Notes from Prof. Brun

- Project plan due next Tuesday (email him if you have questions)
- Be ready to present project plans on Tuesday (10 minutes per group)
Software Security

Ben Ransford
ransford@cs.umass.edu

CS621 Fall 2012
Ross Anderson, *Security Engineering*

≈

Saltzer & Kaashoek, *P. of C. S. D.*
“Security engineering is about building systems to remain dependable in the face of malice, error, or mischance.”
Security = Policy + Mechanism + Assurance + Incentive
Security = Policy + Mechanism + Assurance + Incentive

Insecurity ≈ How can I break this system?
Threat Modeling

- ... is your job in system design
- Think like an attacker
- Understand and prioritize incentives
- Imagine a realistic attacker
Attack Surface

- Which parts of your system interface with other stuff?
  - Network ports, I/O
  - Command-line inputs
  - Dependencies on other systems
Attacker Incentives

- For each element of attack surface:
  - What can a successful attacker gain?
  - What’s it worth?
# (Some) Kinds of Attackers

<table>
<thead>
<tr>
<th>Value</th>
<th>Example</th>
<th>Attacker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Generic PC</td>
<td>Script kiddie</td>
</tr>
<tr>
<td>Medium</td>
<td>Personal bank account</td>
<td>Phisher</td>
</tr>
<tr>
<td>High</td>
<td>State nuclear program</td>
<td>Another state</td>
</tr>
</tbody>
</table>
Script Kiddies

- Largely unskilled; main resource = time
- Use pre-packaged exploits
- May wish to sell compromised resources (e.g., sell zombie PCs to botnet)
Midrange “Hackers”

- Somewhat skilled; may have specific targets
- May be willing to use social engineering
- Motivations include fame, revenge, vandalism, $$$

High-End Hackers

- Deep understanding of target
- Write exploits
- These days, sell exploits for $$$$$
# High-End Hackers

<table>
<thead>
<tr>
<th>Software</th>
<th>Price Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADOBE READER</td>
<td>$5,000–$30,000</td>
</tr>
<tr>
<td>MAC OSX</td>
<td>$20,000–$50,000</td>
</tr>
<tr>
<td>ANDROID</td>
<td>$30,000–$60,000</td>
</tr>
<tr>
<td>FLASH OR JAVA BROWSER PLUG-INS</td>
<td>$40,000–$100,000</td>
</tr>
<tr>
<td>MICROSOFT WORD</td>
<td>$50,000–$100,000</td>
</tr>
<tr>
<td>WINDOWS</td>
<td>$60,000–$120,000</td>
</tr>
<tr>
<td>FIREFOX OR SAFARI</td>
<td>$60,000–$150,000</td>
</tr>
<tr>
<td>CHROME OR INTERNET EXPLORER</td>
<td>$80,000–$200,000</td>
</tr>
<tr>
<td>IOS</td>
<td>$100,000–$250,000</td>
</tr>
</tbody>
</table>
Even Higher-End Hackers

- E.g., state agencies (NSA, Mossad)
- Specific targets for *espionage* or *sabotage*
- Advanced persistent threats — get into target and stay there
Cryptography
Do’s & don’ts

Note: cryptography != security
Rule #1

Don’t design your own cipher!
Use an existing one.

== Use AES.
Don’t pull a Mifare

Figure 1: (a) Source image of layer 2 after edge detection; (b) after automated template detection.
Rule #2

Don’t rely on security through obscurity. Your system’s design will become known.

== Assume only the keys are secret.
Rule #3

Don’t use randomness incorrectly or use predictable “randomness.”
Bad randomness makes attacks easy.

== Use TRNG or a good seeded PRNG
Good PRNG

- Doesn’t repeat itself (long *period*)
- Does use sources of “random” bits
Good PRNG

• Doesn't repeat itself (long period)
• Does use sources of "random" bits
Bad PRNG

Easy to guess secrets!
Bad PRNG

Easy to guess secrets!

Dismantling iClass and iClass Elite

Flavio D. Garcia¹, Gerhard de Koning Gans¹, Roel Verdult¹, and Milosch Meriac²

¹ Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands.
   {flaviog,gkoningg,rverdult}@cs.ru.nl
² Bitmanufaktur GmbH, Germany.
   milosch.meriac@bitmanufaktur.de
Note: Multiple PRNGs

(demo of Linux /dev/urandom vs. /dev/random)

Don’t use urandom when you want random.
Harping on Randomness

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices

Nadia Heninger†* Zakir Durumeric†* Eric Wustrow‡ J. Alex Halderman‡

† University of California, San Diego
nadiah@cs.ucsd.edu

‡ The University of Michigan
{zakir, ewust, jhalderm}@umich.edu
Harping on Randomness

"We found that 5.57% of TLS hosts and 9.60% of SSH hosts share public keys in an apparently vulnerable manner, due to either insufficient randomness during key generation or device default keys" (source: factorable.net)
Debian OpenSSL disaster

(Don’t trust your tools blindly!)
Greatest Hits
(and how not to get hit)

please put on your C/C++ hats
Buffer overflows
(super common)

strcpy(dest, user_supplied_input);
Use-after-free
(somewhat common)

```c
void f (p_t *p) { ...; free(p); }

f(my_pointer);
*my_pointer = 0x1234;
```
Double free
(not all that common)

void f (p_t *p) { ...; free(p); }

f(my_pointer);
free(my_pointer);
Input validation
Cross-site scripting
(super super super super common)

Hello my name is <script>stealStuff();</script>