Objectives for Crime Scene Investigation

- Have some fun
- Prevent death by bullet points
- Introduce strategies for analyzing character corruption
- Compare TV Show and Software investigation of mutilated character corpses
- The latest version of the presentation is at www.i18nguy.com/How-To-Be-A-CSI.pdf

Our CSI Heroes
TV CSI vs. Software CSI

Gil Grissom
Catherine Willows
Nick Stokes
Sara Sidle
Greg Sanders
Warrick Brown
Dr. Al Robbins
Capt. Brass

Our Software CSI Heroes
How to be a CSI (encoding Crime Scene Investigator)

Crime Discovery On TV

- First steps
  - Close off crime scene and gather clues
    - Take lots of odd pictures
    - Collect potential forensic evidence
    - Identify the victim(s) (or what’s left of them)
    - Interview witnesses
      - They lie! They presume! They are prejudiced!
      - Put down either the jaded or newbie cop
        - If it’s suicide then how do you explain…

Crime Discovery in Software

- Characters have died.
  - Decomposition has already set in.
  - Who did it?

A woman calls 911 yelling: “There has been a Mojibake! Someone committed Mojibake!”

Characters have died.
Decomposition has already set in.
Who did it?

Crime Discovery in Software

- Characters Corrupted
  - Wrong characters
  - White or black boxes
- Collect forensic evidence
  - Screen captures, Keystrokes, Environment
- Witnesses/Users: “It happens when I…”
  - “The entire application is busted”
  - They lie, They presume, They’re prejudiced
- The newbie or jaded developer
  - “That vendor always does that. Our code is fine.”

Collecting Images

- Often included in bug reports
  - Better than nothing
  - You can detect patterns
  - But often not helpful
- Better to collect:
  - Expected bytes/chars
  - Actual bytes/chars
  - Fonts referenced, available
  - Expected/detected encoding
  - Keystrokes
  - A way to reproduce

*Mojibake = Japanese for garbage characters.
**Example Case History**

- Developers refused to debug based on an urban legend -
  - “Crime happens. Can’t catch these guys.”
  - “It’s Microsoft IE’s fault- Our software is fine”
- Instead of:
  `<meta http-equiv=Content-Type content="text/html; charset=utf-8">`
- The code said:
  `<meta http-equiv=Content-Type content="text/html; charset=utf8">`
  (Note the missing hyphen in “utf8”)

**Analysis On TV**

- Autopsy: More data collection
- Possibilities: What could have happened?
- Conjecture: How could it have happened?
- Reconstruction: Does it fit the data?
- Research: Does other data fit the theory?
  - Always ask the most narrowing question last
- Experiments
  - Measurements and proof of concept
  - Decay rates under abnormal circumstances

**Analysis on TV**

- Typical computer queries
  - How many trucks on the road?
  - How many on route 1?
  - How many on route 1 at 3pm?
  - How many stopped at Joe’s bar?
- Anatomy and decay science
- Autopsy
- Trajectory
- Reconstruction
- Experiments with controls

**Analysis in Software**

- Typical computer queries
  - Names of encodings
  - Relations between encodings
  - Variations of encodings
  - Quality of encoding conversions
- Core dumps, architecture and debugging
- Data flow analysis
- And points of conversion
- Known data injection
How to be a CSI (encoding Crime Scene Investigator)

Analysis In Software

- Architecture: The Web is complex
  - Clients, Browsers, mix technologies (DOM, java, css, php, active-x, applets, etc.), *ml standard+non-standard behaviors, OS dependencies, etc.
  - Servers, protocols are also mixed
  - Language, locale, encoding, are ambiguous
  - Negotiation tactics are unreliable
- But must be understood & kept up-to-date
  - See Web Internationalization Tutorial, et al.

Character Encoding Negotiation

Unix user
- html

Windows user
- GB2312
- HTML
- CSS
- JavaScript

Encoding Negotiation

- Data flow and points of (mis)conversion
  - Interfaces are common failure points
    - Between Legacy, new and 3rd party software
    - API, web services, protocols, devices
    - Data sources, sinks
  - Types of misconversion
    - Missing conversion Source ➽ Source
    - Incorrect conversion Source ➽ X
    - backwards conversion Target ➽ Source
    - double conversion (Source ➽ Target) ➽ Target

- Other contributors to conversion errors
  - Missing or incorrect encoding identifier
  - False detection
    - e.g. ASCII vs UTF-7
  - Text not like corpus used for detection statistics
  - Encoding name variations
    - “I didn’t know it was a crime. I just did what I was told!”
Encoding Name Variations

• E.g. ISO 8859-1 vs Windows-1252

<table>
<thead>
<tr>
<th>0-1F Control Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 F Control Codes</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>80-9F Control Codes in ISO 8859-1</td>
</tr>
</tbody>
</table>

Encoding Name Variations

• Several variations of big-5, shift-jis, etc.
  – E.g. Microsoft, Apple added chars to shift-jis
• Then there is big5-hkscs
  – Hong Kong Supplementary Character Set
• Application dependent names
  – Windows IE expects “KSC5601” or “Korean”, not cp949, windows-949
  – Firefox would expect euc-kr
• Sometimes the error is font not encoding

Encoding Name Variations

• Poor naming conventions in the industry
• Application dependent

<table>
<thead>
<tr>
<th>Language (L)</th>
<th>Traditional Encoding Name</th>
<th>ISO Encoding Name</th>
<th>EUC Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese Traditional</td>
<td>BIG5</td>
<td>GB18030</td>
<td>GB18030-P</td>
</tr>
<tr>
<td>Korean</td>
<td>EUC-KR</td>
<td>CP949</td>
<td>windows-949, windows-949</td>
</tr>
<tr>
<td>Chinese Simplified</td>
<td>GB2312</td>
<td>GB2312</td>
<td>GB2312</td>
</tr>
<tr>
<td>Western Europe</td>
<td>ISO-8850-1</td>
<td>CP1252</td>
<td>CP1252</td>
</tr>
<tr>
<td>Greek</td>
<td>ISO-8850-7</td>
<td>CP1252</td>
<td>CP1252</td>
</tr>
</tbody>
</table>
Encoding Conversions

- Correct conversion is not always obvious:
  - 0x5C is it a file separator “\” or currency “¥”?
  - (or Won, or other currency)
  - Half-width or Full-Width?
  - Sometimes you need different converters

Missing Character Conversions

- The family secret: Versioning
  - Unicode is an evolving standard
  - As characters are added the opportunity for improved conversions exist
  - “Which Unicode version is the converter for?”
  - If I convert data today how will it compare with the same data converted 3 years ago?
  - How long after death before rigor mortis sets in and when do the maggots come?

Breaking Multibyte Characters

- By not treating all bytes as one character
  - Don’t insert bytes in the middle
  - Don’t delete 1 byte of a multi-byte char.
  - Be careful with block boundaries
  - Don’t treat individual bytes as characters
  - E.g. uppercase a trailing byte
  - E.g. Treat trailing byte 0x5C as “\” in file pathnames
### Making Mojibake

**Splitting multi-byte characters**

<table>
<thead>
<tr>
<th>Byte type:</th>
<th>Bytes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>L 9 3 6 A T 9 7 8 B C A</td>
<td></td>
</tr>
</tbody>
</table>

**Inserting “a” (61) in second byte**

<table>
<thead>
<tr>
<th>Byte type:</th>
<th>Bytes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>L 9 6 3 T A 6 B C A</td>
<td></td>
</tr>
</tbody>
</table>

### Encoding Pathology and Forensics

- Piecing together who did what to whom
  - Knowing what was expected and what resulted, and comparing with typical patterns of failure, we can deduce what occurred
- Do all characters fail or certain ones?
- How do they fail? Wrong characters, Question marks, Blackboxes, etc.

### Encoding Forensics and Pathology

- **Question Marks**
  - Conversion to an encoding that doesn’t have the characters replaced by “?”.
- All non-Ascii characters are mojibake
  - Missing, wrong, backwards, or 2x conversion
    - Expected “ç”. Actual: “Äš”
      - Expected 0xE7 (ç). Actual: 0xC3 0xA7 (Ä§)
      - Note that U+00E7 in UTF-8, “ç” is 0xC3 0xA7
      - Conclusion: UTF-8 bytes displayed as ISO8859-1
  - Expected 0xE7 in UTF-8, “ç” is 0xC3 0xA7

- Only certain characters misconverted
  - Variant encoding, out of date conversion, data flow, incomplete font
  - E.g. Only euro, trademark, smart quotes, OE ligature,....
  - Used iso-8859-1 to utf-8 conversion instead of windows-1252 to utf-8
How to be a CSI (encoding Crime Scene Investigator)

**Encoding Error Patterns**

- Comparing conversion error patterns is like identifying the gun that fired a bullet by the shared striation marks
- It’s encoding DNA
- Often there are obvious candidate patterns
  - “Round up all the usual suspects”
  - A stoolie (er… tool) for testing conversions
    • Given an input string, try the usual and expected error conversions, See if any of the results match
  - Other tools: Encoding validator, byte to %hh

**Encoding Research**

- How many trucks stopped at Joes bar?
- Look for variants you might be incurring
  - [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets)
  - Conversions supported
    • See iconv, ICU, or doc for your converter
  - Check for font versions/updates

**Data Injection**

- Testing with live data feeds
  - like being on a stakeout- Hoping the criminal will come by while you wait
- Create a pseudo-feed or simulate data entry for “controlled experiments”.
  - Known, repeatable values simplify debugging.
  - Less threatening for debugging foreign languages
- Inject directly into subsystems to eliminate other components as “suspects”.

**Setting Traps for Criminals**

- Use validation routines at key points of data acceptance or conversion
  - Especially useful for UTF-8, which has illegal byte patterns.
  - Reference: [www.w3.org/International/questions/qa-forms-utf-8.html](http://www.w3.org/International/questions/qa-forms-utf-8.html)
Conclusions: How to be a CSI

- Gather forensic evidence
- Be objective, be thorough
- Understand the architecture, the data flow, the points of conversion
- Identify the potential patterns of failure

“Concentrate on what cannot lie. The evidence” - Gil Grissom