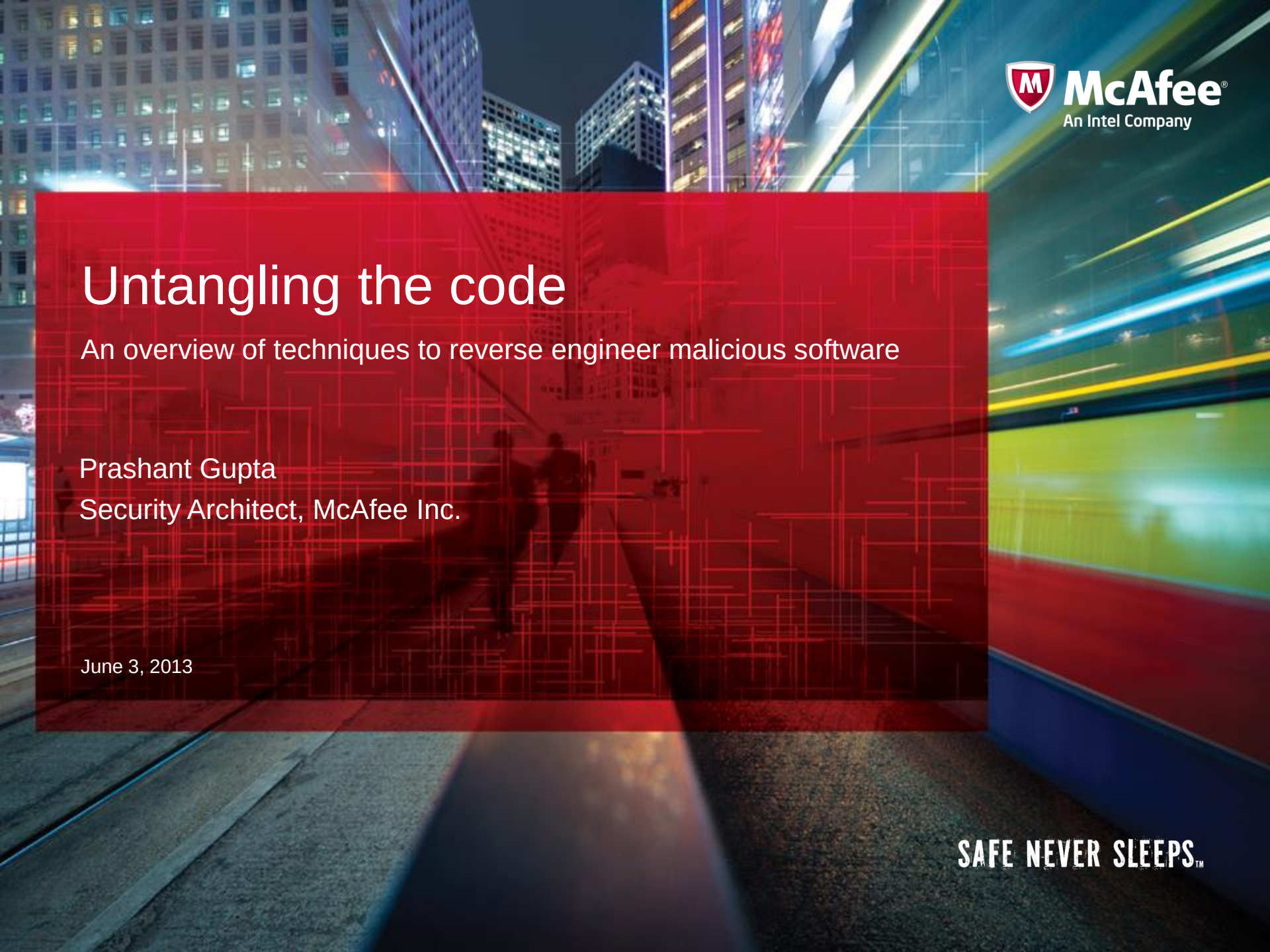


Untangling the code

An overview of techniques to reverse engineer malicious software

Prashant Gupta
Security Architect, McAfee Inc.

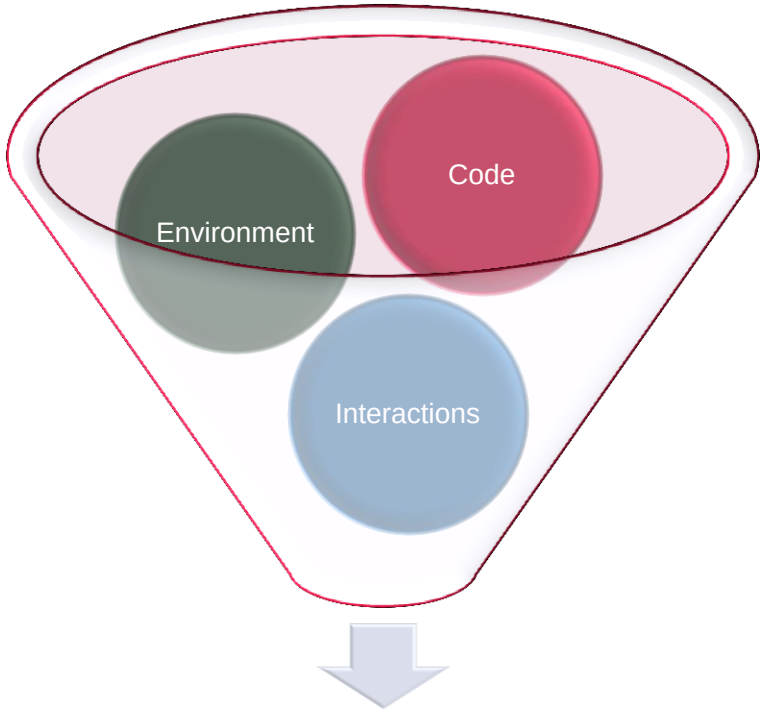
June 3, 2013

SAFE NEVER SLEEPS™

Abstract

Reverse engineering and analysis of binary code has been seen as a black art that requires immense commitment, large degree of experience and intuition to be fruitful. This may have been true a couple of decades ago but with the recent focus on reverse code engineering by security vendors and prolific malicious actors alike this field has progressed significantly. In this talk I will present some of the techniques employed during reversing of unknown binaries when vetting them for malicious traits. The talk would also cover how program analysis can help identify possibly suspicious traits in code and how reverse engineering and program analysis techniques used for code are also relevant for identifying potentially suspicious data when analysing document formats.

Reverse Code Engineering: What?



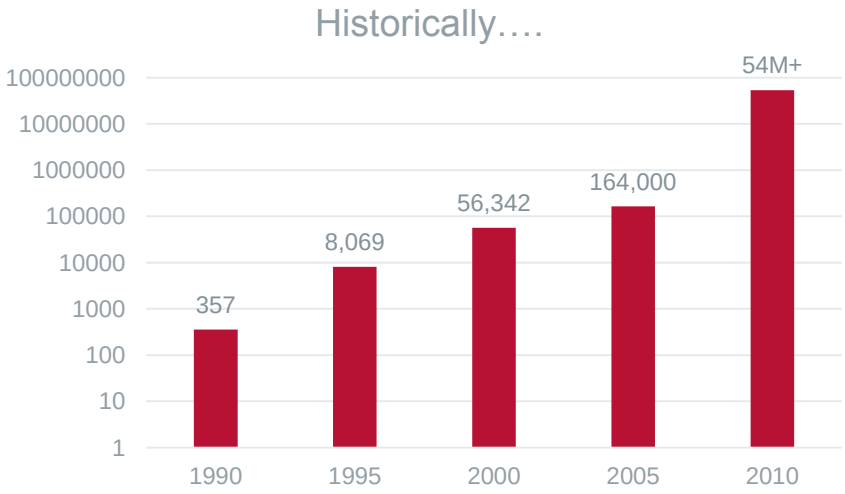
Better Understanding

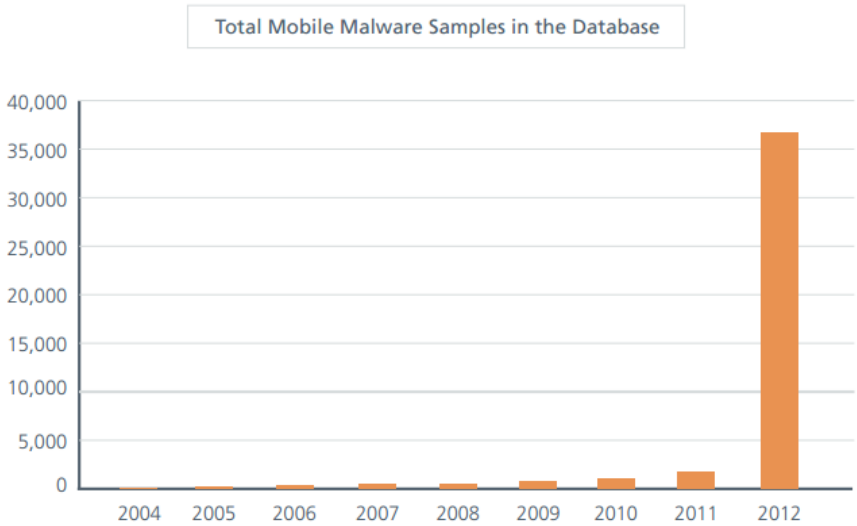
- Scanners/Fingerprinting
- Decoders/Decryptors
- Unpackers
- Behaviour Analysers
- Code reversers
- …
- …

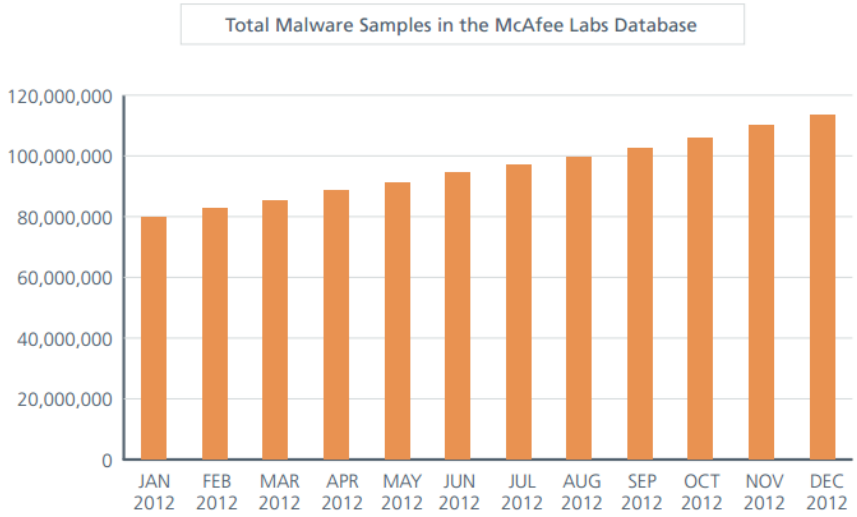
Reverse Code Engineering: Why?

- Improve understanding where source code is not available
- Audit, review or forensic investigation of software systems
- Identifying intellectual property licensing breach
- Malware research & defence

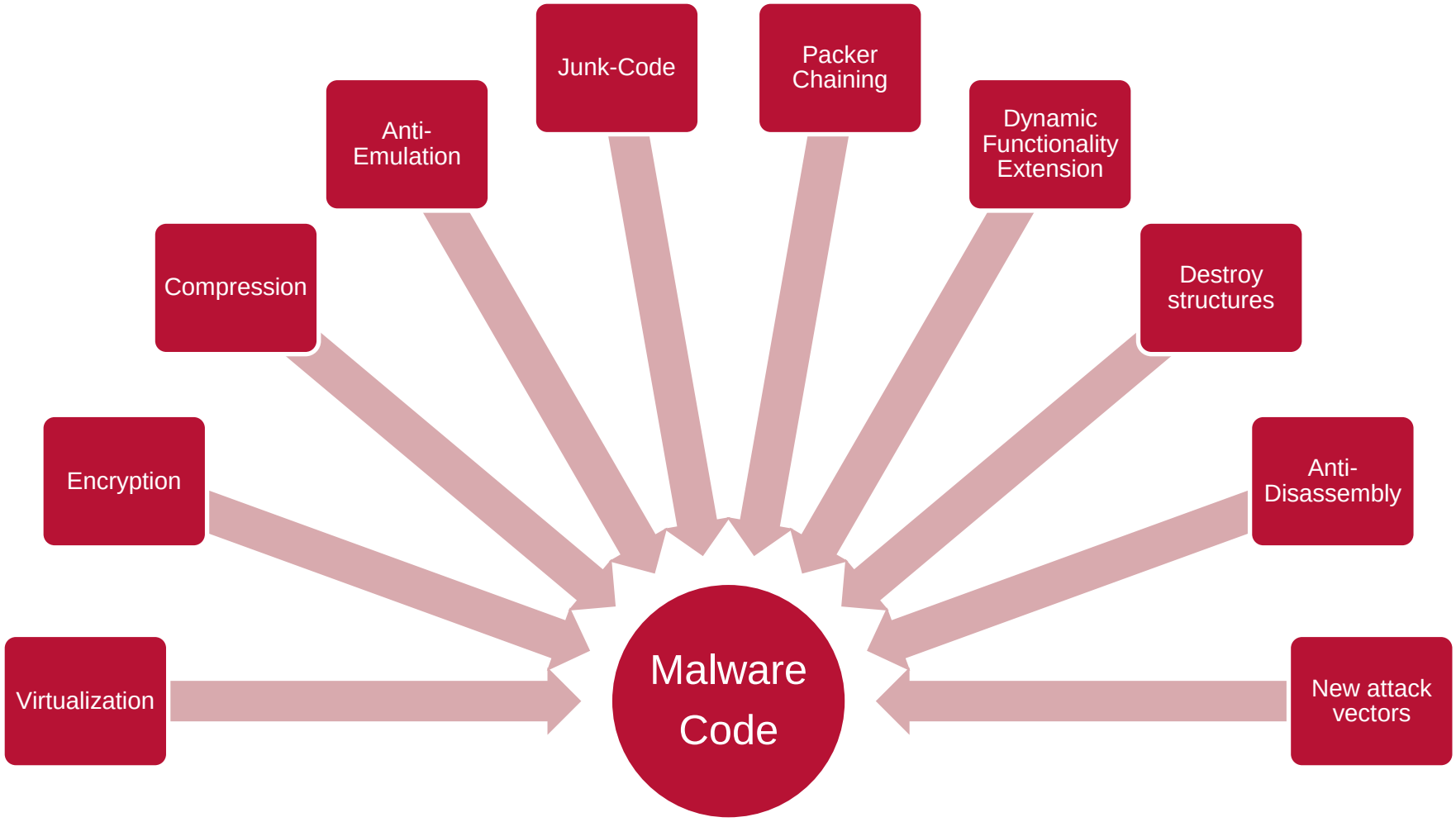
Malware prevalence







Malware prevalence: Why?



Reversing Code: Static Analysis

- Automated Signature Search
  - Text search
  - Binary signatures
  - Known blobs, tables and data structure search
- Decoding/decrypting code and payload
  - Identifying and decoding encoded payloads
  - From XORs to Purpose Built Custom algorithms (e.g. usage pseudo-random)
- Code Block identification
  - Compiler and Library recognition
  - Fingerprinting known obfuscation techniques
  - Dealing with compiler optimizations
  - Semantic code similarity identification

Reversing Code: Static Analysis

- Decompiling
  - Identifying boiler-plate code
  - Make complex code easier to understand (e.g. algorithms)
  - Many issues, after so many years still in it's infancy.

- Packer identification and Un-Packing
  - Identify if binary is a setup, built using a binder/packer/cryptor
  - Unpack using custom, generic or standard algorithms.

- Artefact extraction and logging
  - Structural anomalies
  - Known code patterns (e.g. peculiar function chaining)
  - Junk-code or no-op code search

Reversing Code: Dynamic Analysis

- Virtualization/Sandboxing and Debugging
  - Control malware exposure and it's ability to leave irreversible changes.
  - Debugging to guide execution flows (bypass exceptions)
  - Providing stimulus to force malware to exhibit behaviour
  - Active and passive analysis platforms.

- Behaviour logging
  - Automated behaviour trace logging
  - Identify communication mechanism
  - Identify persistence mechanism

- Post/Part execution memory dumps
  - Generate memory dumps for detailed static analysis
  - In cases where unpacking is not possible/feasible
  - In cases where in-memory data structures need analysis
  - Runtime unpacking

Reversing Code: Machine Correlation

- Correlation for high level behaviour inference
  - correlation of artefacts extracted from automated analysis

- Automated Classification
  - Behavioural trait association
  - Static code relationships
  - Malware family

Reversing Code: Manual Investigation

- Separating the wheat from the chaff
  - Identifying suspicious code
  - Evaluating known patterns and guiding analysis process
  - Making decisions where automation could only provide approximations.

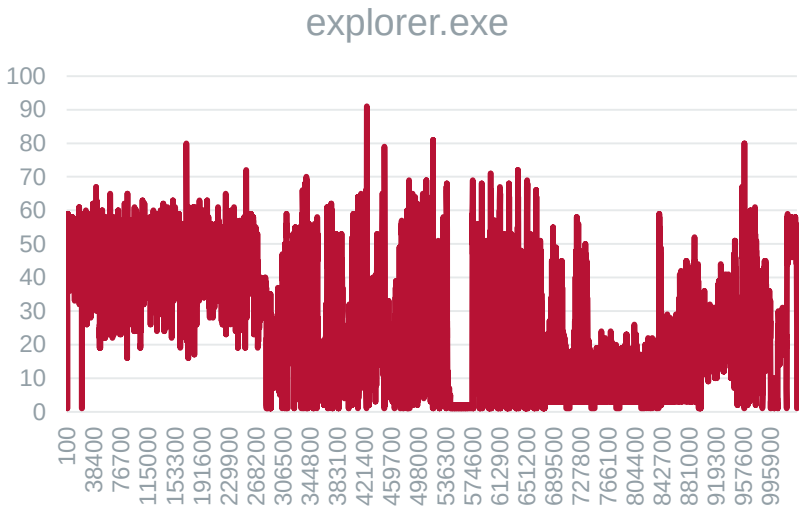
- Optimizing program code analysis
  - Correlating artefacts from static and behavioural analysis
  - Building new algorithms
  - Fixing problems faced by automation (exceptions, resource constraints, etc.)
  - Manual unpacking/decoding

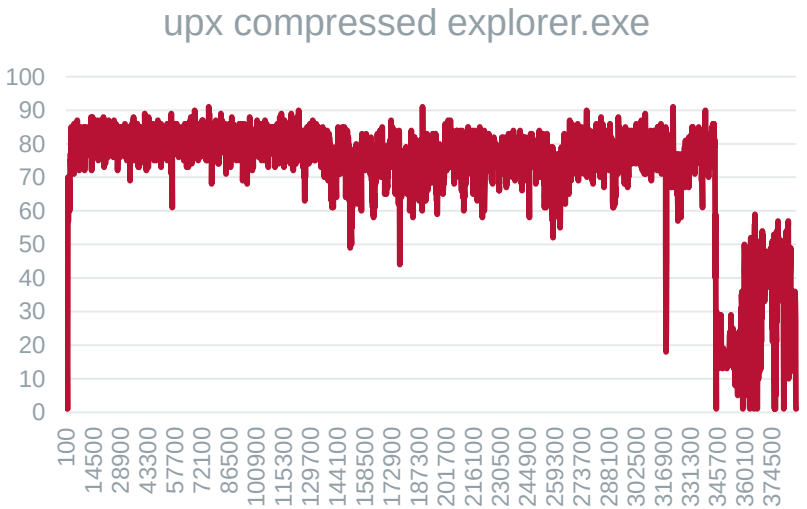
- Heuristic development
  - Adding new feature extraction methods
  - New correlation policies
  - New subsystem development for analysis and detection improvements

Reverse Engineering documents

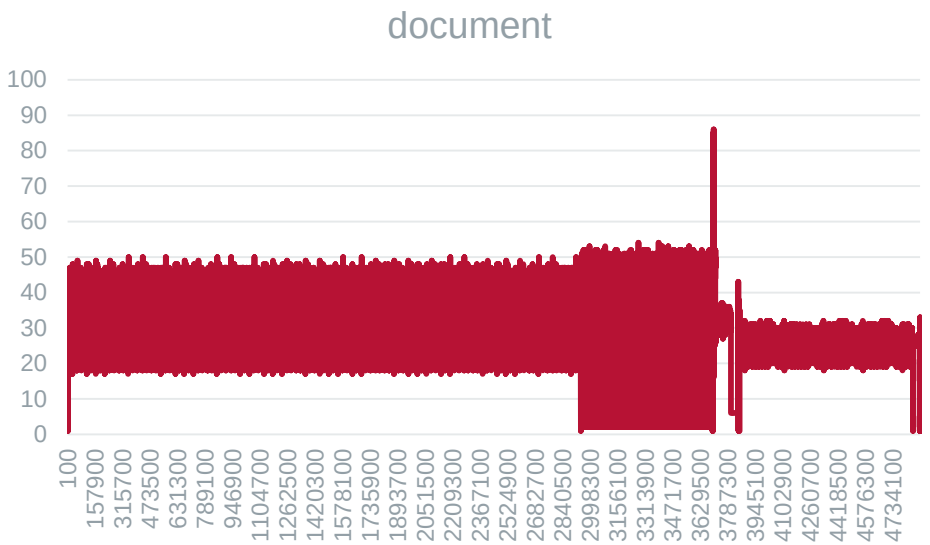
	Binary Code	Binary Documents	Analysis
Not human readable	Yes	Yes	RE tools and environment
Multitude of environments	Many execution environments including VMs	Documents are generally platform agnostic.	Heuristic analysis systems can be shared when analysing artefact correlation.
Can exploit vulnerabilities	Yes, but not always needed.	Yes, generally in document editor/reader but sometimes in OS	Dynamic analysis techniques can be used
Internal formats can be obfuscated	Yes	Yes	Detecting encoded payloads. Identifying presence of obfuscation.
Executable Code	Yes	No	Signature searches and payload analysis techniques.

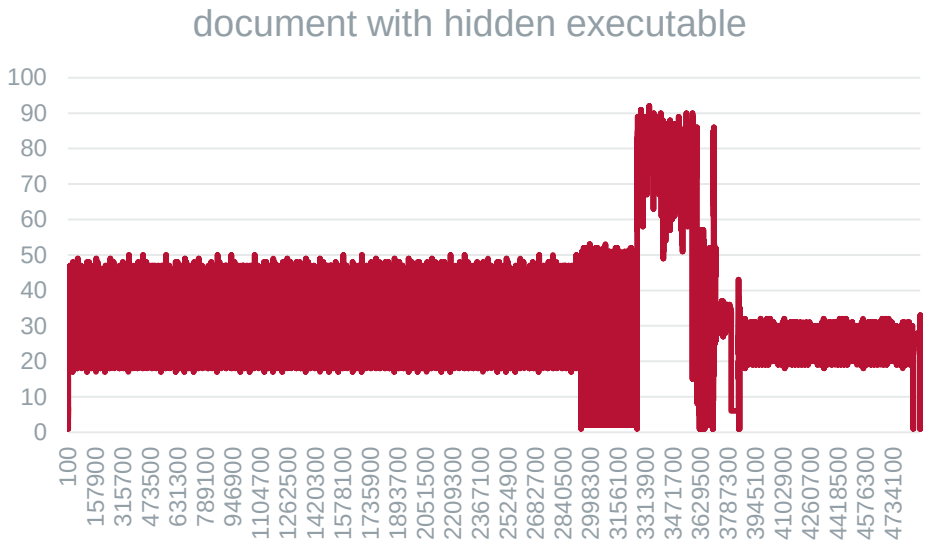
An example technique…





An example technique…





Open source toolsets

SysAnalyzer

- Automated malcode analysis system (not a sandbox!)

Malcode Analyst Pack

- suite of tools useful for malcode analysts

VirtualBox

- x86 and AMD64/Intel64 virtualization product

BeaEngine

- disassembler library x86 x86-64 (IA32 and Intel64)

Libemu

- x86 shellcode emulation

Databases/Tools

RE-Google IDA plugin

Queries Google Code for information about the functions contained in a disassembled binary

ClamAV

Open source (GPL) antivirus engine

Malware lookup services

VirusTotal, The Malware Hash Registry

ThreatExpert, McAfee SiteAdvisor

Utilities and Assessment Tools

McAfee Free Tools, Collaborative RCE Tool Library

Extensible analysis frameworks

McAfee
An Intel Company

Cuckoo Sandbox

- Modular malware analysis system

Zero Wine Malware Analysis Tool

- Research project to dynamically analyse the behaviour of malware

Malheur

- Automatic analysis of malware behaviour

Radare

- Open source tools to disasm, debug, analyse, manipulate binary files

Prashant Gupta
Security Architect, McAfee Inc.
@PrashantGupta

References

IMPORTANT: These are 3rd party websites so please review what you download for malware/suspicious content before use.

1. Cuckoo Sandbox: http://www.cuckoosandbox.org/
2. Zero Wine Malware Analysis Tool: http://zerowine.sourceforge.net/
3. Malheur: http://www.mlsec.org/malheur/
4. Radare: http://www.radare.org/
5. Interactive Disassembler Pro: http://www.hex-rays.com/
6. REGoogle: http://regoogle.carnivore.it/
7. ClamAV: http://www.clamav.net/
8. SysAnalyzer: https://github.com/dzzie/SysAnalyzer
9. Example technique from - Detecting exploits in electronic objects, Alexander Shipp: http://www.google.com/patents/US20080134333

References

IMPORTANT: These are 3rd party websites so please review what you download for malware/suspicious content before use.

10. Malcode Analyst Pack: https://github.com/dzzie/MAP
11. VirusTotal: http://www.virustotal.com/
12. The Malware Hash Registry: http://www.team-cymru.org/Services/MHR/
13. ThreatExpert: http://www.threatexpert.com
14. McAfee SiteAdvisor: http://www.siteadvisor.com/
15. McAfee Free Tools: http://www.mcafee.com/us/downloads/free-tools/
16. Collaborative RCE Tool Library: http://www.woodmann.com/collaborative/tools/index.php/Category:RCE_Tools
17. McAfee Threats Report Q4 2012: http://www.mcafee.com/uk/resources/reports/rp-quarterly-threat-q4-2012.pdf