```

Need to throw an exception
Detection approach – Mapping **crypto properties** → **program analysis**

**Orthogonal exploration in getProperty method**

```
c = getProperty();
```

```
e = e' (in Class myObj)
```

```
a = 1024;
```

```
b = “blah”;
```

```
initialize(a, b, c, myObj.e,...) (Slicing Criteria)
```

**Forward slicing** (refinement for FP reduction)

Support flow-, Context-, Field-sensitive analysis
# Crypto in Android App Libraries (on 6,181 apps)

96% of issues coming from libraries

<table>
<thead>
<tr>
<th>Rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Predictable pwds for PBE</td>
</tr>
<tr>
<td>3</td>
<td>Predictable pwds for keystores</td>
</tr>
<tr>
<td>4</td>
<td>Dummy hostname verifier</td>
</tr>
<tr>
<td>5</td>
<td>Dummy cert. verifier</td>
</tr>
<tr>
<td>7</td>
<td>Use of HTTP</td>
</tr>
<tr>
<td>9</td>
<td>Weak PRNG</td>
</tr>
<tr>
<td>12</td>
<td>Static IV</td>
</tr>
<tr>
<td>16</td>
<td>Broken hash</td>
</tr>
</tbody>
</table>

[Sazzadur et al. ACM CCS 2019]
Crypto Code in Java Can Be Complex to Analyze

```
1 class PasswordEncryptor {
2     Crypto crypto;
3 }
4
5 public PasswordEncryptor(){
6     String passKey = PasswordEncryptor . getKey("pass.key");
7     crypto = new Crypto(passKey);
8 }
9
10 byte[] encPass(String [] arg){
11     return crypto.encrypt(arg[0], arg[1]);
12 }
13
14 static String getKey(String src){
15     String key = Context.getProperty(src);
16     if (key == null) {
17         key = "defaultkey";
18     }
19     return key;
20 }
```

[Sazzadur et al. ACM CCS 2019]
Too Many Security Irrelevant Constants

False positives (i.e., false alarms) 😞
False alarms are counter-productive

Irrelevant vs. irrelevant constants

“UTF-8”: irrelevant (for encoding)

“pass.key”: irrelevant (a file name)

“defaultkey”: what we need (a hardcoded pwd)
Reduction of FPs by Refinements -- Off-the-shelf Program Slicing Would Fail

Reduce false alerts by 76% to 80%

Reduction results of FPs with refinement insights in 46 Apache projects

[Sazzadur et al. ACM CCS 2019]
Deployment-quality Accuracy and Scalability

Max, min, & avg LoC:  
2,571K (Hadoop), 1.1K (Commons Crypto), & 402K, respectively
Detected insecure PBE code from Apache Ranger (They fixed it)

```java
PBEKeySpec getPBEParameterSpec(String password) throws Throwable {
    MessageDigest md = MessageDigest.getInstance(MD_ALGO);
    byte[] saltGen = md.digest(password.getBytes());
    byte[] salt = new byte[SALT_SIZE];
    System.arraycopy(saltGen, 0, salt, 0, SALT_SIZE);
    int iteration = password.toCharArray().length + 1;
    return new PBEKeySpec(password.toCharArray(), salt, iteration);
}
```

- Defined earlier as MD5, no good
- Salt should not depend on pwd
- Should be 1000 iterations
- Side-channel leak, as iteration/runtime reveals the pwd length

Password-based encryption (PBE)
What Exactly is Deployable Accuracy?

98.6% Precision

Out of 1,295 Apache alerts, 18 are false alarms

[Sazzadur et al. ACM CCS 2019]
Crypto Guard

Crypto API Benchmarks -- driving up the industry standards

- SpotBugs
  - 171 man-made test units
  - 40 basic cases
  - 131 advanced cases
  - 16 crypto rules

- CogniCrypt

Benchmark based on Apache software to come!

https://github.com/CryptoGuardOSS/cryptoapi-bench
What does industrial strength code scanner look like?
Oracle’s Parfait – an **industrial strength** static analysis tool for software security (started in 2007)

Parfait is fast -- analyzing 10.6 million of lines of code in 80 mins on a 2.9GHz AMD computer

Parfait is precise -- average false positive rate < 10%

Cristina Cifuentes and her team

Oracle Lab Australia implemented CryptoGuard’s approach (2019) to scan production code

Results of Parfait’s crypto scanning 11 internal Oracle projects (Java) -- detection approach based on CryptoGuard

- Scanned 11 projects; reported 42 vulnerabilities with 0 false positive (100% precision)
- Average runtime 338.8s for 11 projects with average 395.4k LoC
Parfait’s benchmark evaluation (on CryptoAPI-Bench)

**98.4% Recall**

**86.6% Precision -- 100% precision if excluding path sensitive cases**

How many actual vulnerabilities are reported? Higher the better 😊

How many reported alerts are real vulnerabilities? Higher the better 😊

<table>
<thead>
<tr>
<th>Type</th>
<th>Total Cases</th>
<th>Insecure Cases</th>
<th>Secure Cases</th>
<th>Reported Cases</th>
<th>False Positives</th>
<th>False Negatives</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Cases</td>
<td>27</td>
<td>24</td>
<td>3</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Multiple methods</td>
<td>57</td>
<td>56</td>
<td>1</td>
<td>54</td>
<td>0</td>
<td>2</td>
<td>100%</td>
<td>96.43%</td>
</tr>
<tr>
<td>Multiple Classes</td>
<td>23</td>
<td>18</td>
<td>5</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Field Sensitivity</td>
<td>19</td>
<td>18</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Path Sensitivity</td>
<td>19</td>
<td>0</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Heuristics</td>
<td>13</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Total</td>
<td><strong>158</strong></td>
<td><strong>125</strong></td>
<td><strong>33</strong></td>
<td><strong>142</strong></td>
<td><strong>19</strong></td>
<td><strong>2</strong></td>
<td><strong>86.62%</strong></td>
<td><strong>98.40%</strong></td>
</tr>
</tbody>
</table>
Ongoing work in my group on crypto API recommendation with deep learning

Ya Xiao (4-th year PhD student)

Bimal Viswanath (Virginia Tech)

Xinyang Ge (Microsoft Research)
Which API to use in Line 6?

```java
public byte[] encryptGCM(byte[] plaintext, byte[] keyBytes,
    byte[] iv) {
    SecretKey key = new SecretKeySpec(keyBytes, "AES");
    IvParameterSpec ivSpec = new IvParameterSpec(iv);
    Cipher cipher = createCipher("AES/GCM/NoPadding", "BC",
      Cipher.ENCRYPT_MODE, key, ivSpec);
    ByteArrayOutputStream byteArrayOutputStream = new
      ByteArrayOutputStream();
    //...
    CipherOutputStream cipherOutputStream = new
      CipherOutputStream(byteArrayOutputStream, cipher);
    byteArrayOutputStream.write(...)
    Cipher.doFinal(...)
    Cipher.updateAAD(...)
}
```
Need more research addressing practical deployment challenges

IACR

Real World Crypto Symposium

Real World Crypto Symposium aims to bring together cryptography researchers with developers implementing cryptography in real-world systems. The conference goal is to strengthen the dialogue between these two communities. Topics covered focus on uses of cryptography in real-world environments such as the Internet, the cloud, and embedded devices.

Hard Topic Theme: Deployable and Impactful Security

Background

Since 2013, ACSAC has had a hard topic theme that focuses the conference on tackling a hard, cutting-edge, cybersecurity problem requiring cooperation from government, industry, and academia. This year, ACSAC especially encourages contributions in the area of Deployable and Impactful Security.
Check out our recent secure coding tutorial (IEEE SecDev 2020)

(In)secure crypto coding examples

Secure TLS coding strategies

CryptoGuard intro/demo

Tool eval benchmark

Slides: http://yaogroup.cs.vt.edu/videos.html
Video: https://youtu.be/Z0RwBLURp9c
Identifying cryptographic vulnerabilities in today’s million-line programs has become a critical endeavor. Because of the increasing sophistication of cybercriminals, programmers can no longer afford to test for vulnerabilities using only traditional debugging techniques, followed by releasing software, collecting bug reports and patching.

The new frontier being pursued by government, industry, and academia are automated tools that are capable of culling vulnerabilities before releasing source code into the wild. When run on existing software, such as the open-source Apache programs managing the world’s servers, these tools also are finding a surprising number of vulnerabilities in software that is decades old.

Most open-source automated vulnerability checkers are still finding their way, but a team of researchers at the Virginia Polytechnic Institute and State University (Virginia Tech) claim to have vulnerability-checking software that is mature, and approaching deployment. Called CryptoGuard, the software automatically identifies cryptographic vulnerabilities in Java (and soon Python) source code. Funded by the U.S. Navy’s Office of Naval Research (ONR) and the National Science Foundation (NSF), CryptoGuard is...
Related references

**Papers:**

**Online Resources:**
- CryptoGuard. https://github.com/CryptoGuardOSS/cryptoguard
- CryptoAPI-Bench. https://github.com/CryptoGuardOSS/cryptoapi-bench
- Secure TLS/SSL code examples. https://github.com/AthenaXiao/SecureTLSCodeExample
- https://mybinder.org/v2/gh/franceme/cryptoguard/2020_SecDev_Tutorial
Questions and Comments?