Finding Needles in Haystacks

Louis Nyffenegger
<louis@pentesterlab.com>

PentesterLab

Agenda

- Introduction
- Code Review
- Capture-The-Flag
- Answers

About me

- Security Engineer:
  - Huge fan of Grey Box testing
- PentesterLab:
  - Platform to learn web security/penetration testing
  - 100% Hands-on
  - Available for individuals (free and PRO) and enterprises

This talk

- It was really hard to find the balance between giving too many hints and not giving any
- After this talk, I would like **you** to think that **you** can do code reviews!
  - Not because I'm going to teach you everything
  - Just because it's easy

Code Review

Why would you want to do code review?

- Can be faster than penetration testing
- Compliance (PCI 6.3.2)
- You’re tired of penetration testing
- You want to find better bugs
- You want to check if some code is back-doored
- You want to write an exploit for a bug

Multiple ways to go about code review

- Grep for bugs
- Follow user input
- Read some random code
- Read all the code
- Check one functionality at a time (login, password reset…)

Grep for Bugs

- Just search for
  - potential issues
    - Dangerous functions
- grep and find are your friends:

```
$ grep -R ‘system\(\$_’ *

$ find . -name \*.php -exec grep -Hn 'exec' {} \;
```

Grep for Bugs

- Pros:
  - Super fast
  - Good way to find low hanging fruits
- Cons:
  - You end up using very complex regular expressions
  - You need to know all the dangerous functions
  - Very low coverage

Follow user inputs

- Follow all the user-controlled inputs
  - Find all the routes/URI available
  - Find all the way to provide data to the application (example in PHP):
    - $_POST / $_GET / $_REQUEST
    - $_COOKIE / $_SERVER
    - Data coming from the database
    - Data read from a file or from a cache
    - …

Follow user inputs

- Pros:
  - Good coverage
- Cons:
  - Need a good understanding of the framework/language
  - You find yourself reading the same code again and again

By functionality

- Pick one functionality:
  - “Password reset”
  - “Database access”
  - “Authentication”
- And review all the code associated with it

By functionality

- Pros:
  - Excellent coverage for the functionalities reviewed
  - Improve your pentest-FU
  - Work especially well across multiple applications
- Cons:
  - No coverage for the functionalities that didn’t get reviewed

PentesterLab

Read everything

- Just start reading the code one file at the time
- Don’t try to find vulnerabilities, try to find weaknesses
- Keep notes

Read everything

- Pros:
  - Excellent coverage
  - Improve your pentest-FU
    - Work especially well across multiple applications
- Cons:
  - Hard to keep track of everything
  - Very time consuming

Ok, but what to look for?

- Everything!
- If you don’t know a function/class/method:
  - Google it
  - Test its’ behaviour
- It’s going to take time (especially at the beginning):
  - The more code you review, the easier it gets

Test a function’s behaviour

- Just copy the snippet you are reviewing
- Run it (Locally/Docker/VM) https://medium.com/@PentesterLab/use-docker-for-your-pentesting-labs-879fe9feeca8
- Try to find some edge cases
- Save this example for the next time you see this function
- For PHP, you can use ‘php -a’

PentesterLab

Test a function's behaviour

```
🍺 koriander ~/code/java % ls
OneTimePassword.class   TestFile.java        TestRemove.class
OneTimePassword.java    TestHashMap.class    TestRemove.java
ScriptEngine            TestHashMap.java     TestToken.class
Session.class           TestJSF.class        TestToken.java
Session.java            TestJSF.java         TestTrustFact.class
Test.class              TestLong.class       TestTrustFact.java
Test.java               TestLong.java        TestURL.class
TestAnnotation.java     TestMatch.class      TestURL.java
TestBigInt.class        TestMatch.java       TestUUID.class
TestBigInt.java         TestMatch2.class     TestUUID.java
TestCanonical.class     TestMatch2.java      int_no_sqli
TestCanonical.java      TestPattern.class    processbuilder
TestEquals.class        TestPattern.java     ssl
TestEquals.java         TestRandom.class
TestFile.class          TestRandom.java
```

PentesterLab

Ok, but what to look for?

- Weird behaviour
- Differences between 2 functions/methods/classes
- Security checks already in place
- Comparison and conditions (if/else)
- Complexity
- Regular expressions/string matching
- What is missing?

Capture the Flag

Common Misconceptions

- “I need to be an expert in language X”
- “It takes a lot of time”
- “It’s too hard”
- “It’s boring”

Reality

- It’s actually easy (kind of)
- It requires DEEP WORK (See book from Cal Newport)
- It’s a good way to find interesting bugs
- It’s probably the best way to become a better security engineer
- Like for pen testing… **practice makes perfect!**

How to get started

- Read everything
- Don’t try to find vulnerability
    - Try to find weaknesses to start:
        - “That doesn’t seem right”
        - “What happens if I put [X] here”
- Try to see if the weaknesses can become vulnerabilities
    - On their own
    - By combining them

The application

- Get the code:
  - git clone https://github.com/PentesterLab/sectalks
  - https://github.com/PentesterLab/sectalks/archive/master.zip
- Very simple application with ~ 10 security issues:
  - Register/Login/Logout
  - Upload and retrieve file

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
CSS                              2            676             11           3973
PHP                             11             48              5            287
SQL                              1              5              0              5
-------------------------------------------------------------------------------
SUM:                            14            729             16           4265
-------------------------------------------------------------------------------
```

Answers

List of weaknesses

- Hardcoded credentials
- Information leak
- Weak password hashing mechanism
- Cross-Site Scripting
- No CSRF protection

List of weaknesses

- Crypto issue
- Signature bypass
- Authentication bypass
- Authorisation bypass
- Remote Code Execution
- …

Hardcoded Credentials

```php
<?php
    $lnk = mysql_connect("127.0.0.1", "pentesterlab", "pentesterlab");
    $db = mysql_select_db('cr', $lnk);
?>

public static function signature($data) {
  return hash("sha256","donth4ckmebr0".$data);
}
```

Information leak

```
Hardac% ls -a
.               README          deploy.sql      header.php      logout.php
..              classes         files           index.php       register.php
.git            css             footer.php      login.php
```

PentesterLab

Weak password hashing mechanism

```php
public static function register($user, $password) {
  $sql = "INSERT INTO  users (login,password) values (\"";
  $sql.= mysql_real_escape_string($user);
  $sql.= "\", md5(\"";
  $sql.= mysql_real_escape_string($password);
  $sql.= "\"))";
  $result = mysql_query($sql);
  if ($result) {
    return TRUE;
```

Should not use md5… scrypt or bcrypt or PBKDF2

PentesterLab

Cross Site Scripting

- **Weaknesses:**

```
Hardac% grep -R 'echo $error' *
index.php:          <span class="text text-danger"><b><?php echo $error; ?></b></span>
login.php:          <span class="text text-danger"><b><?php echo $error; ?></b></span>
register.php:          <span class="text text-danger"><b><?php echo $error; ?></b></span>
```

- **Exploitable:**

```
<h3>Upload (only PDF)</h3>
  <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post" enctype="multipart/form-data">
        <input type="file" name="file" id="file">
        <input type="submit" value="Upload your PDF" name="submit">
  </form>
```

Crypto issue

```php
public static function signature($data) {
  return hash("sha256","donth4ckmebr0".$data);
}
```

- JWT use HMAC not just a hash
- Weakness: Length extension (https://en.wikipedia.org/wiki/Length_extension_attack)

Authentication issue: Signature bypass

```php
public static function verify($auth) {
  list($h64,$d64,$sign) = explode(".",$auth);
  if (!empty($sign) and (JWT::signature($h64.".".$d64) != $sign)) {
    die("Invalid Signature");
  }
  $header = base64_decode($h64);
  $data = base64_decode($d64);
  return JWT::parse_json($data);
}
```

Authentication issue

- The application doesn't parse JSON properly;

```
public static function parse_json($str) {
  $data = explode(",",rtrim(ltrim($str, '{'), '}'));
  $ret = array();
  foreach($data as $entry) {
    list($key, $value) =  explode(":",$entry);
    $key = rtrim(ltrim($key, '"'), '"');
    $value = rtrim(ltrim($value, '"'), '"');
    $ret[$key] = $value;
  }
  return $ret;
 }
}
```

- Inject some JSON in your username to become another user:

plop”,"username":"admin

PentesterLab

Authorisation issue

```php
public static function addfile($user) {
  $file = "files/".$user."/".basename($_FILES["file"]["name"]);
  if (!preg_match("/\.pdf/", $file)) {
    return  "Only PDF are allowed";
  } elseif (!move_uploaded_file($_FILES["file"]["tmp_name"], $file)) {
    return "Sorry, there was an error uploading your file.";
  }
  return NULL;
}
```

- Files are stored in the web root without any proper access control:

  you just need to know a filename and username to access a file.

Remote Code Execution

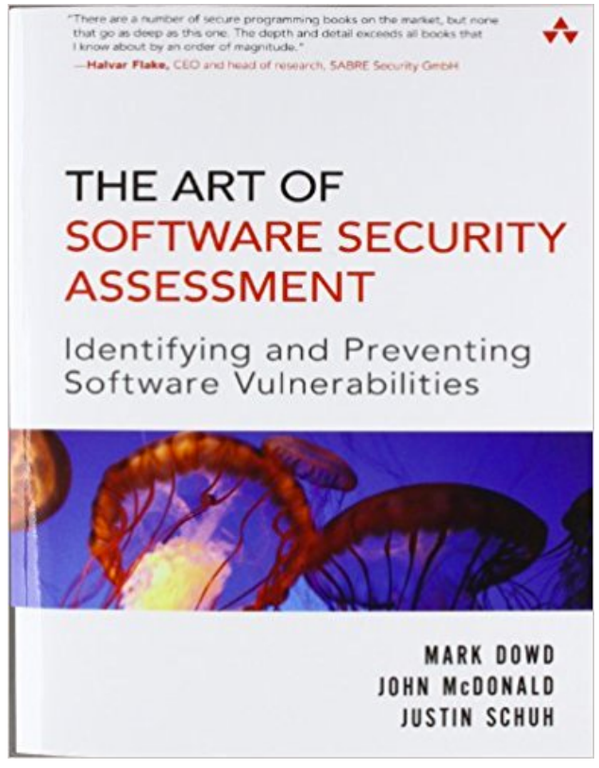
```
if (!preg_match("/\.pdf/", $file)) {
  return  "Only PDF are allowed";
} elseif (!move_uploaded_file($_FILES["file"]["tmp_name"], $file)) {
```

- You can create a file named ‘blah.pdf.php’ and it will be stored in the web root == Code execution

Conclusion

What now?

- Keep practicing
- The more code you read, the easier it gets:
    - Read advisories and diff the changes
    - Read the code of the tools you use
- Keep a list of projects you want to look at
- Compare frameworks/libraries
- Read the TAOSSA



PentesterLab

Thanks for your time!

Any questions?