Where do security bugs come from?

MIT 6.858 (Computer Systems Security), September 19th, 2012

Paul Youn
• Technical Director, iSEC Partners
• MIT 18/6-3 ('03), M.Eng ’04

Tom Ritter
• Security Consultant, iSEC Partners
• (Research) Badass
Agenda

• What is a security bug?
• Who is looking for security bugs?
• Trust relationships
• Sample of bugs found in the wild
• Operation Aurora
• Stuxnet
• I’m in love with security; whatever shall I do?
What is a Security Bug?

• What is security?
• Class participation: Tacos, Salsa, and Avocados (TSA)
What is security?

“A system is secure if it behaves precisely in the manner intended – and does nothing more” – Ivan Arce

• Who knows exactly what a system is intended to do? Systems are getting more and more complex.

• What types of attacks are possible?

First steps in security: define your security model and your threat model
Threat modeling: T.S.A.

- Logan International Airport security goal #3: prevent banned substances from entering Logan
- Class Participation: What is the threat model?
  - What are possible avenues for getting a banned substance into Logan?
  - Where are the points of entry?
- Threat modeling is also critical, you have to know what you’re up against (many engineers don’t)
Who looks for security bugs?

- Engineers
- Criminals
- Security Researchers
- Pen Testers
- Governments
- Hacktivists
- Academics
Engineers (create and find bugs)

• Goals:
  • Find as many flaws as possible
  • Reduce incidence of exploitation
• Thoroughness:
  • Need coverage metrics
  • At least find low-hanging fruit
• Access:
  • Source code, debug environments, engineers
  • Money for tools and staff
People care about features, not security (until something goes wrong)

Engineers typically only see a small piece of the puzzle

“OMG PDF WTF” (Julia Wolf, 2010)

- How many lines of code in Linux 2.6.32?
- How many lines in Windows NT 4?
- How many in Adobe Acrobat?
People care about features, not security (until something goes wrong)

Engineers typically only see a small piece of the puzzle

“OMG PDF WTF” (Julia Wolf, 2010)

- How many lines of code in Linux 2.6.32?
  - 8 – 12.6 million

- How many lines in Windows NT 4?
  - 11-12 million

- How many in Adobe Acrobat?
  - 15 million
Criminals

• Goals:
  • Money (botnets, CC#s, blackmail)
  • Stay out of jail
• Thoroughness:
  • Reliable exploits
  • Don’t need 0-days (but they sure are nice)
• Access:
  • Money
  • Blackbox testing
Goals:
- Column inches from press, props from friends
- Preferably in a trendy platform

Thoroughness:
- Don’t need to be perfect, don’t want to be embarrassed

Access:
- Casual access to engineers
- Source == Lawyers
Pen Testers

• **Goals:**
  • Making clients and users safer
  • Finding vulns criminals would use

• **Thoroughness:**
  • Need coverage
  • Find low-hanging fruit
  • Find high impact vulnerabilities
  • Don’t fix or fully exploit

• **Access:**
  • Access to Engineers
  • Access to Source
  • Permission
• **Goals:**
  - Attack/espionage
  - Defend
• **Thoroughness:**
  - Reliable exploits
• **Access:**
  - Money
  - Talent
  - Time
Hacktivists

• Goals:
  • Doing something “good”
  • Stay out of jail
• Thoroughness:
  • Reliable exploits
  • Don’t need 0-days
• Access:
  • Talent
  • Plentiful targets
• Goals:
  • Finding common flaws and other general problems
  • Developing new crypto
  • Make something cool and useful
  • Make everyone safer
• Thoroughness:
  • Depth in area of research
• Access:
  • Creating new things
  • Blackbox
Techniques

• With access:
  • Source code review
  • Engineer interviews
  • Testing in a controlled environment

• Without access:
  • Blackbox testing
  • Fuzzing (give weird inputs, see what happens)
  • Reverse Engineering
  • Social Engineering
Overall Goals

• All are looking for the similar things: vulnerable systems
• Let’s dive in and look at vulns that we all look for
Bad Engineering Assumptions
• Two modes of operation: image and radiation treatment
• Intended invariant: in radiation treatment mode, a protective focusing shield must be in place
Shield code was something like:

```c
//global persistent variable, single byte value
UBL protectiveShield;  //zero if shield isn’t needed
...
//do we need a shield?
if(treatmentMode) then
{
    protectiveShield++;  
}
else {
    protectiveShield = 0;
}
...
if(protectiveShield) {
    putShieldInPlace();
}
else {
    removeShield();
}
```

Therac-25
• Flawed assumption: protectiveShield would always be non-zero in treatment mode
• Impact: people actually died
• Flawed assumption: protectiveShield would always be non-zero in treatment mode
• Impact: people actually died
• My classmate’s conclusion: “I learned to never write medical software”
Think like a security researcher:

• What assumptions are being made?
• Which assumptions are wrong?
• What can you break if the assumption is wrong?
The Confused Deputy

• Tricking an authority into letting you do something you shouldn’t be able to do
• Most security problems could fall under this broad definition
“How to Shop for Free Online”* (security researcher and academic)

- Three-party payment systems (Cashier as a Service):
  - Merchant (seller)
  - Payment provider
  - Cheater User
- Communication between parties go through the user

The Confused Deputy

0: I’d like to buy Book
1: Pay my CaaS $10, TxID: 123
2: Here is $10 for TxID: 123
3: TxID: 123 has been paid for
4: OK
5: Transaction complete
6: I’m done!
7: I’ll send Book!
The Confused Deputy

0: I'd like to buy Book

1: Pay my CaaS $10, TxID: 123

2: Here is $1 for TxID: 123

3: TxID: 123 has been paid for

4: OK

5: Transaction complete

6: I'm done!

7: I'll send Book!
The Confused Deputy

- The merchant thinks something ties the payment amount to the transaction
- Impact: shopping for free
- Solutions?
- Read the paper, lots of things can and do go wrong
Sample of bugs found in the wild
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249
username=tom&password=hunter2
Stack

Physical Layer

Link Layer

Internet Protocol

ARP

DHCP

HTTP

TCP

UDP

ICMP

TLS

HTTP

HTTP
<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>50 4F 53 54 20 2F 74 61 72 67 65 74 20 48 54 54</td>
</tr>
<tr>
<td>00000010</td>
<td>50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 65 78 61</td>
</tr>
<tr>
<td>00000020</td>
<td>6D 70 6C 65 2E 63 6F 6D 0A 55 53 74 32 2D 41</td>
</tr>
<tr>
<td>00000030</td>
<td>67 65 6E 74 3A 20 4D 6F 77 73 69 6C 6C 6F 31 2F</td>
</tr>
<tr>
<td>00000040</td>
<td>30 20 28 57 69 6E 64 6F 77 73 73 20 4E 54 20 36 2E</td>
</tr>
<tr>
<td>00000050</td>
<td>31 3B 20 2D 4F 67 34 36 39 35 34 33 31 2E</td>
</tr>
<tr>
<td>00000060</td>
<td>30 29 20 24 67 65 63 6F 72 65 73 36 6F 2F 32 30</td>
</tr>
<tr>
<td>00000070</td>
<td>31 20 46 69 72 65 66 6F 77 31 34 32 2E 30 2E</td>
</tr>
<tr>
<td>00000080</td>
<td>30 0A 43 6F 66 69 6B 61 6E 64 20 73 73 73 69 6F</td>
</tr>
<tr>
<td>00000090</td>
<td>6E 69 64 3D 64 38 65 35 3D 63 66 63 61 32 64 63 63 66</td>
</tr>
<tr>
<td>000000A0</td>
<td>38 39 36 66 64 37 63 62 34 63 66 32 30 33 31 36 32 63</td>
</tr>
<tr>
<td>000000B0</td>
<td>61 32 34 39 0D 0A 73 65 73 73 69 66 69 3D 61 36 69 3D</td>
</tr>
<tr>
<td>000000C0</td>
<td>64 3D 61</td>
</tr>
<tr>
<td>Source IP</td>
<td>Destination IP</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>349 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>350 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>351 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>354 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>355 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>356 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>358 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>359 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>361 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>362 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>Time</td>
<td>Source IP</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------</td>
</tr>
<tr>
<td>349</td>
<td>74.125.227.62</td>
</tr>
<tr>
<td>350</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>351</td>
<td>74.125.227.62</td>
</tr>
<tr>
<td>354</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>355</td>
<td>74.125.227.62</td>
</tr>
<tr>
<td>356</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>358</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>359</td>
<td>74.125.227.62</td>
</tr>
<tr>
<td>361</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>362</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>Source IP</td>
<td>Destination IP</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>349 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>350 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>351 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>354 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>355 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>356 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>358 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>359 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>361 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>362 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>Source IP</td>
<td>Destination IP</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>349 74.125.227.62</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>350 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>351 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>354 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>355 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>356 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>358 192.168.24.100</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>359 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>361 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>362 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>Source IP</td>
<td>Destination IP</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>349 74.123.123.2</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>350 123.456.789</td>
<td>97.107.139.108</td>
</tr>
<tr>
<td>351 74.123.123.2</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>354 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>355 74.123.123.2</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td>356 97.107.139.108</td>
<td>192.168.24.100</td>
</tr>
<tr>
<td></td>
<td>358 192.168.24.100</td>
</tr>
<tr>
<td></td>
<td>359 74.123.245.62</td>
</tr>
<tr>
<td></td>
<td>361 97.107.139.108</td>
</tr>
<tr>
<td></td>
<td>362 97.107.139.108</td>
</tr>
</tbody>
</table>
### Traffic Analysis. Huge Field

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Data Type</th>
<th>Data Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>349 74.125.227.62</td>
<td>97.107.139.108</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>720 Application Data, Application Data</td>
</tr>
<tr>
<td>350 192.168.24.100</td>
<td>97.107.139.108</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>351 97.107.139.108</td>
<td>192.168.24.100</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>354 97.107.139.108</td>
<td>192.168.24.100</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>355 74.125.227.62</td>
<td>97.107.139.108</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>356 97.107.139.108</td>
<td>192.168.24.100</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>358 192.168.24.100</td>
<td>97.107.139.108</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
<tr>
<td>359 97.107.139.108</td>
<td>192.168.24.100</td>
<td>TLSv1</td>
<td>Application Data</td>
<td>Application Data</td>
</tr>
</tbody>
</table>
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

username=tom&password=hunter2
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249
username=tom&password=hunter2

Attacker wants to know this
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

username=tom\&password=hunter2
HTTP

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

username=tom&password=hunter2
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

sessionId=a
### HTTP

<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>50 4F 53 54 20 2F 74 61 72 67 65 74 20 48 54 54</td>
</tr>
<tr>
<td>00000010</td>
<td>50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 65 78 61</td>
</tr>
<tr>
<td>00000020</td>
<td>6D 70 6C 65 2E 63 6F 6D 0A 55 33 65 78 72 2D 41</td>
</tr>
<tr>
<td>00000030</td>
<td>67 65 6E 74 3A 20 4D 6F 77 73 20 3A 31 3B 20 32</td>
</tr>
<tr>
<td>00000040</td>
<td>30 20 28 57 69 6E 64 6F 77 73 20 2D 4E 54 54 55</td>
</tr>
<tr>
<td>00000050</td>
<td>31 3B 20 57 4F 57 36 34 3B 20 72 33 3A 34 32 3E</td>
</tr>
<tr>
<td>00000060</td>
<td>30 29 20 4F 66 69 65 63 66 6F 2F 32 30 30 30 30</td>
</tr>
<tr>
<td>00000070</td>
<td>30 20 46 69 72 65 66 6F 77 3E 32 3F 31 34 32 3E</td>
</tr>
<tr>
<td>00000080</td>
<td>0D 0A 43 3F 6F 6B 69 65 3A 20 73 73 3A 39 39 6F</td>
</tr>
<tr>
<td>00000090</td>
<td>6E 69 64 3D 44 38 65 38 66 63 61 32 64 63 36 66</td>
</tr>
<tr>
<td>000000A0</td>
<td>36 36 66 64 37 33 62 34 63 63 32 30 30 30 31 62</td>
</tr>
<tr>
<td>000000B0</td>
<td>61 32 34 39 0D 0A 4D 73 65 73 73 69 6F 72 61 6C</td>
</tr>
<tr>
<td>000000C0</td>
<td>64 33 61</td>
</tr>
<tr>
<td>Offset (h)</td>
<td>00</td>
</tr>
<tr>
<td>-----------</td>
<td>----</td>
</tr>
<tr>
<td>00000000</td>
<td>00</td>
</tr>
<tr>
<td>00000010</td>
<td>53</td>
</tr>
<tr>
<td>00000020</td>
<td>31</td>
</tr>
<tr>
<td>00000030</td>
<td>65</td>
</tr>
<tr>
<td>00000040</td>
<td>74</td>
</tr>
<tr>
<td>00000050</td>
<td>57</td>
</tr>
<tr>
<td>00000060</td>
<td>4F</td>
</tr>
<tr>
<td>00000070</td>
<td>65</td>
</tr>
<tr>
<td>00000080</td>
<td>72</td>
</tr>
<tr>
<td>00000090</td>
<td>6B</td>
</tr>
<tr>
<td>000000A0</td>
<td>30</td>
</tr>
<tr>
<td>000000B0</td>
<td>31</td>
</tr>
</tbody>
</table>

187 Bytes
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

sessionid=d
### HTTP

```
<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>0A</th>
<th>0B</th>
<th>0C</th>
<th>0D</th>
<th>0E</th>
<th>0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00</td>
<td>2E</td>
<td>31</td>
<td>01</td>
<td>73</td>
<td>65</td>
<td>73</td>
<td>73</td>
<td>69</td>
<td>6E</td>
<td>69</td>
<td>64</td>
<td>3D</td>
<td>64</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>00000010</td>
<td>4F</td>
<td>53</td>
<td>54</td>
<td>20</td>
<td>2F</td>
<td>74</td>
<td>61</td>
<td>72</td>
<td>67</td>
<td>65</td>
<td>74</td>
<td>20</td>
<td>48</td>
<td>54</td>
<td>54</td>
<td>50</td>
</tr>
<tr>
<td>00000020</td>
<td>2F</td>
<td>31</td>
<td>00</td>
<td>00</td>
<td>0A</td>
<td>48</td>
<td>6F</td>
<td>73</td>
<td>74</td>
<td>3A</td>
<td>20</td>
<td>65</td>
<td>78</td>
<td>61</td>
<td>6D</td>
<td>70</td>
</tr>
<tr>
<td>00000030</td>
<td>6C</td>
<td>65</td>
<td>2E</td>
<td>63</td>
<td>6F</td>
<td>6D</td>
<td>0D</td>
<td>0A</td>
<td>55</td>
<td>73</td>
<td>65</td>
<td>72</td>
<td>2D</td>
<td>41</td>
<td>67</td>
<td>65</td>
</tr>
<tr>
<td>00000040</td>
<td>6E</td>
<td>74</td>
<td>3A</td>
<td>20</td>
<td>4D</td>
<td>6F</td>
<td>7A</td>
<td>69</td>
<td>6C</td>
<td>6C</td>
<td>61</td>
<td>2F</td>
<td>35</td>
<td>2E</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>00000050</td>
<td>28</td>
<td>57</td>
<td>69</td>
<td>6E</td>
<td>64</td>
<td>6F</td>
<td>77</td>
<td>73</td>
<td>20</td>
<td>4E</td>
<td>54</td>
<td>20</td>
<td>36</td>
<td>00</td>
<td>3B</td>
<td>20</td>
</tr>
<tr>
<td>00000060</td>
<td>57</td>
<td>4F</td>
<td>57</td>
<td>36</td>
<td>34</td>
<td>3B</td>
<td>20</td>
<td>72</td>
<td>76</td>
<td>3A</td>
<td>31</td>
<td>34</td>
<td>2E</td>
<td>30</td>
<td>29</td>
<td>20</td>
</tr>
<tr>
<td>00000070</td>
<td>47</td>
<td>65</td>
<td>63</td>
<td>6B</td>
<td>6F</td>
<td>2F</td>
<td>32</td>
<td>30</td>
<td>31</td>
<td>30</td>
<td>31</td>
<td>30</td>
<td>31</td>
<td>20</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>00000080</td>
<td>69</td>
<td>72</td>
<td>65</td>
<td>66</td>
<td>6F</td>
<td>78</td>
<td>2F</td>
<td>31</td>
<td>34</td>
<td>2E</td>
<td>30</td>
<td>00</td>
<td>0D</td>
<td>0A</td>
<td>43</td>
<td>6F</td>
</tr>
<tr>
<td>00000090</td>
<td>6F</td>
<td>6B</td>
<td>69</td>
<td>65</td>
<td>3A</td>
<td>20</td>
<td>01</td>
<td>38</td>
<td>65</td>
<td>38</td>
<td>66</td>
<td>63</td>
<td>61</td>
<td>32</td>
<td>64</td>
<td>63</td>
</tr>
<tr>
<td>000000A0</td>
<td>30</td>
<td>66</td>
<td>38</td>
<td>39</td>
<td>36</td>
<td>66</td>
<td>64</td>
<td>37</td>
<td>63</td>
<td>62</td>
<td>34</td>
<td>63</td>
<td>62</td>
<td>30</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>000000B0</td>
<td>31</td>
<td>62</td>
<td>61</td>
<td>32</td>
<td>34</td>
<td>39</td>
<td>0D</td>
<td>0A</td>
<td>0D</td>
<td>0A</td>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

..1.sessionid=d3

OST /target HTTP
/1...Host: examp
le.com..User-Agent
nt: Mozilla/5.0
(Windows NT 6.;
WOW64; rv:14.0)
Gecko/20100101 F
irefox/14.0...Co
okie: .8e8fca2dc
0f896fd7cb4cb003
1ba249...."}

186 Bytes
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

sessionid=da
HTTP

POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
  Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

  sessionid=da

188 Bytes
POST /target HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0)
Gecko/20100101 Firefox/14.0.1
Cookie: sessionid=d8e8fca2dc0f896fd7cb4cb0031ba249

sessionId=d8

187 Bytes
Fundamental Internet Protocols Still Have Bugs!

- SSL!
- DNS!
- DNSSEC (Ho Boy, DNSSEC)
- IPv6 (Ho Boy, IPv6)
Memory Corruption: Operation Aurora
Operation Aurora (government)

Use after free vulnerability (MS10-002 – Remote Code Execution in IE 5-8)

• Memory typically has a reference counter (how many people have a handle to me?)
• Improper reference counter allowed Javascript to still reference a function in a freed block of memory
  • Free memory
  • Heap spray attack code (likely it gets written to the freed block because of how IE memory management works)
  • Call function
  • Fairly reliable code execution
function window :: onload ()
{
    var SourceElement =
        document.createElement("div");
    document.body.appendChild
        (SourceElement);
    var SavedEvent = null;
    SourceElement.onclick = function () {
        SavedEvent =
            document.createEventObject (event);
        document.body.removeChild
            (event.srcElement);
    }
    SourceElement.fireEvent ("onclick");
    SourceElement = SavedEvent.srcElement;
}
function window :: onload ()
{
    var SourceElement =
        document.createElement("div");
    document.body.appendChild
        (SourceElement);
    var SavedEvent = null;
    SourceElement.onclick = function () {
        SavedEvent =
            document.createEventObject (event);
        document.body.removeChild
            (event.srcElement);
    }
    SourceElement.fireEvent ("onclick");
    SourceElement = SavedEvent.srcElement;
}
Operation Aurora

• Heap Spray!
  • Create a bunch of elements with attack code and then free them (attack code gets written to lots of heap blocks)
  • IE Small Block Manager Reuses memory pages
• Call the event pointing to freed memory
• Code execution!
Operation Aurora

- Valuable exploit! How was it used?
- Social Engineering (get someone to click a link), almost always the weakest link
- Escalate privileges (cached credentials)
- Spread (Active Directory, brute force attack)
- Gather (source code, financial data)
- Exfiltration (to China, out of intranet on Christmas)
Operation Aurora

- Advanced Persistent Threat
  - Advanced attackers with talent (zero days) and time (months or years)
  - Targeted attacks (not just going after the vulnerable)
  - Non-traditional attacks, likely hard to monetize
- Whodunit?
Stuxnet (gov’t / security researcher)
Stuxnet (so Amazing)

- Five zero-day vulnerabilities
- Two stolen certificates
- Almost surgically targeted
- Eight propagation methods
- Partridge in a malware pear tree
Stuxnet

The Target

- Mixed MS Windows environment = Redundant
- Not exploiting memory corruption = Reliable
- Target: Iranian air-gapped networks operating centrifuges to enrich nuclear material (Natanz)
- How can you get a foot in the door? USB keys
USB Vulnerability

Zero-Day* Vulnerabilities:

• **MS10-046** (Shell LNK / Shortcut)
• **MS10-061** (Print Spooler Service)
• **MS10-073** (Win32K Keyboard Layout)
• **MS08-067** (NetPathCanonicalize()), (Patched)
  
  [http://www.phreedom.org/blog/2008/decompiling-ms08-067/](http://www.phreedom.org/blog/2008/decompiling-ms08-067/)
• **MS10-092** (Task Scheduler)
• **CVE-2010-2772** (Siemens SIMATIC Static Password)
You know, shortcuts and such
Where does the icon come from?
Loaded from a CPL (Control Panel File) specified by the user
A CPL is just a DLL
USB keys have attack DLL and a shortcut referencing the DLL
Plugging in the USB stick leads to arbitrary code execution
Flaw: we should run a user-specified DLL to display an icon for a shortcut?!
But I’m not Admin!

Zero-Day* Vulnerabilities:

- **MS10-046** (Shell LNK / Shortcut)
- **MS10-061** (Print Spooler Service)
- **MS10-073** (Win32K Keyboard Layout)
- **MS08-067** (NetPathCanonicalize()), (Patched)
  [http://www.phreedom.org/blog/2008/decompiling-ms08-067/](http://www.phreedom.org/blog/2008/decompiling-ms08-067/)
- **MS10-092** (Task Scheduler)
- **CVE-2010-2772** (Siemens SIMATIC Static Password)
• Keyboard layouts can be loaded into Windows
• In XP, anyone can load a keyboard layout (later version only allow admins)
• Integer in the layout file indexes a global array of function pointers without proper bound checking
• Call any function, but I want to call my function...
• How do we call attack code?
• Find the pointer to the global function array
• Find a pointer into user-land (modifiable by your program)
• Inject your attack code there
• Call the modified function (runs as SYSTEM)
Flaws: improper bound checking on the keyboard layout function index and allowing standard users to specify layouts
But I’m not an Admin!

Zero-Day* Vulnerabilities:

• MS10-046 (Shell LNK / Shortcut)
• MS10-061 (Print Spooler Service)
• MS10-073 (Win32K Keyboard Layout)
• MS08-067 (NetPathCanonicalize()), (Patched)  
  http://www.phreedom.org/blog/2008/decompiling-ms08-067/
• MS10-092 (Task Scheduler)
• CVE-2010-2772 (Siemens SIMATIC Static Password)
MS10-092 (Task Scheduler)

- Standard users can create and edit scheduled tasks (XML)
- After a task is created, a CRC32 checksum is generated to prevent tampering
- ... CRC32 ...
• Standard users can create and edit scheduled tasks (XML)
• After a task is created, a CRC32 checksum is generated to prevent tampering
• ... CRC32 ...
CRC32

let me Google that for you

crc32

Google Search  I'm Feeling Lucky

Was that so hard?
Cyclic redundancy check

From Wikipedia, the free encyclopedia

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents; on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

CRCs are so called because the check (data verification) value is a redundancy (it adds no information to the message) and the algorithm is based on cyclic codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function. The CRC was invented by W. Wesley Peterson in 1961; the 32-bit polynomial used in the CRC function of Ethernet and many other standards is the work of several researchers and was published during 1975.

Contents
1 Introduction
2 Application
3 CRCs and data integrity
4 Computation of CRC
5 Mathematics of CRC
6 8-bit CRC and 16-bit CRC polynomials
7 Specification of CRC
8 Commonly used and standardized CRCs
9 See also
10 References
11 External links

Introduction

CRCs are based on the theory of cyclic error-correcting codes. The use of systematic cyclic codes, which encode messages by adding a fixed-length check value, for the purpose of error detection in communication networks, was first proposed by W. Wesley Peterson during 1961. CRCs are not only simple to implement but also have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages. This is important because burst errors are common transmission errors in many communication channels, including magnetic and optical storage devices. Typically an n-bit CRC applied to a data block of arbitrary length will detect any single error burst not longer than n bits and will detect a fraction 1−2−n of all longer error bursts.

Specification of a CRC code requires definition of a so-called generator polynomial. This polynomial resembles the divisor in a polynomial long division, which takes the message as the dividend and in which the quotient is discarded and the remainder becomes the result, with the important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field. The length of the remainder is always less than the length of the generator polynomial, which therefore determines how long the result can be.

In practice, all commonly used CRCs employ the finite field GF(2). This is the field of two elements, usually called 0 and 1, comfortably matching computer architecture. The rest of this article will discuss only these binary CRCs, but the principles are more general.

The simplest error-detection system, the parity bit, is in fact a trivial 1-bit CRC: it uses the generator polynomial x+1.

Application

A CRC-enabled device calculates a short, fixed-length binary sequence, known as the check value or improperly the CRC, for each block of data to be sent or stored and appends it to the data, forming a codeword. When a codeword is received or read, the device either compares its check value with one freshly calculated from the data block, or equivalently, performs a CRC on the whole codeword and compares the resulting check value with an expected residue constant. If the check values do not match, then the block contains a data error. The device may take corrective action, such as retransmitting the block or requesting that it be sent again. Otherwise, the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking).
“However, [CRCs] are not suitable for protecting against intentional alteration of data.” – Wikipedia (Cyclic redundancy check)
MS10-092 (Task Scheduler)

- Created task as normal user, record CRC32 value
- Modified user definition in the task to LocalSystem
- Take CRC32 of the task XML, pad until the CRC32 matches original
• Created task as normal user, record CRC32 value
• Modified user definition in the task to LocalSystem
• Take CRC32 of the task XML, pad until the CRC32 matches original
• ??????
• Profit!
Flaw:

We should use CRC32 to-

NEVER USE CRC32 FOR ANYTHING!
“Our job is to read one more sentence in the man page than the developer did.” –Chris Palmer (former iSECer)

• Be really curious
• Think about how components interact with each other
Zero-Day* Vulnerabilities:

- MS10-046  (Shell LNK / Shortcut)
- **MS10-061**  (Print Spooler Service)
- MS10-073  (Win32K Keyboard Layout)
- MS08-067  (NetPathCanonicalize()), (Patched)
  
  [http://www.phreedom.org/blog/2008/decompiling-ms08-067/](http://www.phreedom.org/blog/2008/decompiling-ms08-067/)
- MS10-092  (Task Scheduler)
- CVE-2010-2772  (Siemens SIMATIC Static Password)
Enumerates printer shares
Connects to printer and asks to print two files to SYSTEM32
Should fail?! Printer should connect as Guest, which shouldn’t have privilege to create files in SYSTEM32
• “//We run as system because in XP the guest account doesn’t have enough privilege to do X/Y/Z”

• Stuxnet payload is dropped
• How do we execute? Enter the MOF
• MOF files are basically script files
• A process monitors the following directory for new files and executes them:
  Windows\System32\wbem\mof\n• MOF file executes the Stuxnet payload
MS10-061 (Print Spooler Service)

Flaws:

• Printer spooler runs as SYSTEM (highest privilege) and allows arbitrary files to be written to arbitrary places
• File creation leads to arbitrary code execution
Let’s Spread!

Zero-Day* Vulnerabilities:

• MS10-046 (Shell LNK / Shortcut)
• MS10-061 (Print Spooler Service)
• MS10-073 (Win32K Keyboard Layout)
• MS08-067 (NetPathCanonicalize()), (Patched)  
  http://www.phreedom.org/blog/2008/decompiling-ms08-067/
• MS10-092 (Task Scheduler)
• CVE-2010-2772 (Siemens SIMATIC Static Password)
Known, patched (recent) vulnerability that allowed you to drop a payload and schedule it for execution

Flaws:

- Unpatched systems
- RPC flaw that allows unauthorized remote users to schedule tasks
Rootkits

- Goal: maintain control in secret
- Anti-Virus: Behavior Blocking
  - Hook (modify behavior) of ntdll.dll (used to load DLLs)
  - Load a fake DLL name
  - AV says “that doesn’t exist, that’s fine”
  - Hook reroutes to a Stuxnet DLL
  - Hook “trusted” binaries (based on installed AV)
- Two stolen certificates:
  - Signs MrxCls.sys: launches Stuxnet on boot
  - Signs MRxNet.sys: hides Stuxnet filesystem objects and hooks new filesystem objects
Zero-Day* Vulnerabilities:

- MS10-046 (Shell LNK / Shortcut)
- MS10-061 (Print Spooler Service)
- MS10-073 (Win32K Keyboard Layout)
- MS08-067 (NetPathCanonicalize()), (Patched)
  [Link](http://www.phreedom.org/blog/2008/decompiling-ms08-067/)
- MS10-092 (Task Scheduler)
- **CVE-2010-2772** (Siemens SIMATIC Static Password)
When and Where?

• Stuxnet is targeted for the Natanz Nuclear Facility
  • Targets a configuration with six centrifuge cascades in a very specific configuration
  • Attacks specific controllers/hardware used at Natanz
  • Certainly had a test environment
• Where did the intelligence come from?
When and Where?

President Ahmadinejad’s homepage! Here he is at Natanz. Wait, what’s that on the screen?
## When and Where?

Full resolution photos?? ENHANCE!

### R-1 cascade model

<table>
<thead>
<tr>
<th>RCG</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row</td>
<td>43</td>
<td>42</td>
<td>41</td>
<td>40</td>
<td>39</td>
<td>38</td>
</tr>
<tr>
<td>Stage</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**RCG:** Rotor Control Group, a group of up to 28 centrifuges

**Stage:** Enrichment stage, with the general flow direction from right to left

**Row:** Row number of a centrifuge quadruple, corresponding to the floor markings
When and Where?

Don’t get too ‘Merica on me, we do it too...
CVE-2010-2772 (Static Password)

- Siemens’ controllers for centrifuges run WinCC
- WinCC SQL database servers
  - Connect using a hardcoded password
  - Loads Stuxnet as binary into a table
  - Executes binary as a stored procedure
CVE-2010-2772 (Static Password)

- Step7 DLL is renamed and replaced with an attack DLL
- If the PLC matches the desired profile, it’s infected
- Breaks centrifuges by spinning them in weird ways while reporting everything is fine
Stuxnet: Fun Facts

• Black Market value of these vulns... probably millions
• Probably set back Iran’s nuclear program by years
• Stolen code signing certificates actually signed the virus to make it look legitimate
• Virus phoned command and control centers to gather data, update, and presumably limit the scope of infection
• Whodunit?
• Learn more:
  • [http://www.youtube.com/watch?v=rOwMW6agpTI](http://www.youtube.com/watch?v=rOwMW6agpTI)
  • [http://www.digitalbond.com/2012/01/31/langners-stuxnet-deep-dive-s4-video/](http://www.digitalbond.com/2012/01/31/langners-stuxnet-deep-dive-s4-video/)
  • [https://www.youtube.com/watch?v=rsXe2Gr2e3Q](https://www.youtube.com/watch?v=rsXe2Gr2e3Q)
But Wait... There's More!
Flame (Stuxnet’s Cousin)

- Spyware
- Does crazy things like:
  - Get all the GPS tags from all your photos
  - Get your contact list from any Bluetooth attached phone
  - Screenshots, keystroke logging, audio recording
MD5 is Broken (an Interlude)

• MD5 is broken because you can find collisions
• Specifically, chosen-prefix collision
• Demonstrated to be feasible in 2008 to generate a rogue CA ([http://marc-stevens.nl/research/papers/CR09-SSALMOdW.pdf](http://marc-stevens.nl/research/papers/CR09-SSALMOdW.pdf))
• Attack required 3 days running on 215 PS3s to find a collision
• Everyone panics, CAs stop using MD5 entirely
Microsoft forgot about one Microsoft Terminal Server still issuing MD5 certificates

Attackers devised a new way to find MD5 collisions

Harder challenges, 1 ms time window to get the right timestamp

Created an arbitrary MS root certificate for signing anything
Flame (Stuxnet’s Cousin)

- Microsoft forgot about one Microsoft Terminal Server still issuing MD5 certificates
- Attackers devised a new way to find MD5 collisions
- Harder challenges, 1 ms time window to get the right timestamp
- Created an arbitrary MS root certificate for signing anything
- .... Like Windows Updates
Flame (Stuxnet’s Cousin)

• “Oh Hai! I’m a Windows Update server!”
• “Oh Hello, I need an update.”
• “Here, have delicious delicious Flame!”
• “You silly goose, this is signed by MS! I’ll install it!”
I Love Security, What’s Next?

• Ethics in security
• Possible Careers
Ethics in Security

- Big ethical debates used to be:
  Responsible vs Full Disclosure
Ethics in Security

• Big ethical debates used to be:
  Responsible vs Full Disclosure

• Debate has shifted to:
  Disclosure vs Selling Weapons
Careers in Security

• Shape your job around your ethical standpoint, not vice versa
Careers in Security

• Shape your job around your ethical standpoint, not vice versa
• Write security relevant software
Careers in Security

• Shape your job around your ethical standpoint, not vice versa
• Write security relevant software
• Write (more) secure software
Careers in Security

• Shape your job around your ethical standpoint, not vice versa
• Write security relevant software
• Write (more) secure software
• Be a criminal
Careers in Security

- Shape your job around your ethical standpoint, not vice versa
- Write security relevant software
- Write (more) secure software
- Be a criminal
- Academia
Careers in Security

- Shape your job around your ethical standpoint, not vice versa
- Write security relevant software
- Write (more) secure software
- Be a criminal
- Academia
- Pen testing!
Pen Testing (at iSEC Partners)

• See new companies every 2-3 weeks and touch a wide variety of technologies
• Do awesome research (be a pen tester and a security researcher)
• Have a big impact by making the world safer
• Spend most of your time being clever and thinking
• See us at the job fair on Friday!
Come to Hotel Kendall on Thursday evening for free food and a talk about IPv6 by Tom (the American Room @6pm 9/20)

Help with material from:
- Aaron Grattafiori (Senior Security Consultant, iSEC Partners)
- Alex Stamos (Co-Founder iSEC Partners)

Images:
- http://www.babylifestyles.com/images/blog/2009/05/stork.gif
- http://www.dan-dare.org/FreeFun/Images/Cartoons/MoviesTV/BugsLifeWallpaper800.jpg
- http://www.sirlin.net/storage/street_fighter/dhalsim_yoga_flame.gif?__SQUARESPACE_CACHEVERSION=122658938179
- http://img.timeinc.net/time/photessays/2009/blame_3s/blame_3s_madoff.jpg
- http://www.imbase.info/images/safe-wallpapers/miscellaneous/1_other_wallpapers/16x62_1_other_wallpapers_hal_9000.jpg
- http://www.thecfpgroup.com/images/engineers.gif
- http://www.moviefanatic.com/gallery/yan-goming-m-nhve
UK Offices
Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Thame

North American Offices
San Francisco
Atlanta
New York
Seattle

Australian Offices
Sydney

European Offices
Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland