

Content-Type Attacks

ECE568 – Lecture 23
Courtney Gibson, P.Eng.
University of Toronto ECE

Outline

Content-Type Attacks

- Understanding the vectors
- PDF files
- Future Trends



Understanding
the Vectors

JPEG Files

- Reasonably simple file format, not popularly associated with a virus risk
- Multiple vulnerabilities have appeared over the years in libraries that parse even this simple format
- **2004:** Microsoft (Graphics Device Interface) GDI+
 - Hundreds of effected programs; arbitrary code execution possible
- **2010:** Microsoft Paint, Avaya Meeting Exchange
 - Arbitrary code execution possible

File Formats

- The format specifications for many popular document types (PDF, DOC, etc.) are now extremely complex
 - **PDF spec (v6):** 1,310 pages
 - **Microsoft Open XML spec:** 929 pages just to describe how Microsoft has interpreted the ISO/IEC 29500 standard that specifies the Open XML document format.

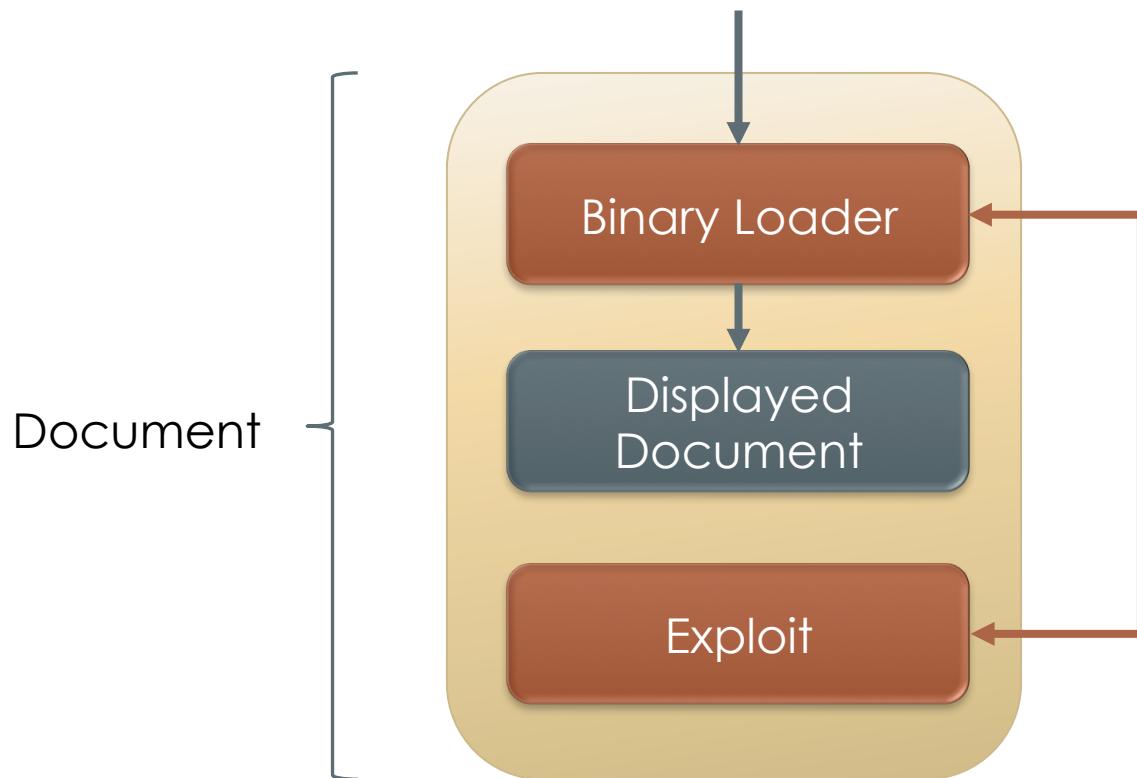
File Formats

- Even the simplest document requires thousands of lines of code to process

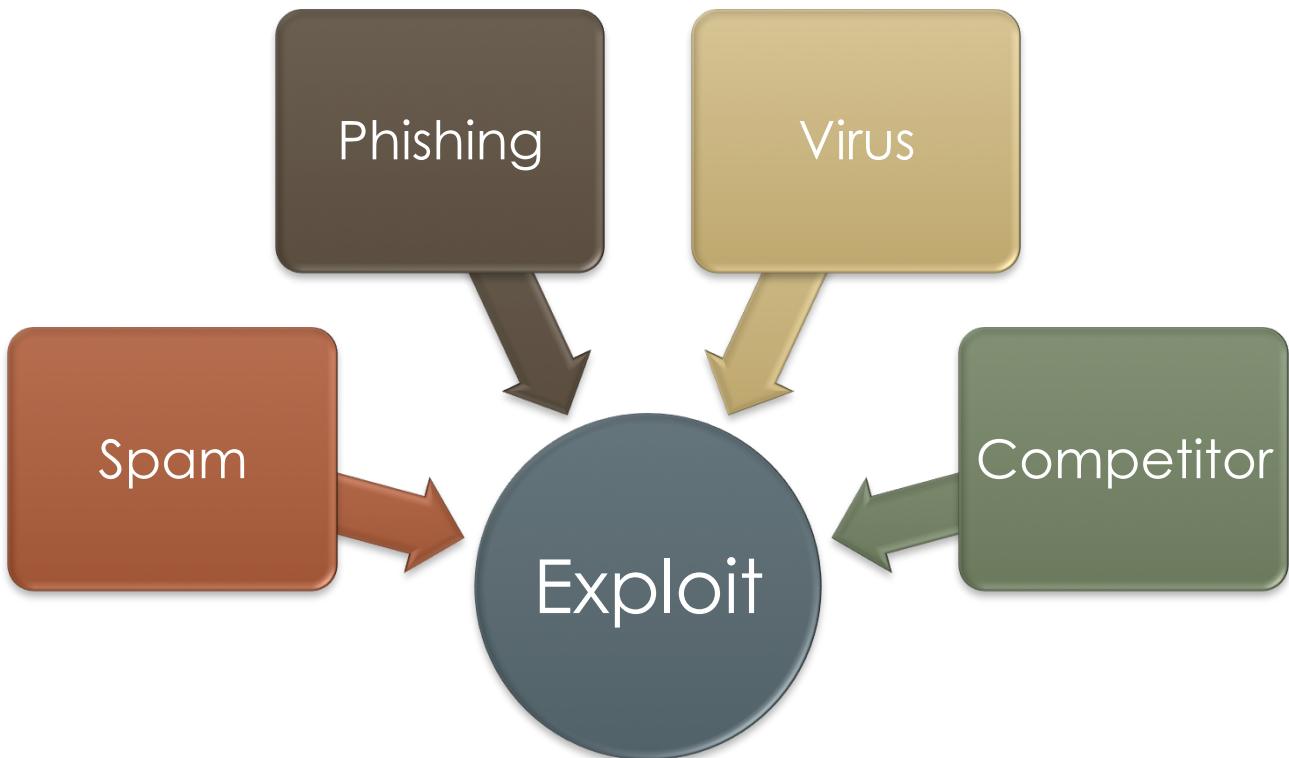
```
Hello world.
```

- **TXT:** 12 bytes
- **DOC:** 22,016 bytes
- **DOCX:** 21,917 bytes
- **PDF:** 12,535 bytes
- **PPTX:** 35,574 bytes
- **XLSX:** 27,437 bytes

Content-Type Attack

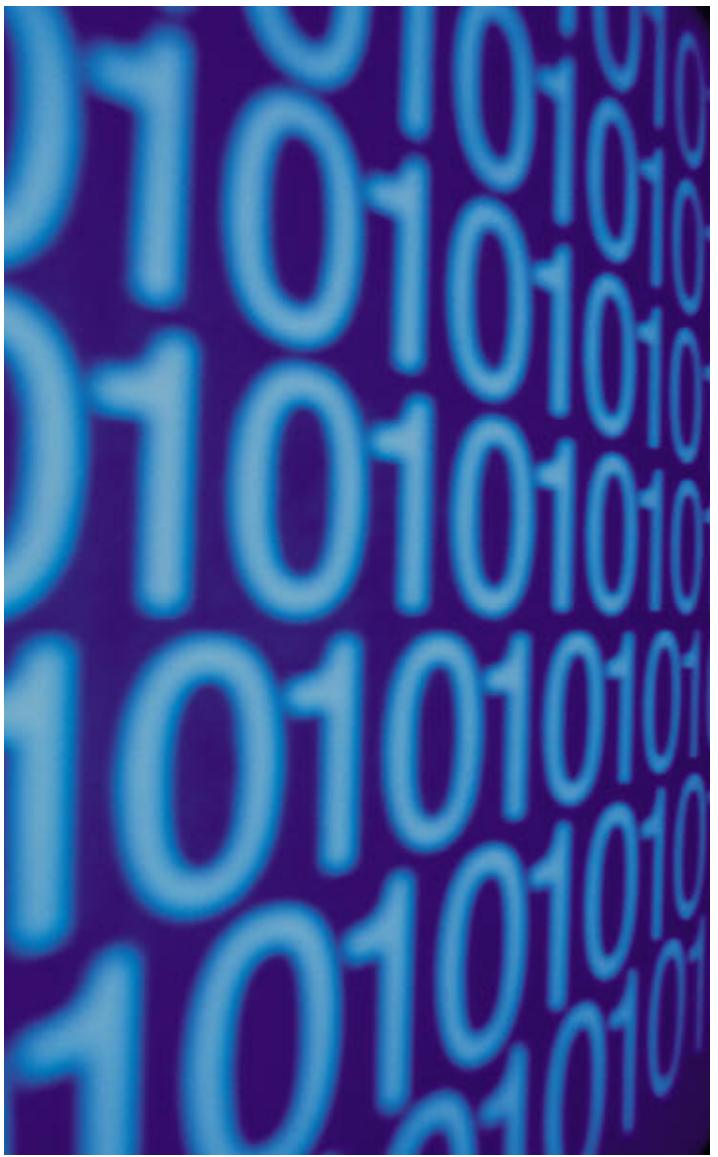


Attack Vector



PDF Files

- PDF files have been the most-targeted file format for attacks in recent years
- Likely the result of many proof-of-concept attacks available in the public domain



PDF Files

PDF File Format

- PDF Files start with a line describing the PDF language version:

%PDF-1.1

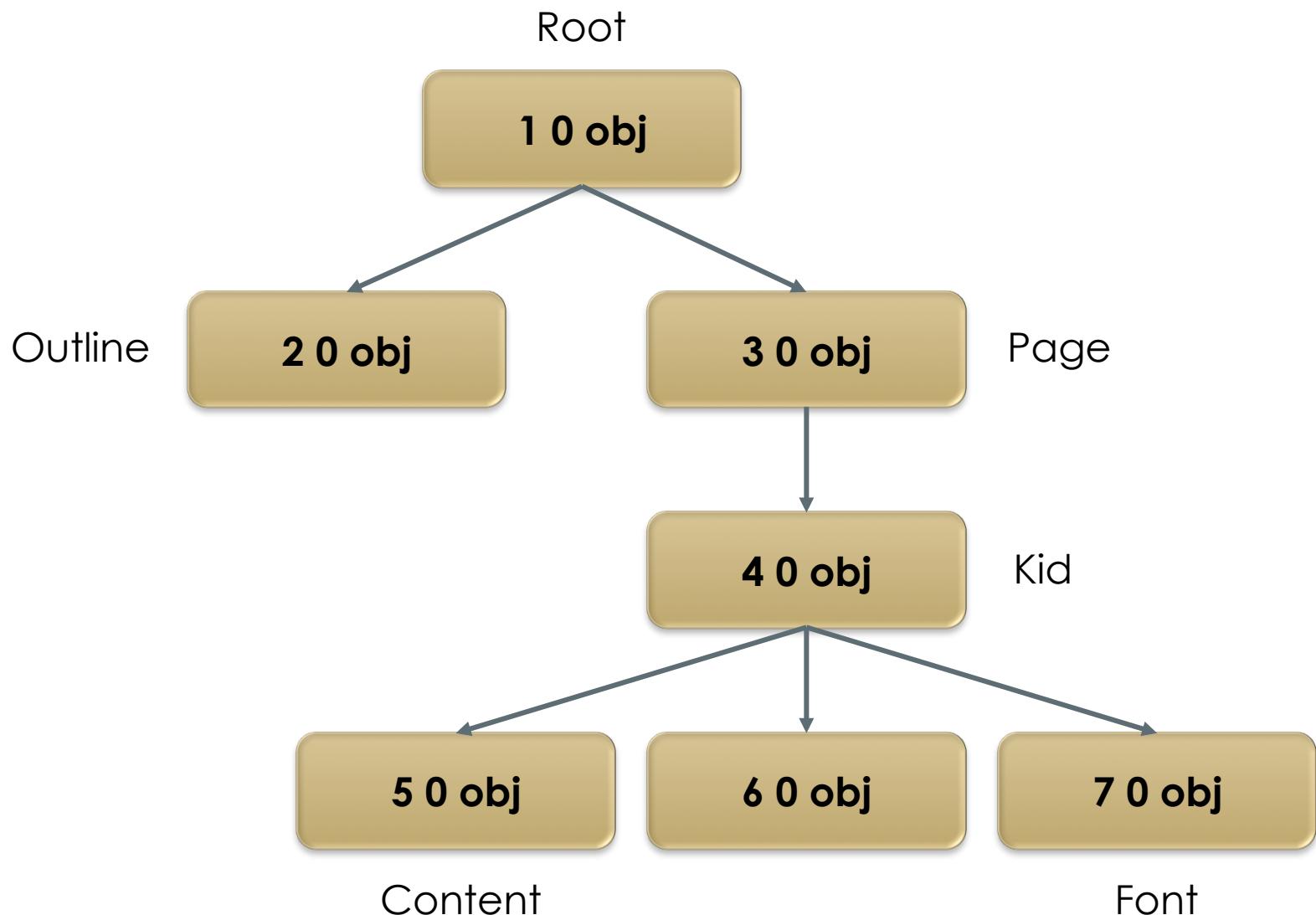
- Remainder of the file describes a hierarchy of **objects** that make up the document:

[index #] [version #] obj

<<

[content]

- See sample PDF file on course website



PDF File Format

- Data is uncompressed by default:

```
2 0 obj
<< >> stream
BT/default 99 Tf 1 0 0 1 1 715 Tm
(Hello World!) Tj ET
endstream
endobj
```

- ...but is often compressed to obfuscate the contents; checking PDF files for malicious code is hard.
- Didier Stevens has created a handy suite of tools for analyzing PDF files (`pdf-parser`):

<http://blog.didierstevens.com/programs/pdf-tools/>

PDFs and JavaScript

- Objects can contain malicious JavaScript
- The objects can be automatically called when the document opens:

```
7 0 obj
<<
/Type /Action
/S /JavaScript
/JS (app.alert('Hello world'));
>>
endobj
```

Detecting Attacks

- The PDF spec has a great deal of flexibility: many opportunities for attackers to obfuscate their code:

```
/OpenAction 42 0 R  
/#4FpenAction 42 0 R  
/#4Fpen#41ction 42 0 R  
/#4f#70#65#63#41#63tio#63 42 0 R  
...
```

- Exploits can be triggered through a number of other means (annotations, forms, etc.)
 - Contents of those objects may be compressed
 - See the MetaSploit project for example exploits

Other Attacks

- One solution is just to disable JavaScript
 - Breaks a number of PDF forms
 - Assumes that JavaScript is the only attack vector
- PDF parsing engines are complicated, and vulnerabilities are regularly discovered
- **Example:** CVE-2008-2992
 - Overflow vulnerability in implementation of printf
 - Could be exploited to allow arbitrary code to be executed if a very large number was printed
 - Many proof-of-concept scripts were created

Prevention Options

- Turning off library features breaks standards
- Virus scanners are reactive, not proactive
- A more generalized approach is needed
 - Recent versions of Adobe Reader for Windows now all ship with DEP (non-executable stack/heap) turned on by default
 - Is this sufficient?



Future Trends

Future Trends: Dorifel

- The Dorifel virus started spreading in late 2012
 - Downloaded onto many systems already infected with another piece of malware designed to steal banking credentials (Citadel)
 - Launches on startup and takes some steps to avoid detection (terminates itself if it sees “taskmgr.exe”)
 - Uses a “right-to-left” Unicode vulnerability to hide itself from casual inspection / trick users

Right-to-Left

- Unicode is replacing ASCII as the standard for encoding text using
- Unicode character (U+202E) is defined as the Right-to-Left Override (RLO)
 - Switches the direction that text is displayed in
 - Increasing use in a variety of attacks:

Resume – John Al[RLO]cod.exe

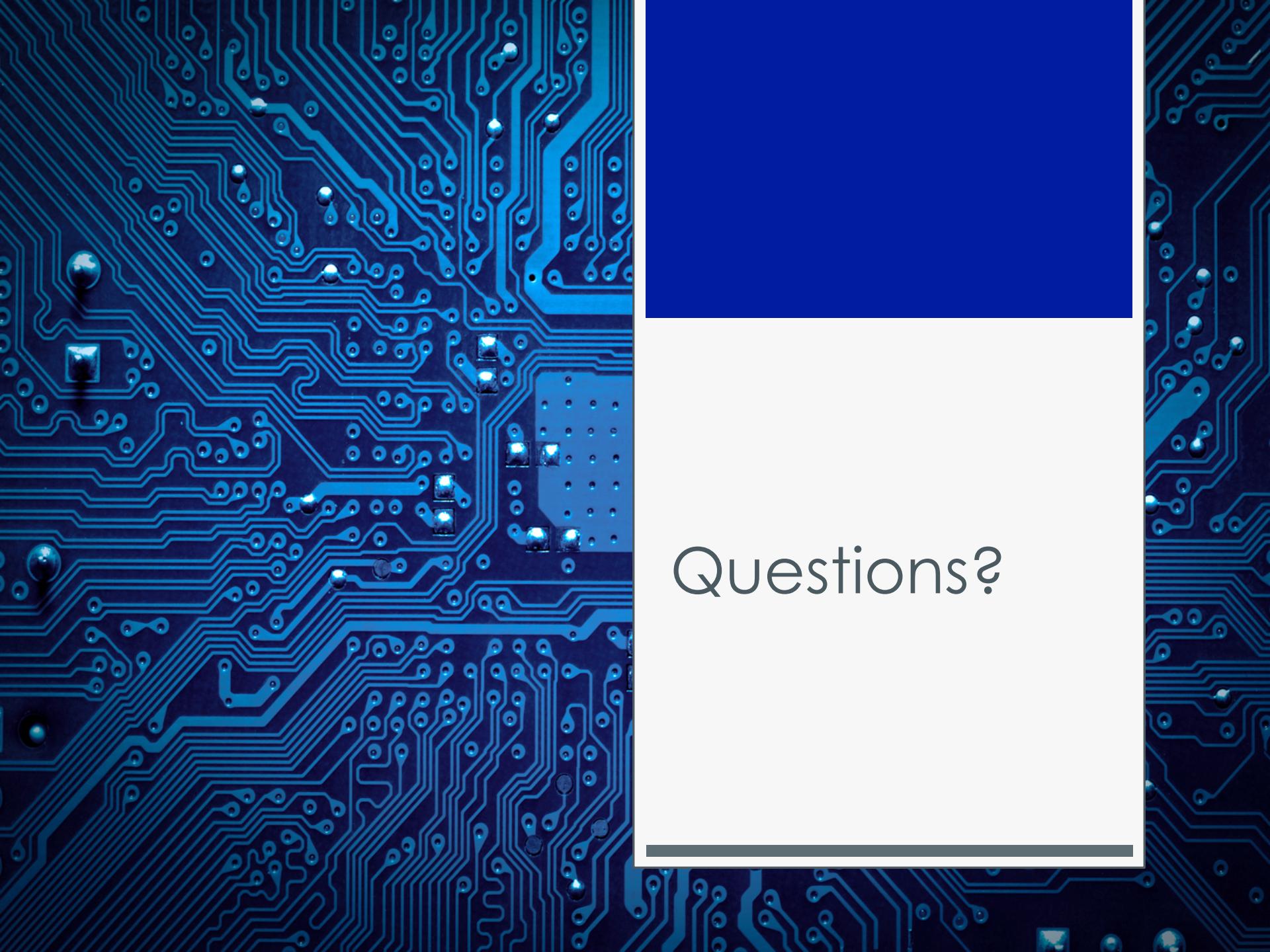
displays as: **Resume – John Alexe.doc**

www.payp[RLO]moc.la

displays as: **www.paypal.com**

Future Trends: Dorifel

- Scans the system every 5 seconds for Excel and Word documents
- Reads in the document, encrypts it, deletes the original and then replaces it with an executable that, when run:
 - Infects the host machine with Dorifel
 - Decrypts the document to a temporary location
 - Opens the document
- The executable has the original name, but the “.doc” or “.xls” extension overwritten with a Right-to-Left hack (“[R\0]cod.scr”)



Questions?