Extreme Penetration Testing

Yaniv Simsolo, CISSP
CTO, Palantir Security

OWASP Israel

Israel Chapter Meeting April 2016

Palantir security

About Me

Yaniv Simsolo
CISSP
CTO, Palantir Security

- Over 12 years in Information Security
- Information Systems security expert of:
  - Compound and Hybrid systems
  - Cloud solutions
  - Applications
  - Database Security
  - Back-End systems and components
  - Middleware

Agenda

- Intro
- PT – normal
- PT – Advanced
- PT - Extreme
- Summary

Extreme Penetration Testing

Intro

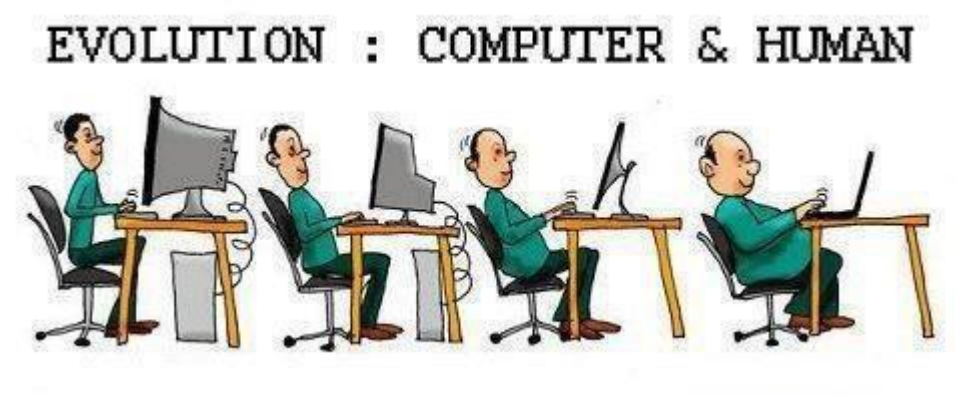
Palantir security

Be Cool

- Hacking is cool
- Hacking highly secure systems is COOOOOLER

Be Cool

- Take a system.
  - The customer will supply you with at least one!
- Break it to kingdom come.
- Enjoy yourself, have fun.
- Learn something new.
- Experience.
- Exercise your greatest muscle!



Penetration Testing

- The art of…
  - Hacking?
  - Reviewing?
  - Using tools & technologies?
  - Observing?
  - Analyzing?
- Penetration Testing is cool!

Extreme Penetration Testing

PT - Normal

Palantir security

Normal PT

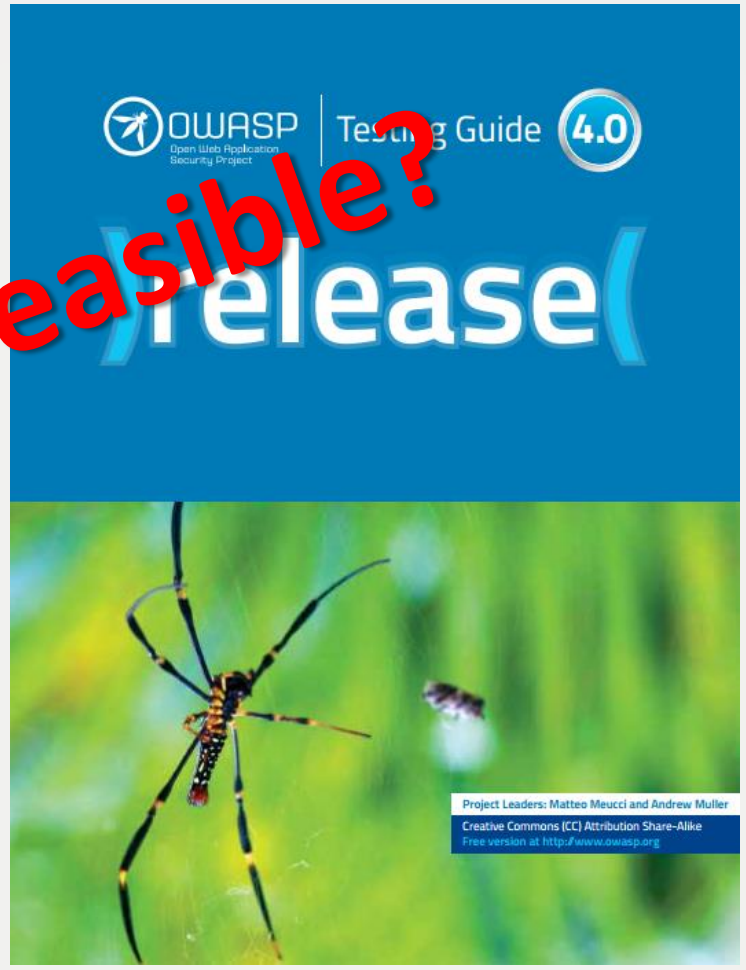
- Get some access to a system
- Connect
- Get a feel for the system
- Attack!
- Use some tools. Attack!
- Analyze. Revise and recuperate. Attack even harder!
- Get the Bonanza! Attack, attack, attack until you get it!
- Attack some more!
- Report.

Is that the correct process?

PT Process

- Get the OWASP Testing Guide
- It is the Best Practice
- Follow it
  - Meticulously
  - All 224 pages

Is that process feasible?



What is the described test?

browser environments simply displaying an email message containing the image would result in the execution of the request to the web application with the associated browser cookie.

Things may be obfuscated further, by referencing seemingly valid image URLs such as

```
<img src="https://[attacker]/picture.gif" width="0"
height="0">
```

where [attacker] is a site controlled by the attacker, and by utilizing a redirect mechanism on

```
http://[attacker]/picture.gif to http://[thirdparty]/action.
```

Cookies are not the only example involved in this kind of vulnerability. Web applications whose session information is entirely supplied by the browser are vulnerable too. This includes applications relying on HTTP authentication mechanisms alone, since the authentication information is known by the browser and is sent automatically upon each request. This DOES NOT include form-based authentication, which occurs just once and generates some form of session-related information (of course, in this case, such information is expressed simply as a cookie and can we fall back to one of the previous cases).

Sample scenario

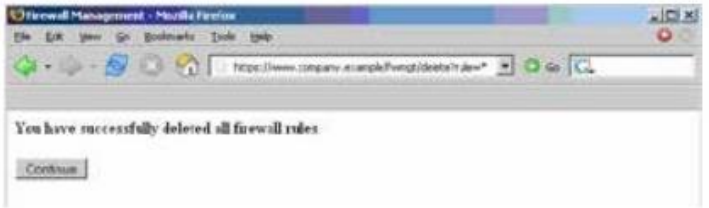
Let's suppose that the victim is logged on to a firewall web management application. To log in, a user has to authenticate himself and session information is stored in a cookie.

Let's suppose the firewall web management application has a function that allows an authenticated user to delete a rule specified by its positional number, or all the rules of the configuration if the user enters '*' (quite a dangerous feature, but it will make the example more interesting). The delete page is shown next. Let's suppose that the form – for the sake of simplicity – issues a GET

Therefore, if we enter the value '*' and press the Delete button, the following GET request is submitted.

```
https://www.company.example/fwmgt/delete?rule=*
```

with the effect of deleting all firewall rules (and ending up in a possibly inconvenient situation).



Now, this is not the only possible scenario. The user might have accomplished the same results by manually submitting the URL or by following a link pointing, directly or via a redirection, to the above URL. Or, again, by accessing an HTML page with an embedded img tag pointing to the same URL.

```
https://[target]/fwmgt/delete?rule=*
```

In all of these cases, if the user is currently logged in the firewall management application, the request will succeed and will modify the configuration of the firewall. One can imagine attacks targeting sensitive applications and making automatic auction bids, money transfers, orders, changing the configuration of critical software components, etc.

An interesting thing is that these vulnerabilities may be exercised behind a firewall; i.e., it is sufficient that the link being attacked be reachable by the victim (not directly by the attacker). In par-

Normal PT

- At least – test for OWASP Top 10.
- How do you test A5?

Extreme Penetration Testing

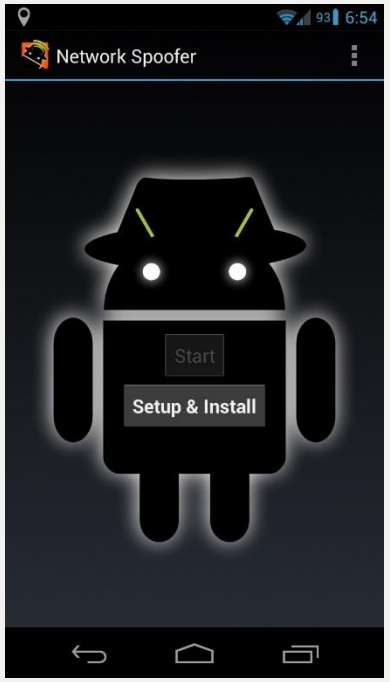
PT - Advanced

Palantir security

Advanced Penetration Testing

- Following some Best Practices is:
  - Not always feasible
  - Not always practical
  - Resource consuming
- So, use some shortcuts:
  - Scan tools
  - Hack tools

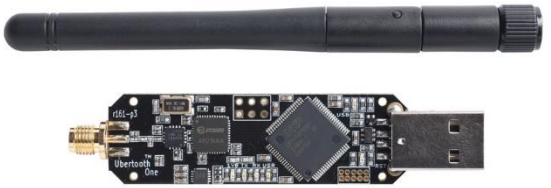
That Which is Common



That Which is Common

- Tools progress with technology

That Which is Common



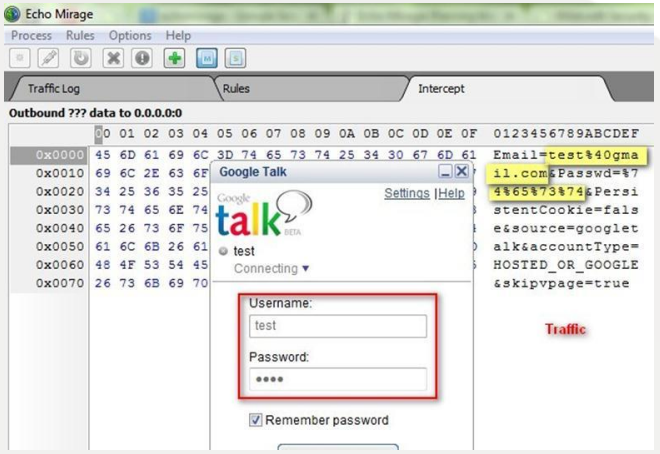

```
bt ~ # hcidump -i hci0 -X -t
HCI sniffer - Bluetooth packet analyzer ver 1.37
device: hci0 snap_len: 1028 filter: 0xffffffff
1240692321.957111 < ACL data: handle 12 flags 0x02 dlen 52
    L2CAP(s): Echo req: dlen 44
    0000: 41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  ABCDEFGHIJKLMNOP
    0010: 51 52 53 54 55 56 57 58  59 5a 5b 5c 5d 5e 5f 60  QRSTUVWXYZ[\]^_`
    0020: 61 62 63 64 65 66 67 68  41 42 43 44              abcdefghABCD
1240692321.991241 > ACL data: handle 12 flags 0x02 dlen 52
    L2CAP(s): Echo rsp: dlen 44
    0000: 41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  ABCDEFGHIJKLMNOP
    0010: 51 52 53 54 55 56 57 58  59 5a 5b 5c 5d 5e 5f 60  QRSTUVWXYZ[\]^_`
    0020: 61 62 63 64 65 66 67 68  41 42 43 44              abcdefghABCD
1240692322.184240 > HCI Event: Number of Completed Packets (0x13) plen 5
  0000: 01 0c 00 01 00                                     .....
1240692322.993200 < ACL data: handle 12 flags 0x02 dlen 52
    L2CAP(s): Echo req: dlen 44
    0000: 41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  ABCDEFGHIJKLMNOP
    0010: 51 52 53 54 55 56 57 58  59 5a 5b 5c 5d 5e 5f 60  QRSTUVWXYZ[\]^_`
    0020: 61 62 63 64 65 66 67 68  41 42 43 44              abcdefghABCD
1240692323.012326 > ACL data: handle 12 flags 0x02 dlen 52
    L2CAP(s): Echo rsp: dlen 44
    0000: 41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  ABCDEFGHIJKLMNOP
    0010: 51 52 53 54 55 56 57 58  59 5a 5b 5c 5d 5e 5f 60  QRSTUVWXYZ[\]^_`
    0020: 61 62 63 64 65 66 67 68  41 42 43 44              abcdefghABCD
1240692323.185332 > HCI Event: Number of Completed Packets (0x13) plen 5
  0000: 01 0c 00 01 00                                     .....
1240692324.013175 < ACL data: handle 12 flags 0x02 dlen 52
    L2CAP(s): Echo req: dlen 44
    0000: 41 42 43 44 45 46 47 48  49 4a 4b 4c 4d 4e 4f 50  ABCDEFGHIJKLMNOP
    0010: 51 52 53 54 55 56 57 58  59 5a 5b 5c 5d 5e 5f 60  QRSTUVWXYZ[\]^_`
    0020: 61 62 63 64 65 66 67 68  41 42 43 44              abcdefghABCD
```

That Which is Common

Palantir security

Fiddler

soapUI



SQL

APK inspector

Cold Hard Facts

Misconceptions:

- We CAN test THE security
- Our tools CAN scan the ENTIRE scope
- Our protection tools CAN PROTECT us

Cold Hard Facts

Palantir security

InfoSec Natural Selection, Shay Chen, OWASP IL 09_2014

WAVSEP
Web Application Vulnerability Scanner Evaluation Project

The tools are not complete

#	Logo	Vulnerability Scanner		COUNT	GET	POST	COOKIE	HEADER	SECRET	PName	XML	XmlATT	XmlTAG	JSON	.NetENC	AMF	JavaSER	.NetSER	WCF	WCF-B	WebSock	DWR	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]XML	[illegible]JSON	[illegible]PName	Multipart	GWT	ODataID	ODataFilter
1		Burp Suite Professional		19	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
2		IBM AppScan		17	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗
3		NTOSpider		16	✓	[illegible]	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗
4		WebInspect		1[illegible]	[illegible]	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
5		Netsparke		9	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
6		ScanToSecure		9	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
7		Acunetix WVS		7	✓	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
8		Ammonite		7	✓	✓	✓	✓	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
9		Syhunt Dynamic		7	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
10		N-Stalker		6	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗

Advanced Penetration Testing

- Code analysis
  - Will not cover newest technologies
  - Limited strength
  - Static Vs. Dynamic

Advanced Penetration Testing

- Memory analysis, disk analysis
  - Resource intensive
  - Pinpointed approach only

Software Security Analysis without Code Schematics

Software Security Analysis without a control and data flow diagram of logic and design, is like home security analysis without schematics, such as a flooring plan or circuitry diagram.

Simply scanning for known exploits without verifying control flow integrity is comparable to the same security expert explaining the obvious, such as windows are open and doors are unlocked, and being completely oblivious to the fact that there is a trap door in your basement.

Those known exploits, just like the insecure doors and windows, are only the low hanging fruit.

- McCabe Software

(CVE-2010-0188 Adobe PDF)

A 256 AES strong key implementation

- 16*8 = 256?

```
    /**
    * this function generates random string for given length
    * @param length
    *               Desired length
    * * @return
    */
    public static String generateRandomIV(int length)
    {
        SecureRandom ranGen = new SecureRandom();
        byte[] aesKey = new byte[16];
        ranGen.nextBytes(aesKey);
        StringBuffer result = new StringBuffer();
        for (byte b : aesKey) {
            result.append(String.format("%02x", b)); //convert to hex
        }
        if(length> result.toString().length())
        {
            return result.toString();
        }
        else
        {
            return result.toString().substring(0, length);
        }
    }
}
```

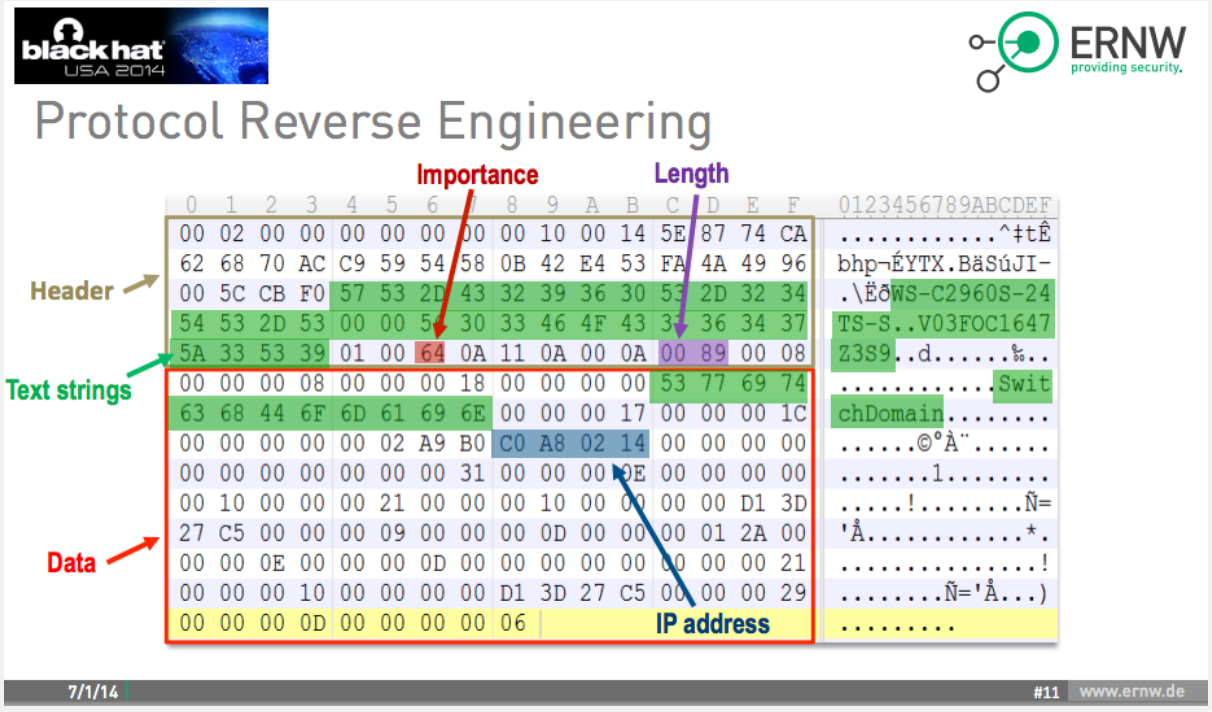
Yes. It can get worse.

```
import com.      .im.utils.  Log;
import com.      .im.utils.ValidationUtil;
/**
 * This is a singleton class which handles all the encryption-decryption along with key-randomiv generation.
 * @author
 *
 */
public class AESEncryptionUtil {

    private final String TAG = AESEncryptionUtil.class.getSimpleName();

    private static AESEncryptionUtil mInstance = new AESEncryptionUtil();

//  private HashMap<String, String[]> mKeyMapping = new HashMap<String, String[]>();

    private final String TAG_GENERATE_KEY = "       key";

 ////////////// removed code for clear      image /////////////////

  }
  */
  /**
   * Handles the generation and caching of key and randomIV.
   * @return
   */
  public String[] getEncryptionKeyRandomIV(){
      BzLog.dumpLog(Log.INFO, TAG, "getEncryptionKeyRandomIV called.");
      String[] keyRandom = new String[2];
      try {
          keyRandom[0] = CryptLib.SHA256(TAG_GENERATE_KEY, 32);
          keyRandom[1] = CryptLib.generateRandomIV(16);
      } catch (NoSuchAlgorithmException e) {
          // TODO Auto-generated catch block
          BzLog.dumpLog(Log.INFO, TAG, e.getMessage());
          return null;
      } catch (UnsupportedEncodingException e) {
          BzLog.dumpLog(Log.INFO, TAG, e.getMessage());
          return null;
      }

      return keyRandom;
  }
```

The company name

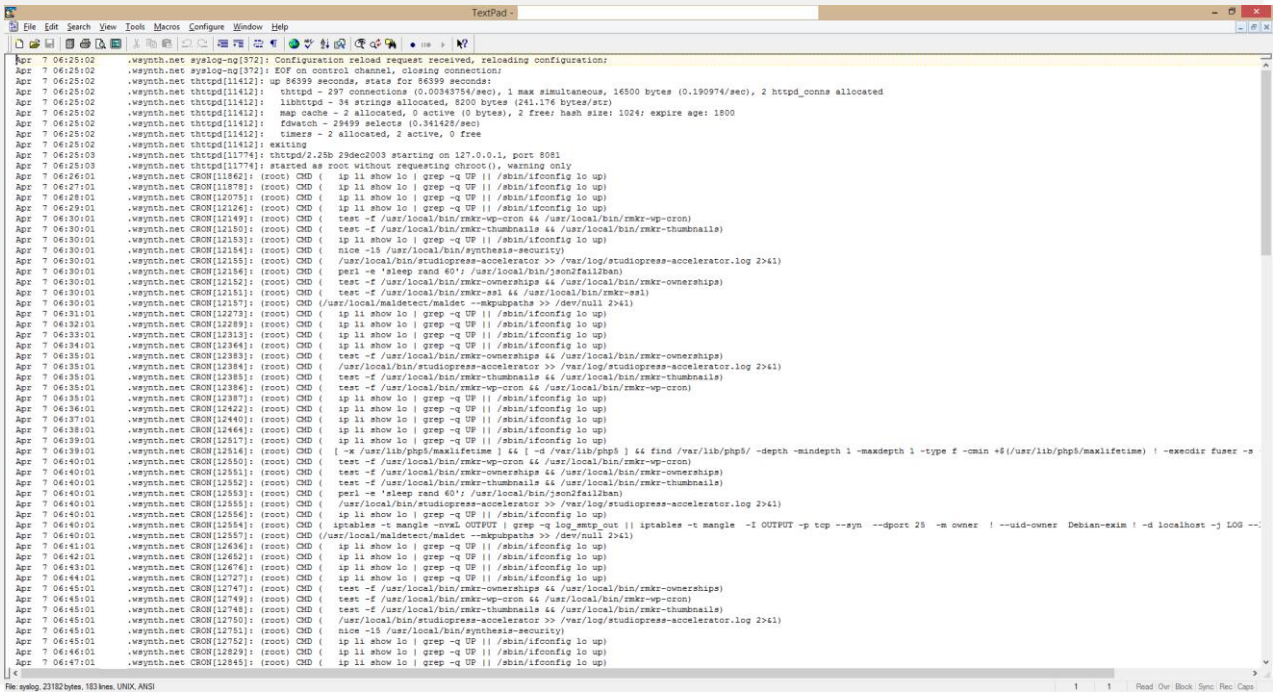
More Advanced Methods

- Reversing
  - Resource intensive
  - Highly advanced inter-disciplinary skills mandatory



More Advanced Methods

- Debugging
  - Can that be considered a PT technique?
  - Limited



Extreme Penetration Testing

Extreme
Penetration Testing

Palantir
security

The Problem

A system must be penetration tested

- Imagine a system or a component that has no GUI whatsoever.
- This is no obstacle for professionals, since proven techniques like Reversing, Memory Analysis, Code Review and Debug will be implemented.

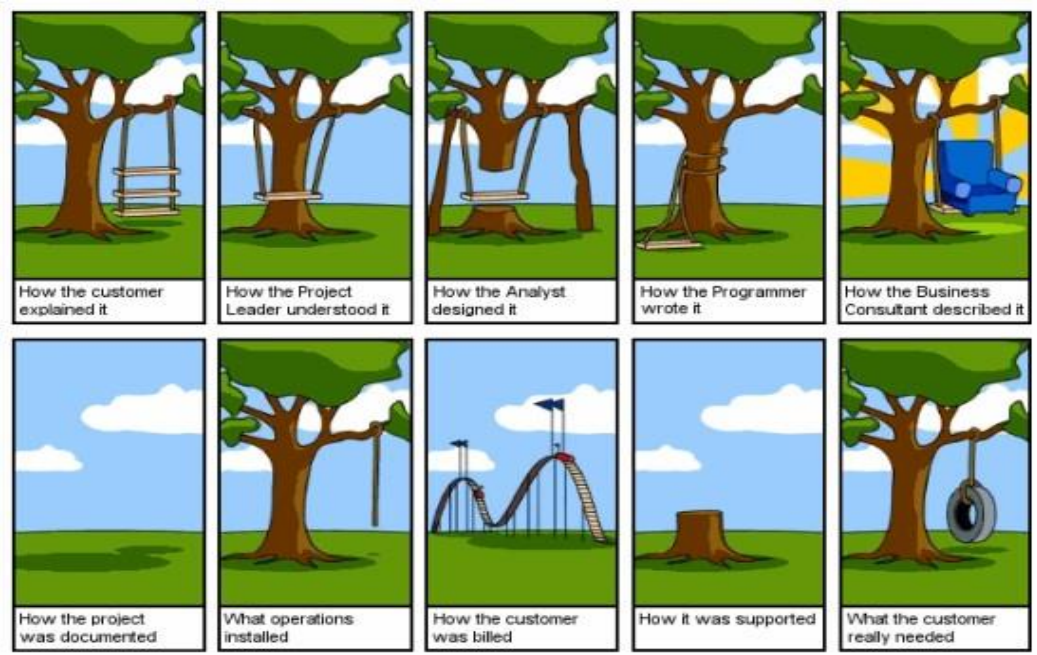
The Problem

- About The system:
  - Passed a full Security Design Review
  - A high complexity, multi layer solution
  - Employs Best Cryptography Practices
  - Employs Best Coding Practices
  - Internal code process, no “normal” interfaces

The Problem

- So, what's left?
  - Code analysis tests – completed, bugs fixed.
  - Not yet PT'ed
- Is PT even required?
  - ???

If all previously identified bugs are fixed?



Penetration Testing

- Some penetration tests are not about coolness
- Modus operandi:
  - Do not work hard unless absolutely necessary

  - When necessary: work the HARDEST
  - Be very fast! ALWAYS!

PT it is! But…

- No “normal” interfaces
  - No attack entry points
  - Hack tools irrelevant
  - Scan tools irrelevant

PT it is! But…

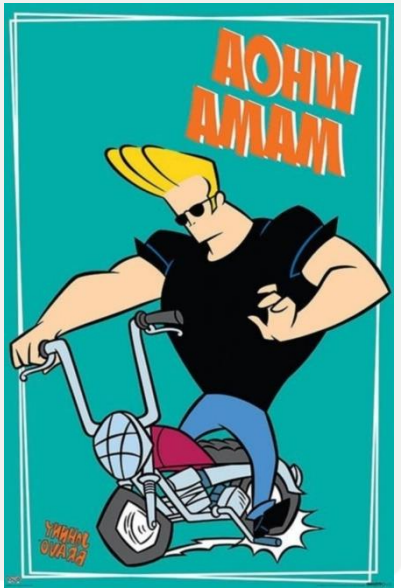
OK.

The challenge is on!

- Bring in the heavy artillery
  - Memory analysis – not relevant due to cryptography multi layers
  - Disk analysis – not relevant due to no save to disk functionality

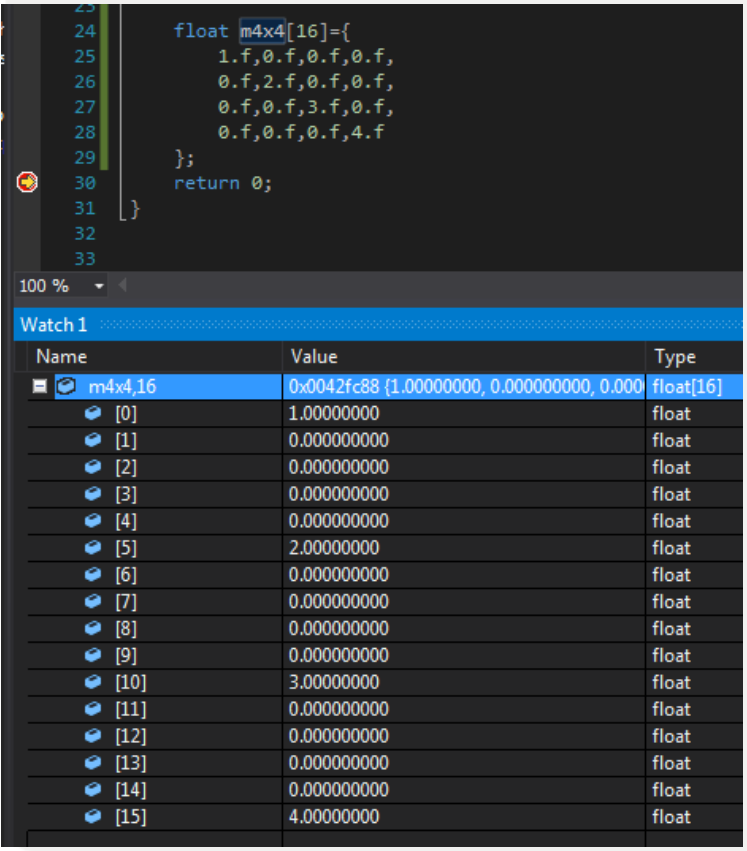
PT it is! But…

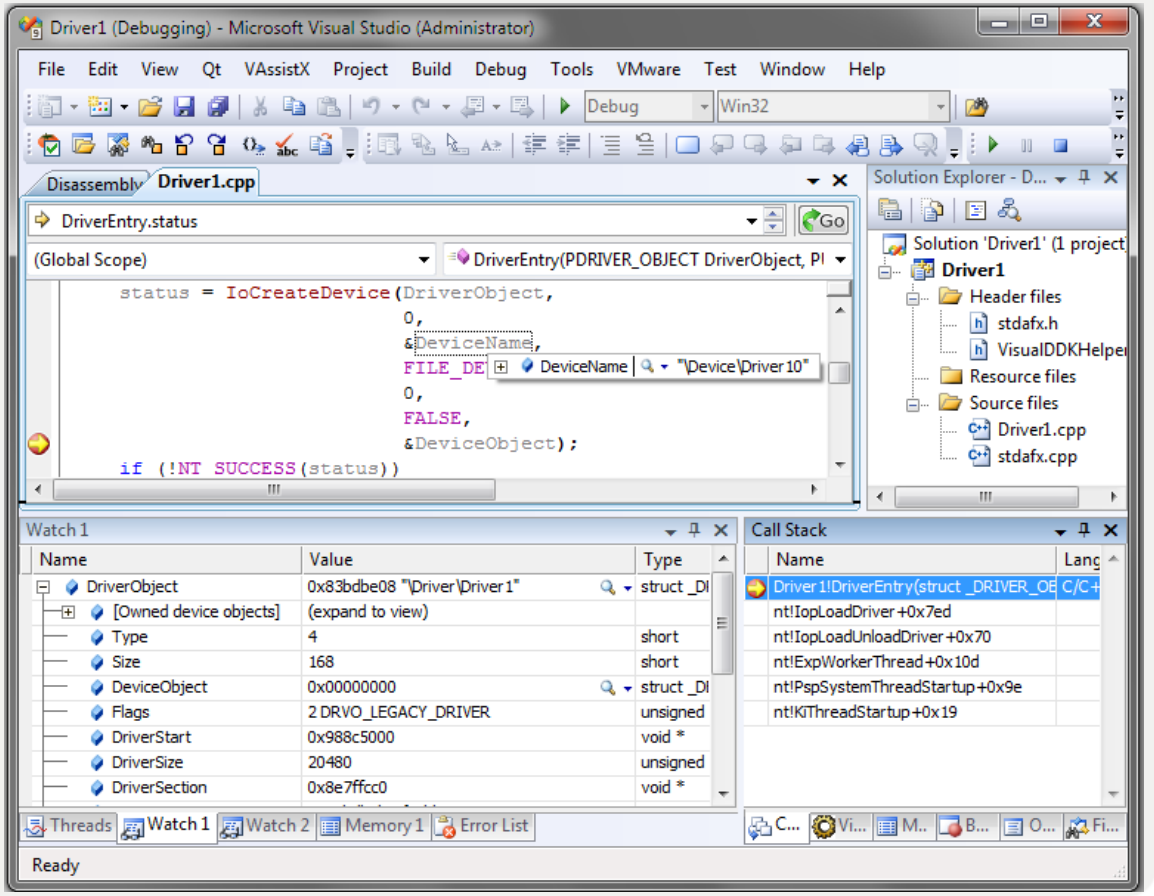
- So we are left with reversing and debugging.
- Reversing is viable but the scope of testing is wide.
- Debugging is the only acceptable PT approach.
- Eureka!



PT it is! But…

- Ever tried an entire PT from the development environment Debugging?





PT it is! But…

- If I know where to look for:
  - Easy to use Debugging or runtime manipulation
- What if I don't know where to look?

One More Joker Up My Sleeve

- Unit Testing
  - unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use

PT it is! But…

- Unit Testing practicality
  - Whatever unit tests were created – is now legacy to the product/system
  - Cen be retested automatically upon any release/version/sprint etc.
  - Each payload requires a unit test to be created individually
  - Coding as a means of hacking is slower by scales

PT it is! But…

- Consider that under modern methods (DevOps, Fast IT) there is never enough time for implementing these techniques.
- Hence, Reversing, Memory Analysis or Debugging are not feasible or practical.

Extreme Penetration Testing

- We need a testing method that is:
  - Simple
  - Comprehensive
  - Accurate, deep and pin pointed
  - FAST!
  - Controllable

Extreme Penetration Testing

- The method:
  - Create a "basic flow" unit test code
  - Sanity test the unit test on testing environment. Verify that the unit test can be run from within a development environment (e.g.Studio)
  - Identify key points of the flow that are to be tested

Extreme Penetration Testing

The method (cont.)

- In the key points:
  - Modify the "basic flow" unit test:
    - Save all parameters, even binary, encrypted, signed, etc. parameters to one single file, serialized.
    - After save – toggle a breaking point.
    - After breakpoint – Read the file and reseed all parameters back from the accepted values.

Familiar concept?

A "file-based" runtime proxy

Extreme Penetration Testing

The method (cont.)

- When the test is stopped –
  - Read the serialized file, using whatever text editor
  - Modify the parameters, web-proxy-style
  - Save

Familiar concept?

A “file-based” runtime proxy

Extreme Penetration Testing

- Demo (by Team Incomplete)
- Disclaimer, scale, shortcuts & others

Extreme Penetration Testing

- Management
  - Having a custom testing method is nice
  - PT is nothing without control

Extreme Penetration Testing

- PT management is an issue

B28 | 0x0D, 0x0A

	A	B	C	D	E	F	G
1		**PT is nothing without control**					
2		**Attack type**					
3		Fuzz	Payload sent	Reviewed response	Expected response	Acceptable?	Comments
4							
5		AAAAA%c					
6		AAAAA%d					
7		AAAAA%e					
8		AAAAA%f					
9		AAAAA%I					
10		AAAAA%o					
11		AAAAA%p					
12		AAAAA%s					
13		AAAAA%x					
14		AAAAA%n					
15		AAAAA%c					
16		AAAAA%d					
17		AAAAA%e					
18		AAAAA%f					
19		AAAAA%I					
20		AAAAA%o					
21		AAAAA%p					
22		AAAAA%s					
23		AAAAA%x					
24		AAAAA%n					
25		';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//\";alert(String.fromCharCode					
26		CRLF					
27		NULL					
28		0x0D, 0x0A	0x0D 0x0A	Faild signing	successful signing	No	
29		0x00					
30		\";alert('XSS');//					
31		</TITLE><SCRIPT>alert("XSS");</SCRIPT>					
32		<DIV STYLE="background-image: url(javascript:alert('XSS'))">					
33		<DIV STYLE="background-image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029'\0029">					
34							
35							

Extreme Penetration Testing

Summary

Palantir security

Summary

- One cannot rely on a single security testing approach
  - Design Review
  - Code Analysis
  - Penetration Testing
- Some system testing require innovation
  - Known tools irrelevant
  - Known payloads irrelevant
- Know thy limitations
  - Relying on standard approaches is not enough

Summary

- Innovate
  - Consider each technique advantages and disadvantages
  - Consider PT scope and limitations
  - Create your own technique
- Manage
  - Control the PT process using custom technique
- Success
  - Get that Bonanza!

Q&A

Questions?