Security in Office 2013

Tom Gallagher
Principal Security Test Lead
Office Trustworthy Computing
Microsoft Corporation

Agenda

- Previous Work: Office 2010 Security
- Office 2013 New/Enhanced Features
- Engineering Improvements to the codebase
- Internal Security Testing Efforts

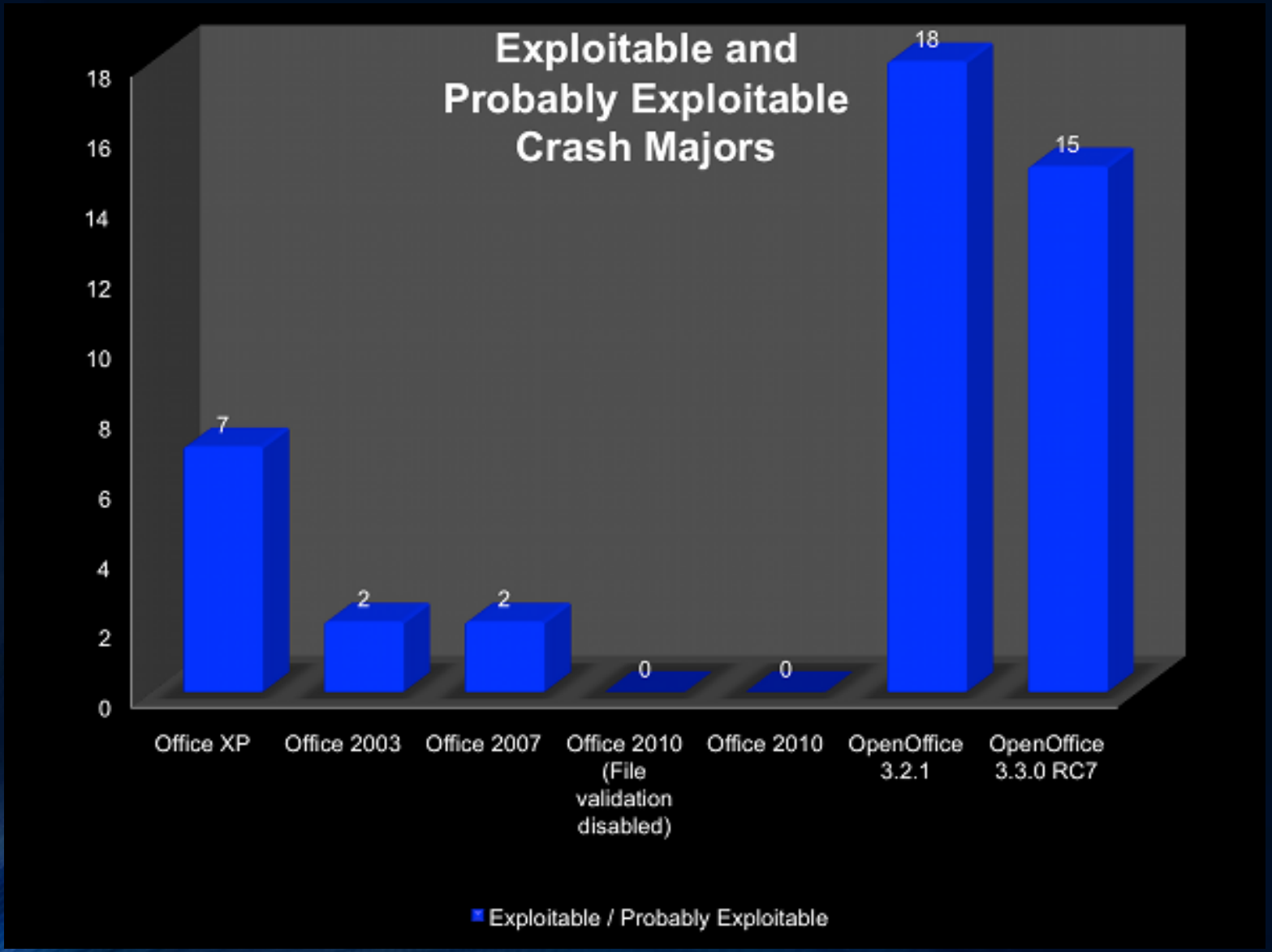
Office 2010 Security

- Office 2010 significantly raised the bar with both security features and codebase improvements based on internal testing.
- Key security features include:
  - Office File Validation – Schema based validation of binary file formats
  - ProtectedView – Sandboxed viewing of documents from untrusted sources.
  - File Block – Option to prevent opening files based on the file format.
- Investment in internal fuzzing efforts.
  - Distributed Fuzzing Framework

Demo: Office 2010 Security Mitigations

Office 2010 Engineering/Codebase Improvement Results

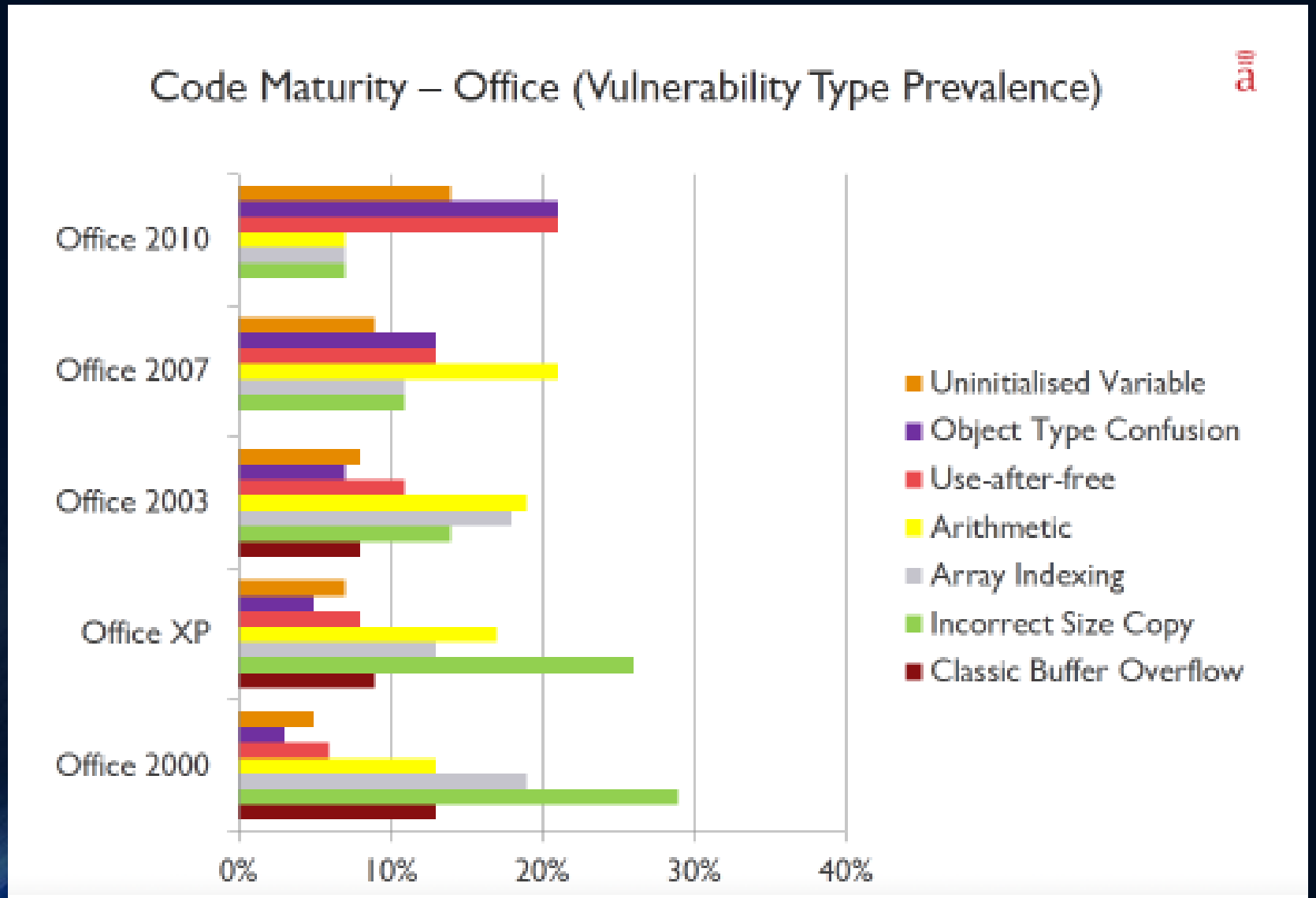
CERT/CC Investigation: "Office shootout"



Bugs are deeper

- Secunia performed analysis of Microsoft Office security patches to determine the root cause of each.
- The work performed by Carsten Eiram of Secunia is detailed and breaks memory corruption bugs into more specific categories.
- His results are consistent with our internal investigation. Bugs that are easier to find and exploit are consistently becoming rarer.

Secunia's Analysis of Office Patches



http://rvasec.com/slides/2012/3_eiram_code_maturity_rvasec_2012.pptx

Office 2013

Office 2013 Security

- Improvements in Office 2010 helped raise the bar.
- Attackers continue to innovate with bug finding and attack techniques
- In Office 2013 we've invested in the following areas –
  - New/Enhanced Security Features
  - Engineering Improvements to the codebase
  - Internal Security Testing Efforts

New/Improved Security Features

- Encryption Improvements
  - SHA512 by default
  - AES 256-bit encryption
  - 100,000 iteration key derivation function
  - "Office 2013 sets a new standard in document encryption, pretty much taking brute force out of the question."
    - -- Elcomsoft (http://blog.crackpassword.com/2012/09/elcomsoft-breaks-into-ms-office-2013)

- Digital Signatures
  - MD5 Collisions are feasible and use of MD5 is not a security best practice.
  - By default, Office now warns if a document is protected using MD5.
  - Hashing algorithms to warn on are configurable through policy.

- Click To Run

Click To Run

- Technology that allows Office to run locally in a virtualized environment.
- Used for customers subscribing to Office 365.
- By default, updates are automatically detected, downloaded, and installed.
- Group policy still used to help control the environment.

Security Engineering Improvements

Three areas of investment

1. Investments to directly improve the codebase
2. Test efforts to identify areas that need to be improved
3. Leverage operating system improvements

Codebase Improvements

- Enhancements to rules for automated code review tools for both managed and unmanaged code.
  - Examples: Integer overflows, XML attacks, uninitialized variables checks
- Central lockdown of hosted MSHTML (WebOC).
  - Helps protect against leveraging Office as a vector for non-Office vulnerabilities. Some of these attacker were presented by TT Tsai and Nanika at SyScan Singapore 2010 - *Office Is Still Yummy - How To Defeat Memory Protections In Office Document Exploitation*

Leveraging Platform Improvements

- 3 architectures are supported –
  - X86, X64, ARM
  - Attacker will need to write shell code specific to these platforms.
  - No legacy add-ins/code exists on ARM.
  - Since ARM legacy doesn't exist, we are able turn on security mitigations that normally could impact backward compatibility.

Leveraging Platform Improvements

- Windows 7 and later is supported.
  - No support for Windows XP allows using APIs with security enhancements.
- Insecure Library Loading (aka Binary Planting Preloading/hijacking)
  - If KB2533623, we have central code to use LoadLibrary with a secure search order.
- Use of the system heap
  - Office no longer uses it's own memory manager. The hardened system heap is used instead.
- Windows 8 Isolation
  - On Windows 8 the system provided Application Container APIs are leveraged.

Windows 8 Application Container

- Windows 8 introduces a new isolation model.
- Applications running in an Application Container are isolated and can make app capability declarations to gain access to specific capabilities.
- This allows Office ProtectedView to be more restrictive.
  - Example: Previously ProtectedView had access to network resources. In Windows8, ProtectedView doesn't have the network capabilities (proximity, internetClient, internetClientServer, and privateNetworkClientServer).
- More detail on App capabilities in Windows 8 - http://msdn.microsoft.com/en-us/library/windows/apps/hh464936.aspx

Testing Improvements

- For web applications, focus is on automatic XSS detection.
  - Automatically identify attack surface and test for persistent, reflected, and DOM based XSS.
  - DOM based is a challenge so JavaScript taint analysis is performed.
- For the client, focus is on file fuzzing.
  - Depth
  - Testing the tools and test environment itself

3 Fuzzing Depth Improvements

1. Leveraging Office File Validation for testing.
2. Gridlock Fuzzer
3. Distributed SAGE Fuzzer

Fuzzing Depth: Office File Validation

- The Office File Validation feature is based on an XML file that contains a description of the file format and limits.
- We repurpose that data programmatically to create C# objects that can manipulated and saved as an Office document.
- This allows more advanced and format specific fuzzing manipulations.
- Allows creating a systematic test plan for testing the file format.

Fuzzing Depth: Gridlock

- We use the tools presented by Dustin Duran, Matt Miller, David Weston and presented at Cansecwest 2011
  - http://cansecwest.com/csw11/Metrics%20for%20Targeted%20Fuzzing%20-%20Duran,%20Miller%20&%20Weston.pptx
- Allows very targeted fuzzing and building smarter template collections.
- Fast and less painful regression testing that targets code changed.

Dynamic trace	File Offset	Tainted Function
d.bmp.trace	d.bmp:7777	Func3
a.bmp.trace	a.bmp:4444	Func1
a.bmp.trace	a.bmp:8888	Func4
c.bmp.trace	c.bmp:6666	Func2
…	…	…

Fuzzing Depth: SAGE

- Scalable, Automated, Guided Execution (SAGE) is a white box fuzzer created by Microsoft Research
- **SAGE: Whitebox Fuzzing for Security Testing** *Patrice Godefroid & Michael Y. Levin & David Molnar* - http://research.microsoft.com/en-us/um/people/pg/public_psfiles/ndss2008.pdf
- Constraint solver is good at identifying multiple input manipulations that cause exploitable crashes.
- Dev comment in a bug when fixing it –
  - "I think the fuzzers are starting to become sentient. We must crush them before it is too late."
  - The input required multiple small manipulations to hit the vulnerable code path.
- Parts of SAGE are now distributed in our fuzzing environment.

Improved robustness of the test tools

- Fuzzers rarely validate they are working correctly unless you find a bug.
- How to FAIL at Fuzzing, Ben Nagy (Kiwicon 2010)
- While we have the defender's dilemma, we have some advantages.
  - Source, symbols, and the ability to change Office source code to help testing.

Validating fuzzers

- We want to ensure:
  - Code we expect to fuzz is getting exercised
  - Fuzzing manipulations are smart enough to bypass validation
- Through the use of runtime instrumentation and symbols this is achieved cross-build.
- Halting runs based on failing certain metrics.
  - Example: Intended parser is not invoked a certain percentage of the time.

Screen shot of fuzzing instrumentation

Code Hits

For Application	**ID: 503**	**File Load**	**99.5%**
For Run	ID: 8572	HyphenatorParse	90.4%

Speed

- Fuzzing is often measured by the number of iterations.
- Format aware manipulations are more important.
- However, speed should not be overlooked.
- By modifying and monitoring the Office code, we can create more comprehensive and more efficient fuzz runs.
  - Example: Close application immediately after parsing is complete

Demo of fuzz speed comparison

Office 2013 RTM Fuzzing Results

- 2100+ fuzzing bugs fixed. More than any previous release of Office.

- Many issues are fixed without determining exploitability.
  - Easier to fix a crash than determine its security implications

- Fixed count is a result of being more aggressive in fixing benign crashes and our work to become more efficient and deep with fuzzers.

Thanks…

This presentation represents the work of many. Special thanks to the Office Trustworthy Computing team.

Summary

- Office 2013 continues to raise the security bar.
- Office 2013 helps customers work securely by providing security features and protection against attack when performing actions unrelated to security.
- Engineering improvements have allowed finding and fixing more than 2100 bugs to improve security and robustness.

Questions?