RSA CONFERENCE 2014
FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.
Capitalizing on
Collective Intelligence

Writing Secure Software is hard, but at least add mitigations!

SESSION ID: ASEC-F02

Simon Roses Femerling

CEO
VULNEX
@simonroses

ME?

- Simon Roses Femerling
  - Founder & CEO, VULNEX www.vulnex.com
  - Blog: www.simonroses.com
  - Twitter: @simonroses
  - Former Microsoft, PwC, @Stake
  - DARPA Cyber Fast Track award on software security project
  - Black Hat, RSA, OWASP, SOURCE, AppSec, DeepSec, MSFT TECHNET

BIG THANKS!

- DARPA Cyber Fast Track (CFT)
- Mudge
- The fine folks at BIT SYSTEMS

TALK OBJECTIVES

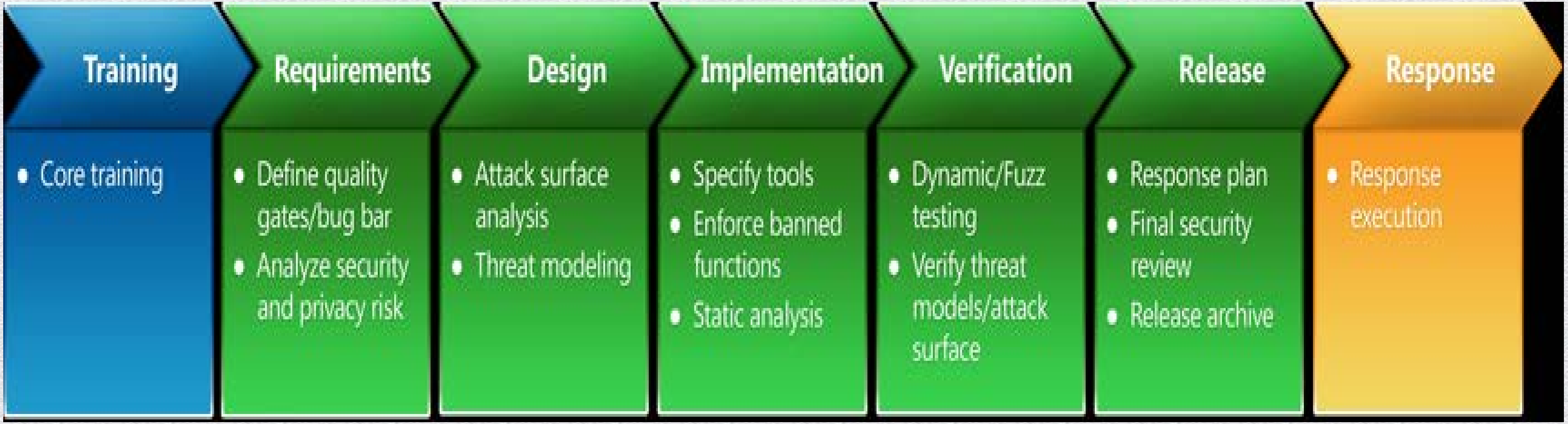
- Secure development
- Verify software security posture

AGENDA

1. Secure Development
2. Security Mitigations
3. BinSecSweeper
4. Case Studies
5. Conclusions

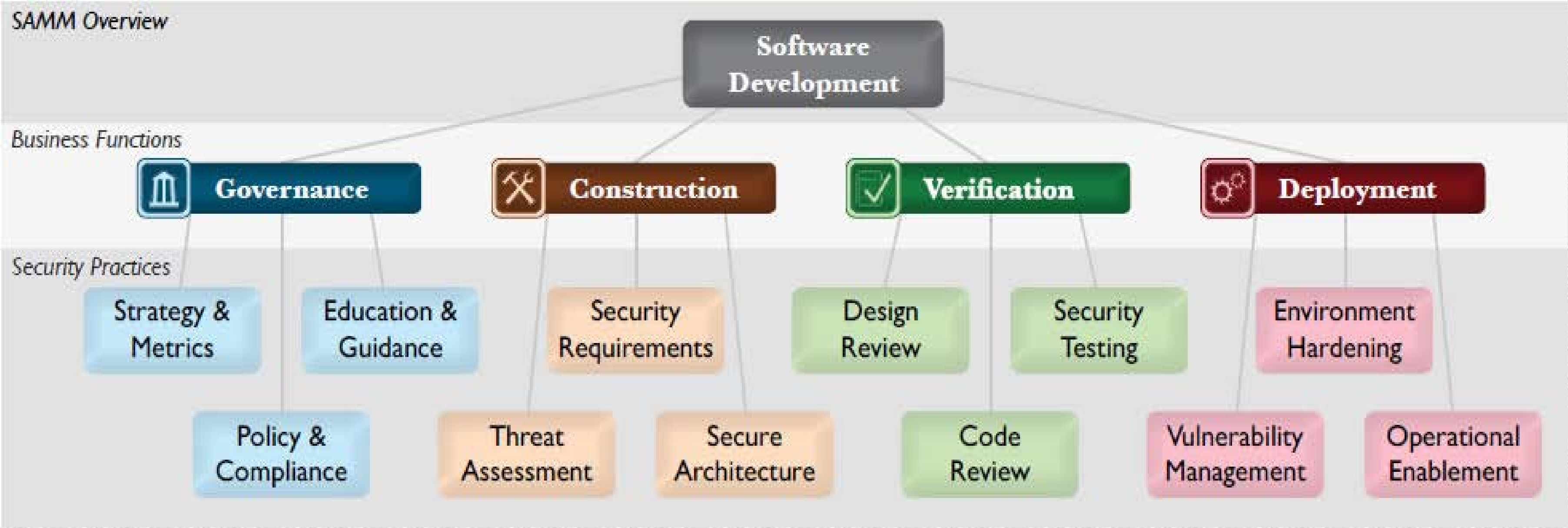
1. Secure Development

1. MICROSOFT SDL



Training	Requirements	Design	Implementation	Verification	Release	Response
• Core training	• Define quality gates/bug bar • Analyze security and privacy risk	• Attack surface analysis • Threat modeling	• Specify tools • Enforce banned functions • Static analysis	• Dynamic/Fuzz testing • Verify threat models/attack surface	• Response plan • Final security review • Release archive	• Response execution

1. OPENSAMM


SAMM Overview
Software Development
Business Functions
Governance
Construction
Verification
Deployment
Security Practices
Strategy & Metrics
Education & Guidance
Policy & Compliance
Security Requirements
Threat Assessment
Secure Architecture
Design Review
Security Testing
Code Review
Environment Hardening
Vulnerability Management
Operational Enablement

1. IT'S ABOUT SAVING MONEY!

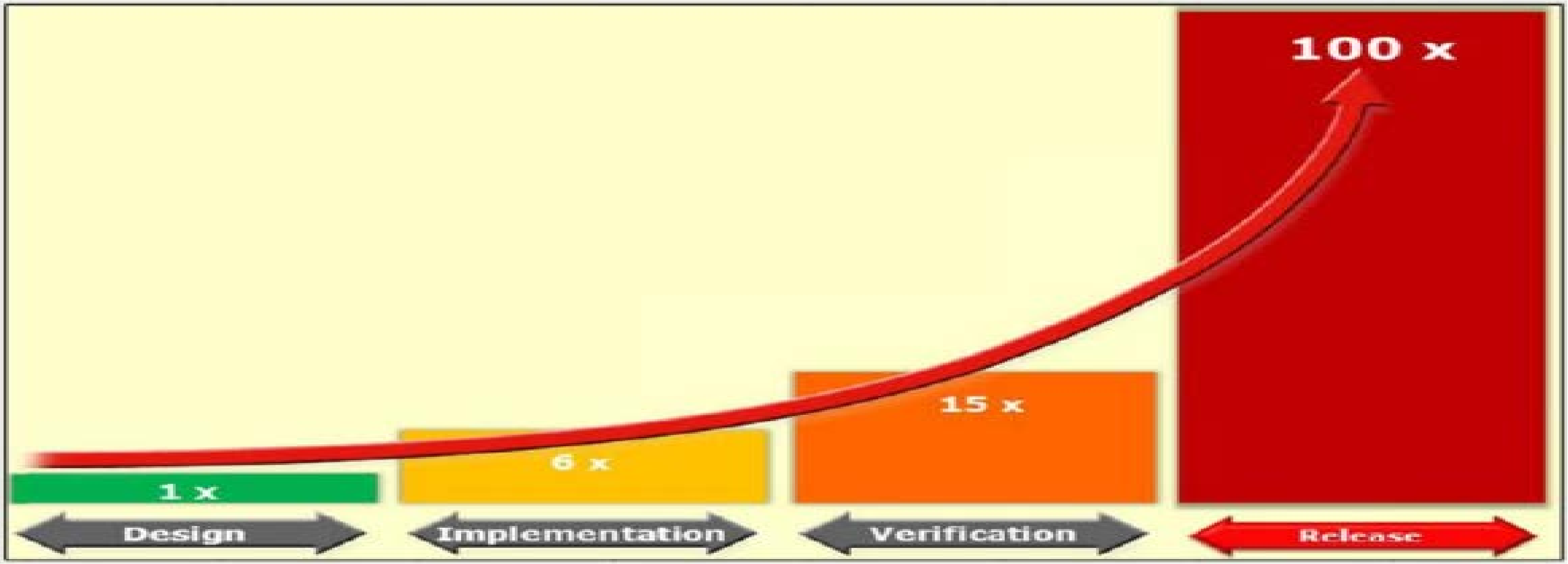


Figure 1: Cost of Bug Elimination in the Software Development Lifecycle [NIST 2002]

1. LET'S AVOID

D-LINK ROUTER BACKDOOR

The DI-524UP is a Wireless Broadband Router ideal for home networking and small business networking. The Setup Wizard will guide you to configure the DI-524UP to connect to your ISP (Internet Service Provider). The DI-524UP's easy setup will allow you to have Internet access within minutes. Please follow the setup wizard step by step to configure the DI-524UP.

CVE-2013-6462: Stack buffer overflow (20 years old)

Multiple CVEs:

- CVE-2013-5359
- CVE-2013-5358
- CVE-2013-5357
- CVE-2013-5349

1. BINARY INTELLIGENCE

File Information

- Size
- Hash
- Timestamp
- Strings

Compiler

- Name
- Version

Security Mitigations

- DEP
- ASLR
- Stack Cookies

Vulnerabilities

- Unsafe API
- Weak Crypto
- Backdoors

2. Security Mitigations

2. SOME COMPILERS OFFER GOOD SECURITY DEFENSES

- Visual Studio
- GCC
- LLVM (Xcode)

2. SDL MICROSOFT GUIDE 5.2

- “Use minimum code generation suite and libraries. For unmanaged, native C/C++ code, use Visual C++ 2010 as it offers all the SDL-mandated compiler and linker flags, including /GS, /DYNAMICBASE, /NXCOMPAT, and /SAFESEH. For managed code, use Visual Studio® 2008 SP1 or later. Use the currently required (or later) versions of compilers to compile options for the Win32®, Win64, WinCE, and Macintosh target platforms, as listed in Appendix E: SDL Required and Recommended Compilers, Tools, and Options for All Platforms. The biggest change in Visual Studio 2008 SP1 and later is Data Execution Prevention (DEP) support, enabled by default for all binaries, which can help protect against classes of buffer overrun.”
- “For unmanaged C or C++ code, BinScope must indicate a "Pass" in the compiler version field for all binaries. For managed code, an attestation is required that the compiler version used to ship the product is the version outlined in this document or later.”
- “Banned application programming interfaces (APIs). All native C and C++ code must not use banned versions of string buffer handling functions.”

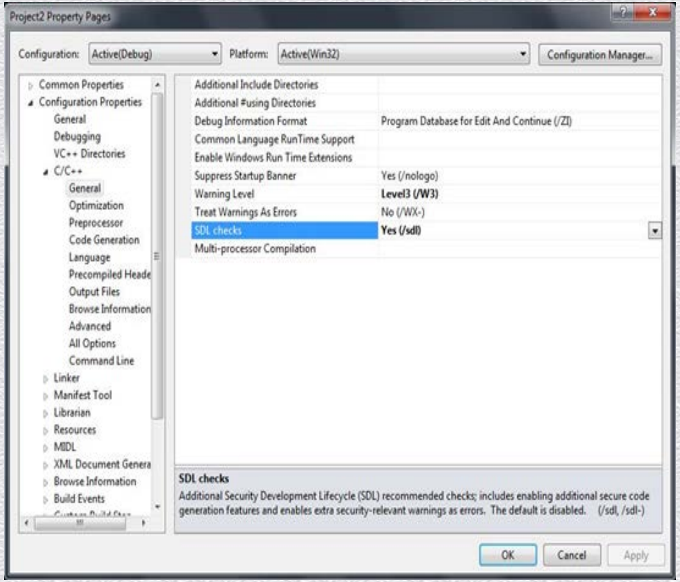
http://www.microsoft.com/en-us/download/confirmation.aspx?id=29884

2. Visual Studio Defenses

VS 2008	VS 2010	VS 2012
/Analyze (1)	/Analyze	/Analyze
/GS	/GS (2)	/GS (2)
strict_gs_check	strict_gs_check	strict_gs_check
/Hotpatch	/Hotpatch	/Hotpatch
/SafeSEH	/SafeSEH	/SafeSEH
/DYNAMICBASE	/DYNAMICBASE	/DYNAMICBASE
/NXCOMPAT	/NXCOMPAT	/NXCOMPAT
		/SDL

1) Only available in Visual Studio Ultimate
2) Defense enhanced

2. Visual Studio Defenses



Project2 Property Pages

Configuration: Active(Debug) Platform: Active(Win32) Configuration Manager...

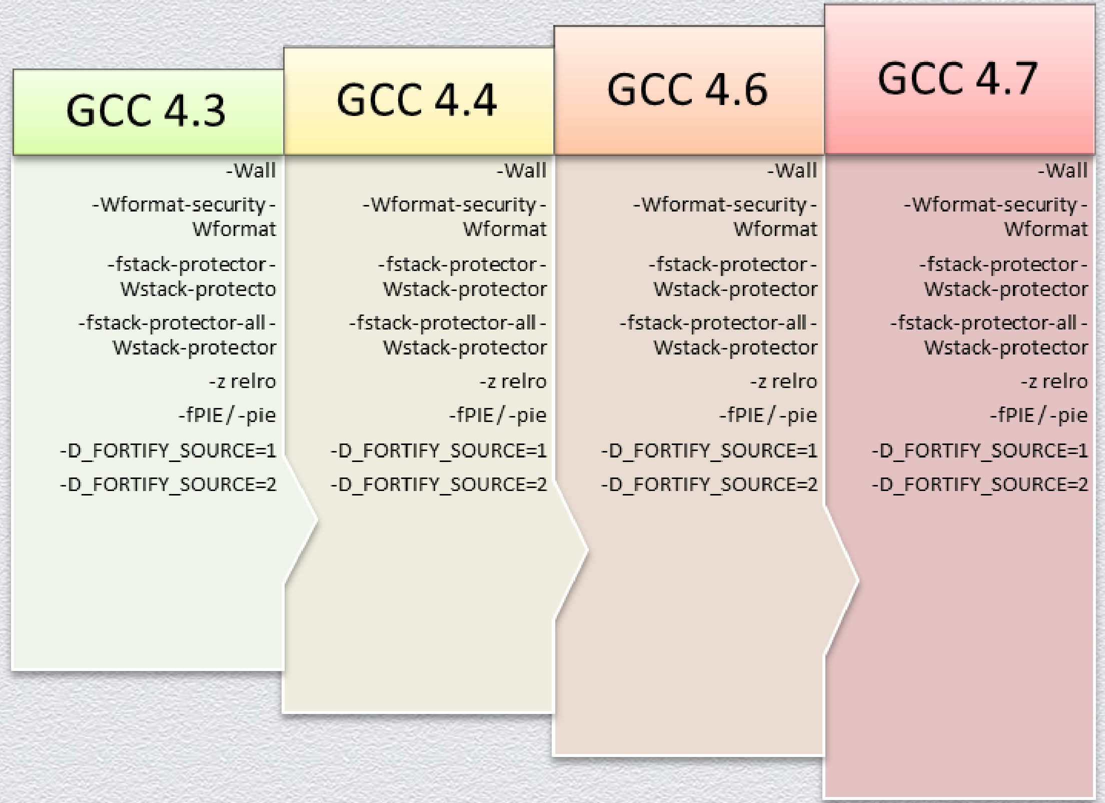
Additional Include Directories	
Additional #using Directories	
Debug Information Format	Program Database for Edit And Continue (/ZI)
Common Language RunTime Support	
Enable Windows Run Time Extensions	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

SDL checks

Additional Security Development Lifecycle (SDL) recommended checks; includes enabling additional secure code generation features and enables extra security-relevant warnings as errors. The default is disabled. (/sdl, /sdl-)

- Stack buffer protection (/GS)
- Code Analysis
- Data Execution Prevention (DEP)
- Address Space Layout Randomization (ASLR)
- Security Development Lifecycle (/SDL)(VS 2012)
  - /sdl causes SDL mandatory compiler warnings to be treated as errors during compilation.
  - /sdl enables additional code generation features such as increasing the scope of stack buffer overrun protection and initialization or sanitization of pointers in a limited set of well-defined scenarios.

2. GCC Defenses



GCC 4.3	GCC 4.4	GCC 4.6	GCC 4.7
-Wall	-Wall	-Wall	-Wall
-Wformat-security -Wformat	-Wformat-security -Wformat	-Wformat-security -Wformat	-Wformat-security -Wformat
-fstack-protector -Wstack-protecto	-fstack-protector -Wstack-protector	-fstack-protector -Wstack-protector	-fstack-protector -Wstack-protector
-fstack-protector-all -Wstack-protector	-fstack-protector-all -Wstack-protector	-fstack-protector-all -Wstack-protector	-fstack-protector-all -Wstack-protector
-z relro	-z relro	-z relro	-z relro
-fPIE / -pie	-fPIE / -pie	-fPIE / -pie	-fPIE / -pie
-D_FORTIFY_SOURCE=1	-D_FORTIFY_SOURCE=1	-D_FORTIFY_SOURCE=1	-D_FORTIFY_SOURCE=1
-D_FORTIFY_SOURCE=2	-D_FORTIFY_SOURCE=2	-D_FORTIFY_SOURCE=2	-D_FORTIFY_SOURCE=2

2. GCC SECURITY

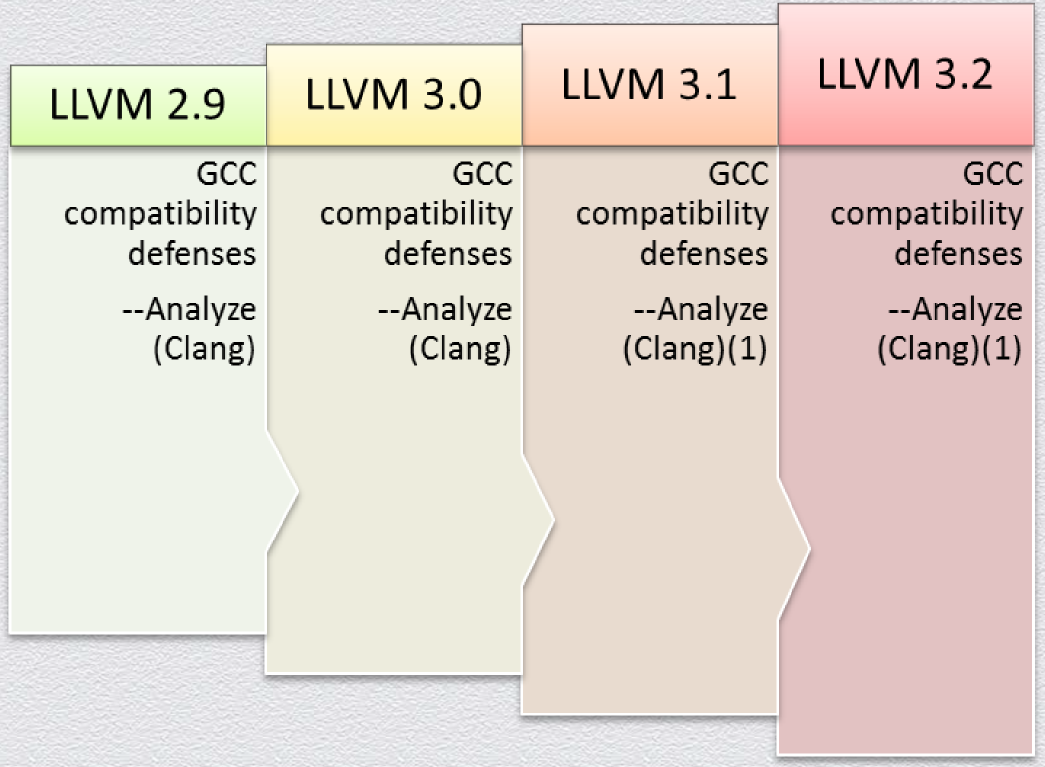
- Decent security defenses by not enabled by default ☹

- Mudflap Pointer Debugging (removed in GCC 4.9, in favor of Address Sanitizer)
  - Instruments for buffer overflows

- Address Sanitizer (http://code.google.com/p/address-sanitizer/) GCC 4.8
  - It finds use-after-free and {heap,stack,global}-buffer overflow bugs in C/C++ programs.

- -fstack-protector strong included in GCC 4.9, previous version as a patch

2. LLVM Defenses



LLVM 2.9	LLVM 3.0	LLVM 3.1	LLVM 3.2
GCC compatibility defenses	GCC compatibility defenses	GCC compatibility defenses	GCC compatibility defenses
--Analyze (Clang)	--Analyze (Clang)	--Analyze (Clang)(1)	--Analyze (Clang)(1)

1) Enhanced defense

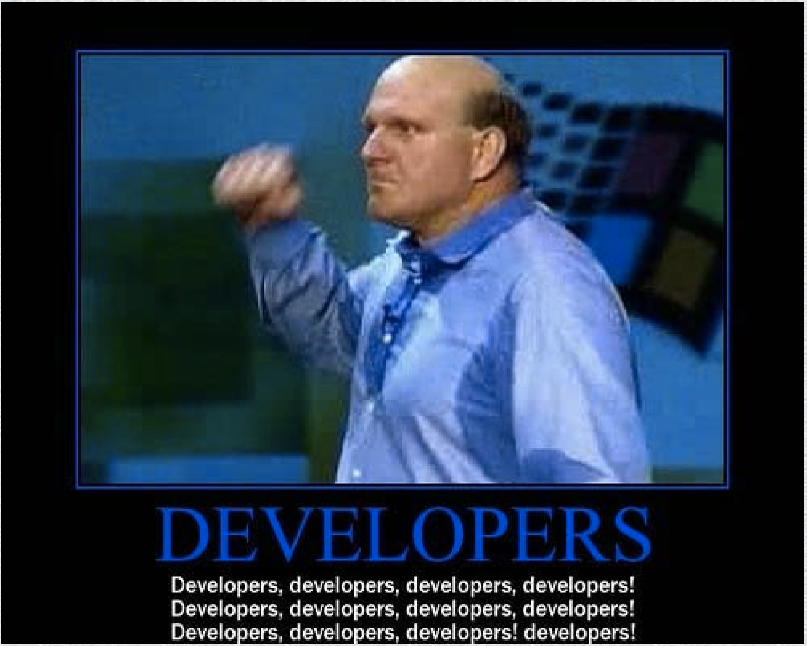
2. LLVM SECURITY

- Some mitigations enabled by default

- Clang Static Analyzer
  - http://clang-analyzer.llvm.org/available_checks.html

2. DEVELOPERS! DEVELOPERS!

- No excuse, build & ship software with defenses enabled


DEVELOPERS
Developers, developers, developers, developers!
Developers, developers, developers, developers!
Developers, developers, developers! developers!

3. BinSecSweeper

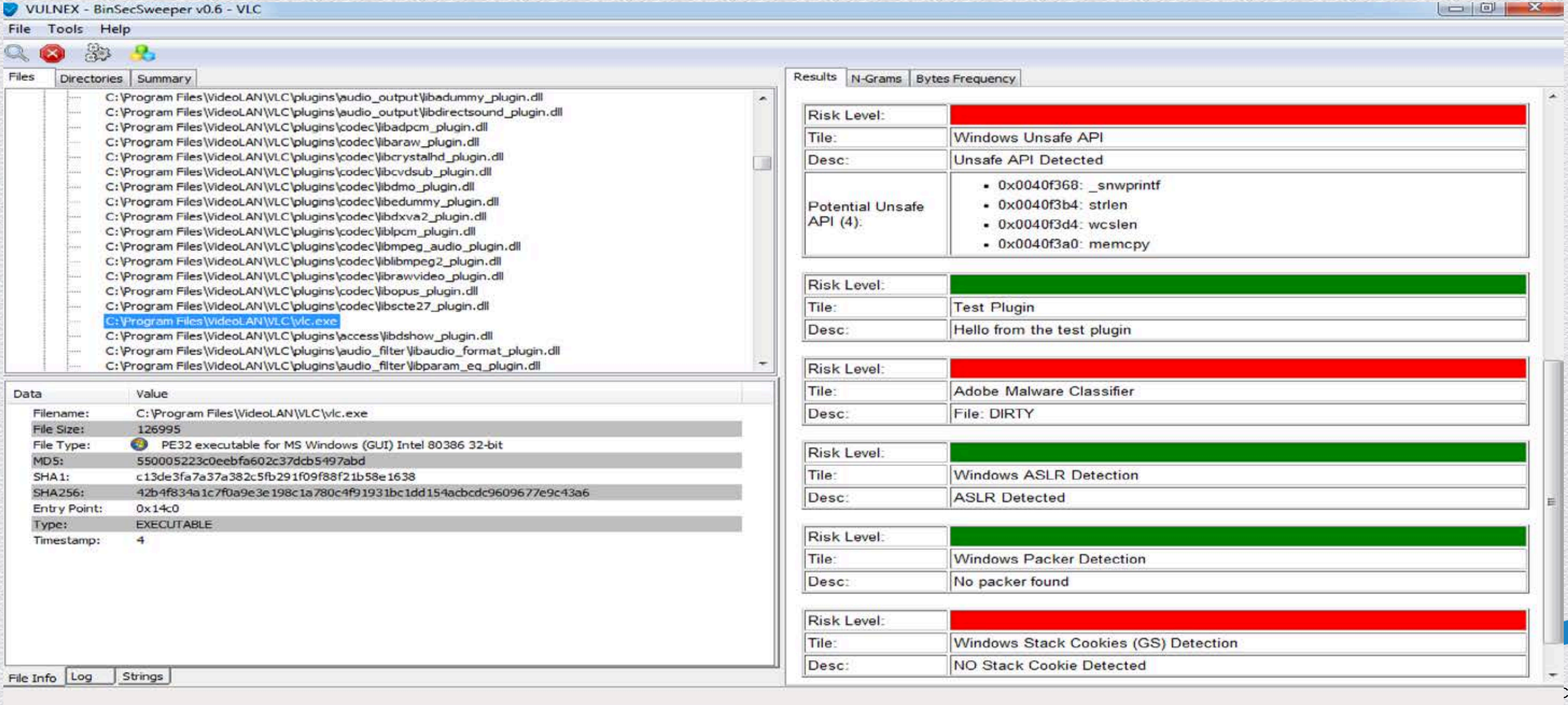
3. Why BinSecSweeper?

- BinSecSweeper is VULNEX binary security verification tool to ensure applications have been built in compliance with Application Assurance best practices
- The goal for BinSecSweeper is a tool:
  - Developers can use to verify that their output binaries are safe after compilation and before releasing their products
  - IT security pros to scan their infrastructure to identify binaries with weak security defenses or vulnerabilities.
- BinSecSweeper is a cross platform tool (works on Windows and Linux) and can scan different file formats: PE and ELF.

3. FEATURES

- 100% open source
- Easy to use
- Cross-platform works on Windows & Linux
- Scans Windows (PE) and Unix (ELF) files for security checks
- Configurable
- Analysis Engine
- Extensible by plugins
- Reporting

3. BinSecSweeper in Action I



```
C:\Program Files\VideoLAN\VLC\plugins\audio_output\libadummy_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\audio_output\libdirectsound_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libadpcm_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libaraw_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libcrystalhd_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libcvdsub_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libdmo_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libedummy_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libdxva2_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\liblpcm_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libmpeg_audio_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\liblibmpeg2_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\librawvideo_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libopus_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\codec\libscte27_plugin.dll
C:\Program Files\VideoLAN\VLC\vlc.exe
C:\Program Files\VideoLAN\VLC\plugins\access\libdshow_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\audio_filter\libaudio_format_plugin.dll
C:\Program Files\VideoLAN\VLC\plugins\audio_filter\libparam_eq_plugin.dll
```

Data	Value
Filename:	C:\Program Files\VideoLAN\VLC\vlc.exe
File Size:	126995
File Type:	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
MD5:	550005223c0eebfa602c37dcb5497abd
SHA1:	c13de3fa7a37a382c5fb291f09f88f21b58e1638
SHA256:	42b4f834a1c7f0a9e3e198c1a780c4f91931bc1dd154acbcdc9609677e9c43a6
Entry Point:	0x14c0
Type:	EXECUTABLE
Timestamp:	4

Risk Level:	🟥
Tile:	Windows Unsafe API
Desc:	Unsafe API Detected
Potential Unsafe API (4):	• 0x0040f368: _snwprintf • 0x0040f3b4: strlen • 0x0040f3d4: wcslen • 0x0040f3a0: memcpy

Risk Level:	🟩
Tile:	Test Plugin
Desc:	Hello from the test plugin

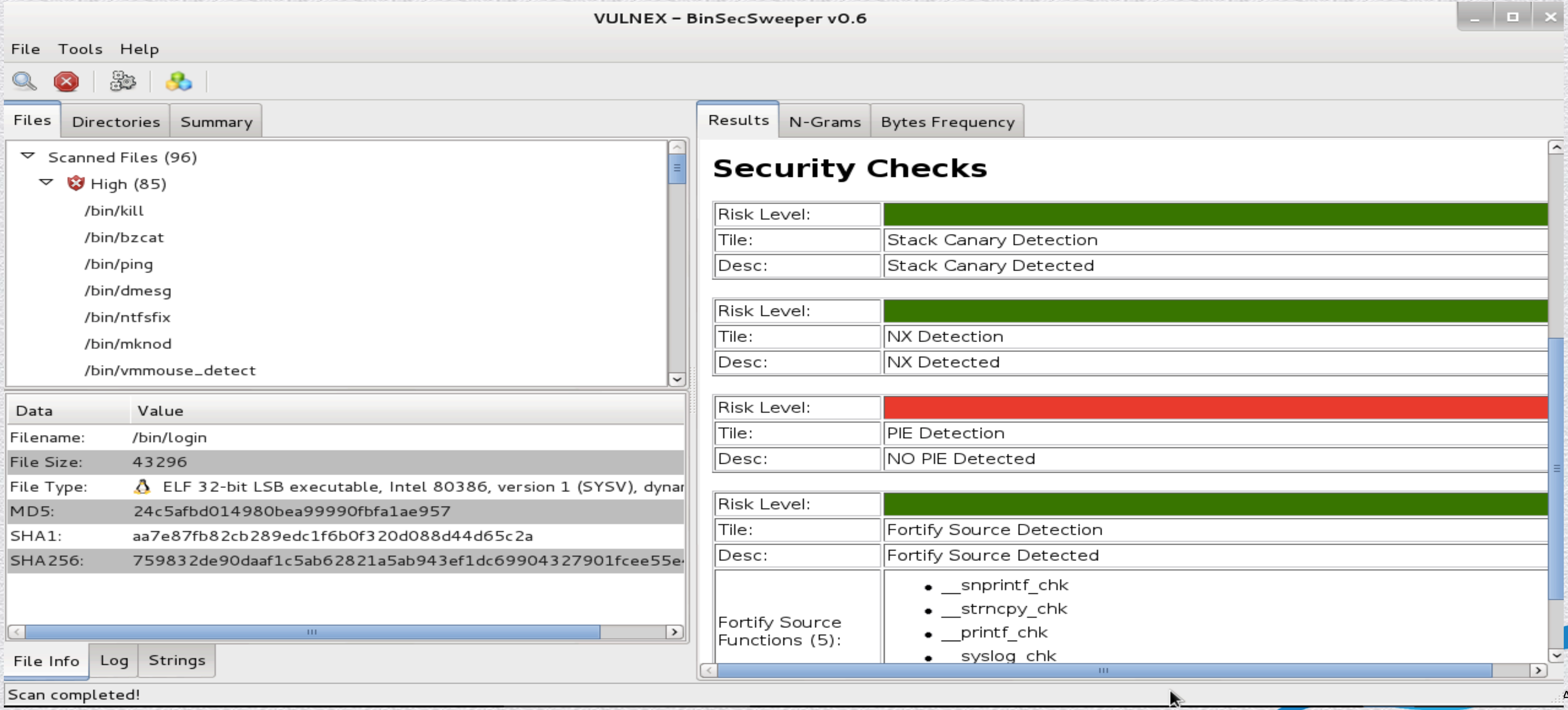
Risk Level:	🟥
Tile:	Adobe Malware Classifier
Desc:	File: DIRTY

Risk Level:	🟩
Tile:	Windows ASLR Detection
Desc:	ASLR Detected

Risk Level:	🟩
Tile:	Windows Packer Detection
Desc:	No packer found

Risk Level:	🟥
Tile:	Windows Stack Cookies (GS) Detection
Desc:	NO Stack Cookie Detected

3. BinSecSweeper in Action II



```
▽ Scanned Files (96)
  ▽ High (85)
      /bin/kill
      /bin/bzcat
      /bin/ping
      /bin/dmesg
      /bin/ntfsfix
      /bin/mknod
      /bin/vmmouse_detect
```

Data	Value
Filename:	/bin/login
File Size:	43296
File Type:	ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynar
MD5:	24c5afbd014980bea99990fbfa1ae957
SHA1:	aa7e87fb82cb289edc1f6b0f320d088d44d65c2a
SHA256:	759832de90daaf1c5ab62821a5ab943ef1dc69904327901fcee55e

Security Checks

Risk Level:	🟩
Tile:	Stack Canary Detection
Desc:	Stack Canary Detected

Risk Level:	🟩
Tile:	NX Detection
Desc:	NX Detected

Risk Level:	🟥
Tile:	PIE Detection
Desc:	NO PIE Detected

Risk Level:	🟩
Tile:	Fortify Source Detection
Desc:	Fortify Source Detected
Fortify Source Functions (5):	• __snprintf_chk • __strncpy_chk • __printf_chk • syslog_chk

Scan completed!

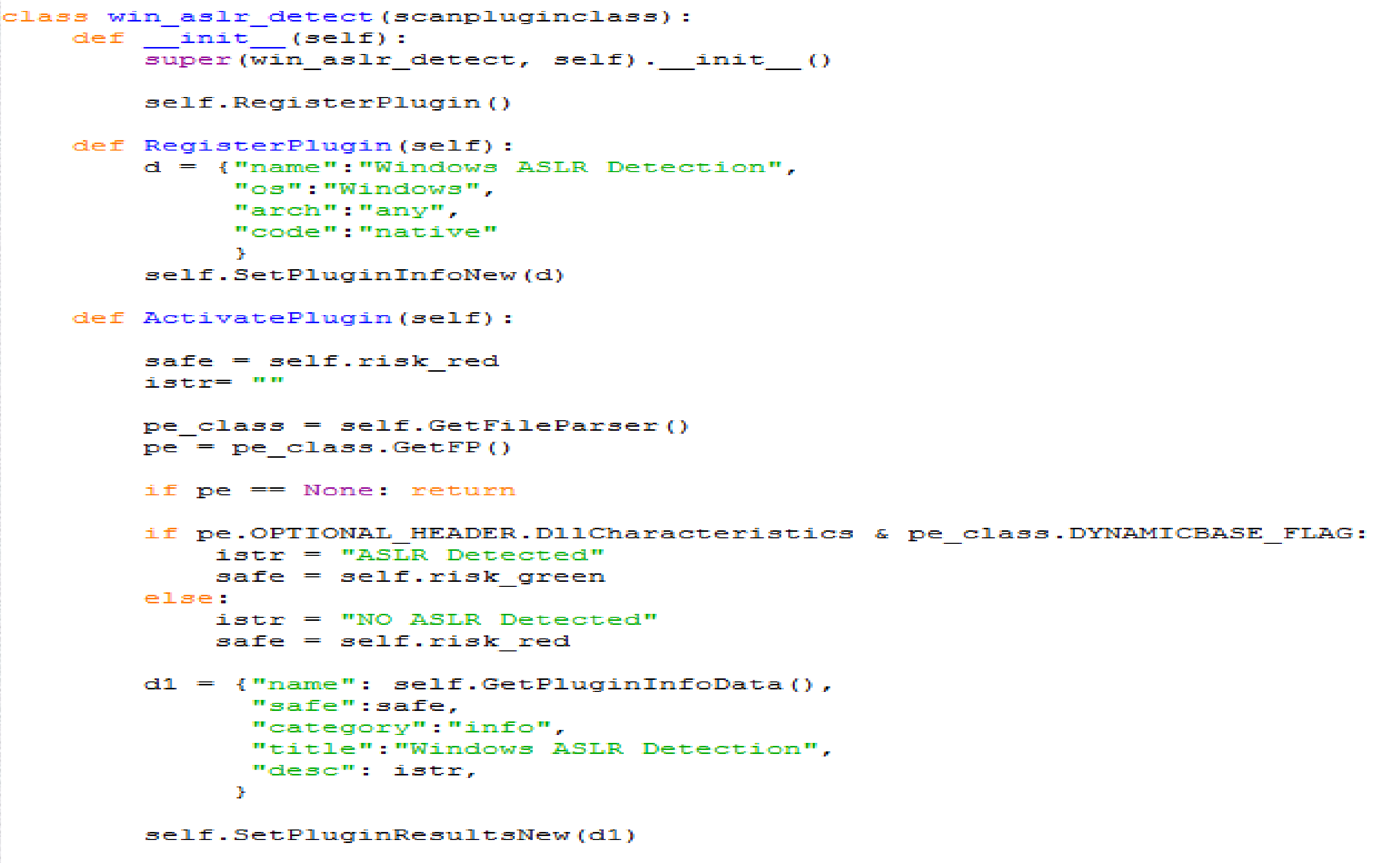
3. Current Windows Checks

CHECK	DESCRIPTION
Address space layout randomization (ASLR)	Checks if binary has opted the ASLR. Link with /DYNAMICBASE
Stack Cookies (GS)	Verifies if binary was compiled with Stack Cookies protection. Compile with /GS
HotPatch	Checks if binary is prepared for hot patching. Compile with /hotpatch
Compatible with Data Execution Prevention (NXCOMPAT)	Validates if binary has opted hardware Data Execution Prevention (DEP). Link with /NXCOMPAT
Structured Exception Handling (SEH)	Checks if binary was linked with SafeSEH. Link with /SAFESEH
Abobe Malware Classifier	Analyzes binary for malware behavior using machine learning algorithms
Visual Studio Compiler Fingerprinting	Identifies if binary was compiled with Visual Studio and version (2005, 2008, 2010 & 2012)
Packer	Checks if binary has been packed
Insecure API	Check if binary uses banned API

3. Current Linux Checks

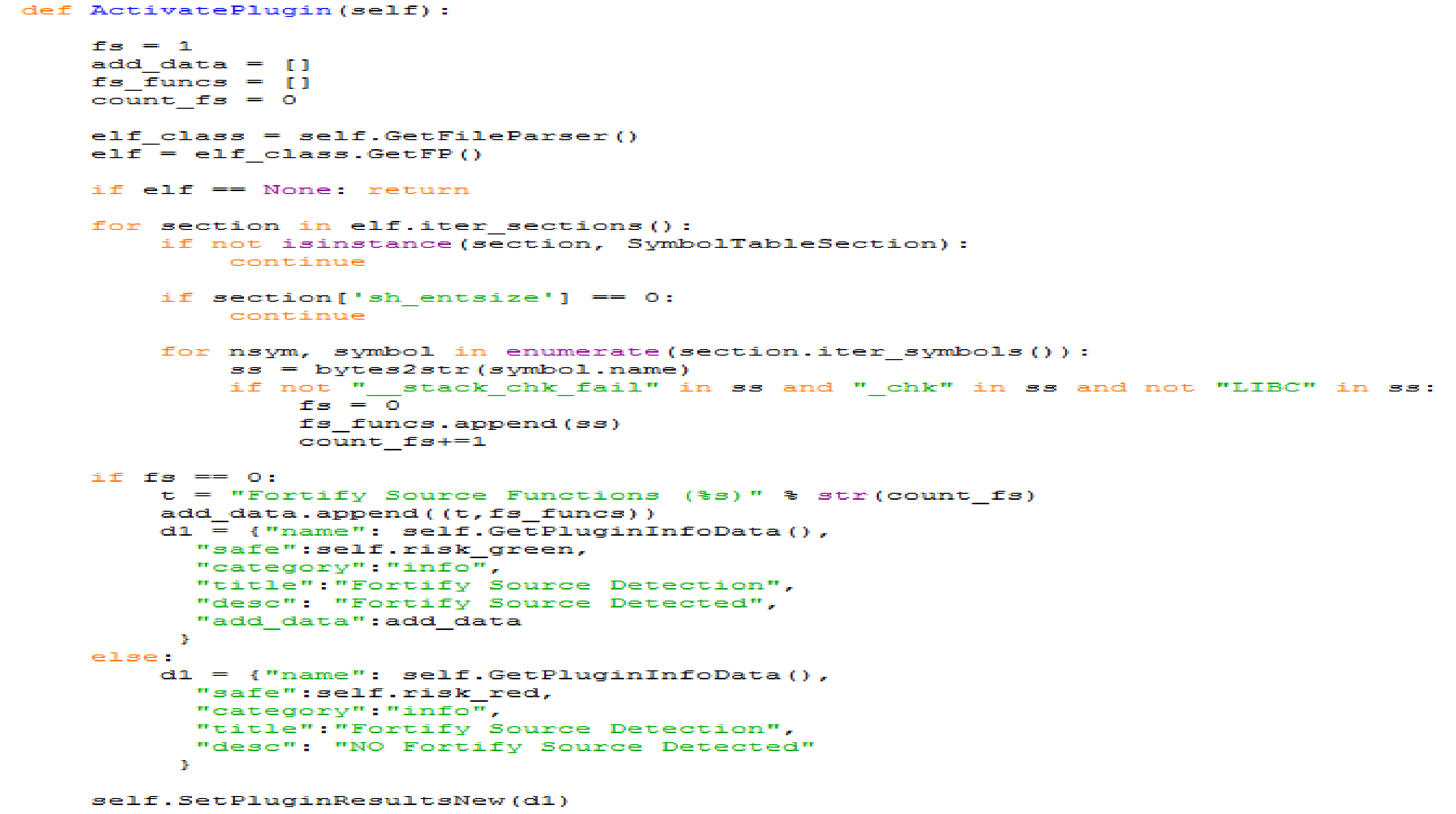
CHECK	DESCRIPTION
Fortify Source	Checks if binary was compiled with buffer overflow protection (bounds checking). Compile with –D_FORTIFY_SOURCE=X
Never eXecute (NX)	Verifies if binary was compiled with NX to reduce the area an attacker can use to perform arbitrary code execution.
Position Independent Code (PIE)	Checks if binary was compiled with PIE to protects against "return-to-text" and generally frustrates memory corruption attacks. Compile with –fPIE -pie
RELocation Read-Only (RELRO)	Validates if binary was compiled with RELRO (partial/full) to harden data sections. Compile with –z,relro,-z,now
Stack Canary	Checks if binary was compiled with stack protector to protect against stack overflows. Compile with –fstack-protector

3. Plugin Example: Windows ASLR



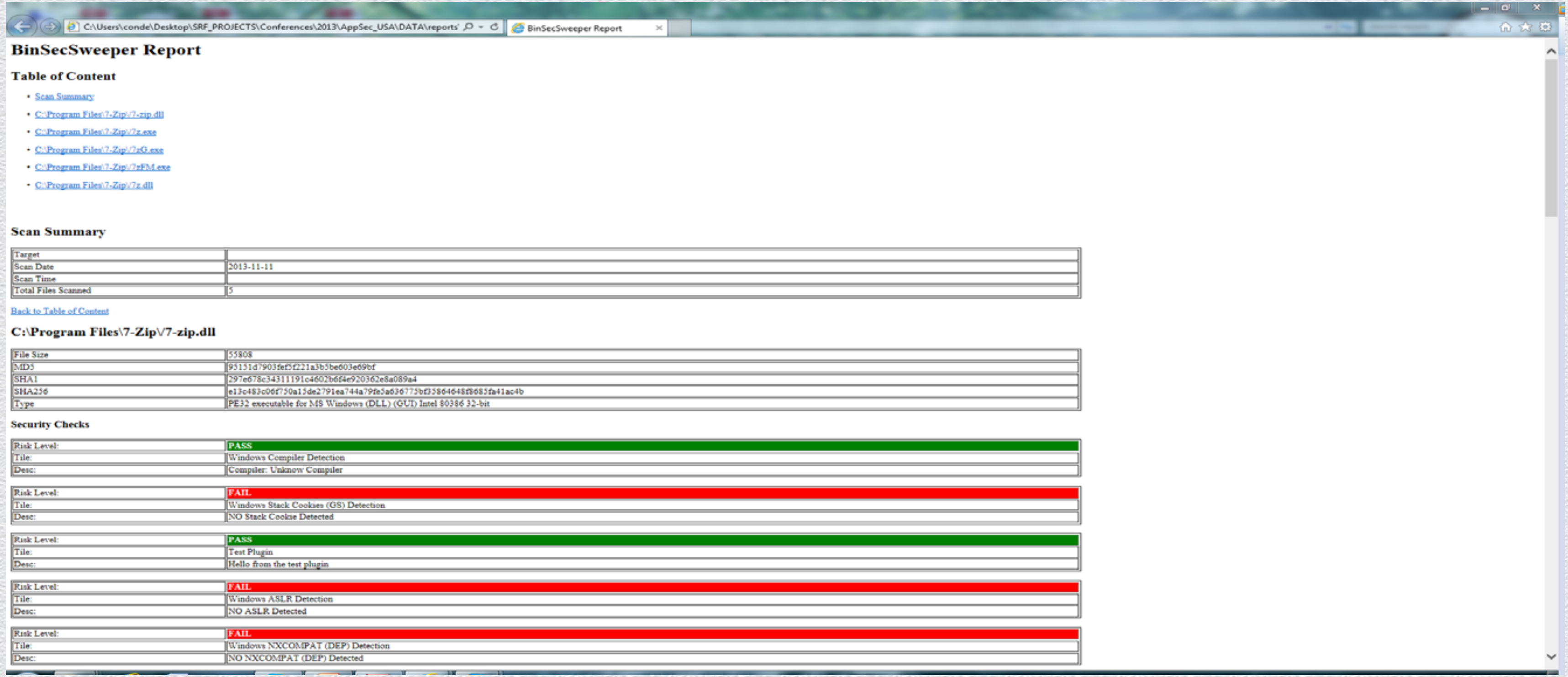
```
class win_aslr_detect(scanpluginclass):
    def __init__(self):
        super(win_aslr_detect, self).__init__()

        self.RegisterPlugin()

    def RegisterPlugin(self):
        d = {"name":"Windows ASLR Detection",
             "os":"Windows",
             "arch":"any",
             "code":"native"
             }
        self.SetPluginInfoNew(d)

    def ActivatePlugin(self):

        safe = self.risk_red
        istr= ""

        pe_class = self.GetFileParser()
        pe = pe_class.GetFP()

        if pe == None: return

        if pe.OPTIONAL_HEADER.DllCharacteristics & pe_class.DYNAMICBASE_FLAG:
            istr = "ASLR Detected"
            safe = self.risk_green
        else:
            istr = "NO ASLR Detected"
            safe = self.risk_red

        d1 = {"name": self.GetPluginInfoData(),
              "safe":safe,
              "category":"info",
              "title":"Windows ASLR Detection",
              "desc": istr,
             }

        self.SetPluginResultsNew(d1)
```

3. Plugin Example: Linux fortify_source



```
def ActivatePlugin(self):

    fs = 1
    add_data = []
    fs_funcs = []
    count_fs = 0

    elf_class = self.GetFileParser()
    elf = elf_class.GetFP()

    if elf == None: return

    for section in elf.iter_sections():
        if not isinstance(section, SymbolTableSection):
            continue

        if section['sh_entsize'] == 0:
            continue

        for nsym, symbol in enumerate(section.iter_symbols()):
            ss = bytes2str(symbol.name)
            if not "__stack_chk_fail" in ss and "_chk" in ss and not "LIBC" in ss:
                fs = 0
                fs_funcs.append(ss)
                count_fs+=1

    if fs == 0:
        t = "Fortify Source Functions (%s)" % str(count_fs)
        add_data.append((t,fs_funcs))
        d1 = {"name": self.GetPluginInfoData(),
          "safe":self.risk_green,
          "category":"info",
          "title":"Fortify Source Detection",
          "desc": "Fortify Source Detected",
          "add_data":add_data
         }
    else:
        d1 = {"name": self.GetPluginInfoData(),
          "safe":self.risk_red,
          "category":"info",
          "title":"Fortify Source Detection",
          "desc": "NO Fortify Source Detected"
         }

    self.SetPluginResultsNew(d1)
```

3. Reporting


C:\Users\conde\Desktop\SRF_PROJECTS\Conferences\2013\AppSec_USA\DATA\reports\
BinSecSweeper Report

BinSecSweeper Report

Table of Content

Scan Summary

Target	
Scan Date	2013-11-11
Scan Time	
Total Files Scanned	5

Back to Table of Content

C:\Program Files\7-Zip\/7-zip.dll

File Size	55808
MD5	95151d7903fef5f221a3b5be603e69bf
SHA1	297e678c34311191c4602b6f4e920362e8a089a4
SHA256	e13c483c06f750a15de2791ea744a79fe5a636775bf35864648f8685fa41ac4b
Type	PE32 executable for MS Windows (DLL) (GUI) Intel 80386 32-bit

Security Checks

Risk Level:	PASS
Tile:	Windows Compiler Detection
Desc:	Compiler: Unknown Compiler

Risk Level:	FAIL
Tile:	Windows Stack Cookies (GS) Detection
Desc:	NO Stack Cookie Detected

Risk Level:	PASS
Tile:	Test Plugin
Desc:	Hello from the test plugin

Risk Level:	FAIL
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected

Risk Level:	FAIL
Tile:	Windows NXCOMPAT (DEP) Detection
Desc:	NO NXCOMPAT (DEP) Detected

3. BinSecSweeper: what's next!

- More plugins:
  - Windows, Linux, etc.
  - Mobile
  - Malware
  - Backdoors
  - Compilers
  - Packers
- Metrics panel
- Diff across product / versions

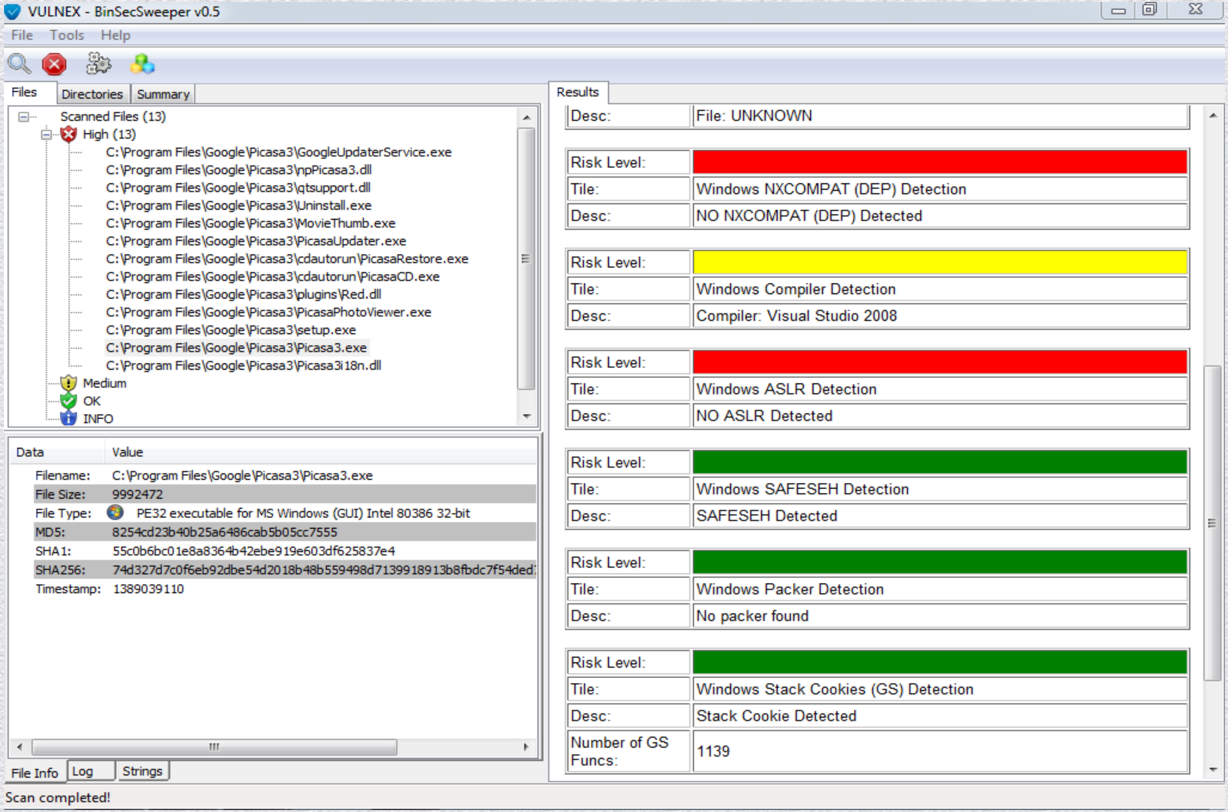
3. BinSecSwepeper: where?

- Download BinSecSweeper software from www.vulnex.com

- After RSA USA (please give us a couple of weeks to finish up doc ☺)

4. Case Studies

4. Remember Picassa?



```
Scanned Files (13)
  High (13)
    C:\Program Files\Google\Picasa3\GoogleUpdaterService.exe
    C:\Program Files\Google\Picasa3\npPicasa3.dll
    C:\Program Files\Google\Picasa3\qtsupport.dll
    C:\Program Files\Google\Picasa3\Uninstall.exe
    C:\Program Files\Google\Picasa3\MovieThumb.exe
    C:\Program Files\Google\Picasa3\PicasaUpdater.exe
    C:\Program Files\Google\Picasa3\cdautorun\PicasaRestore.exe
    C:\Program Files\Google\Picasa3\cdautorun\PicasaCD.exe
    C:\Program Files\Google\Picasa3\plugins\Red.dll
    C:\Program Files\Google\Picasa3\PicasaPhotoViewer.exe
    C:\Program Files\Google\Picasa3\setup.exe
    C:\Program Files\Google\Picasa3\Picasa3.exe
    C:\Program Files\Google\Picasa3\Picasa3i18n.dll
  Medium
  OK
  INFO
```

Data	Value
Filename:	C:\Program Files\Google\Picasa3\Picasa3.exe
File Size:	9992472
File Type:	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
MD5:	8254cd23b40b25a6486cab5b05cc7555
SHA1:	55c0b6bc01e8a8364b42ebe919e603df625837e4
SHA256:	74d327d7c0f6eb92dbe54d2018b48b559498d7139918913b8fbdc7f54ded
Timestamp:	1389039110

Desc:	File: UNKNOWN
Risk Level:	🟥
Tile:	Windows NXCOMPAT (DEP) Detection
Desc:	NO NXCOMPAT (DEP) Detected
Risk Level:	🟨
Tile:	Windows Compiler Detection
Desc:	Compiler: Visual Studio 2008
Risk Level:	🟥
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected
Risk Level:	🟩
Tile:	Windows SAFESEH Detection
Desc:	SAFESEH Detected
Risk Level:	🟩
Tile:	Windows Packer Detection
Desc:	No packer found
Risk Level:	🟩
Tile:	Windows Stack Cookies (GS) Detection
Desc:	Stack Cookie Detected
Number of GS Funcs:	1139

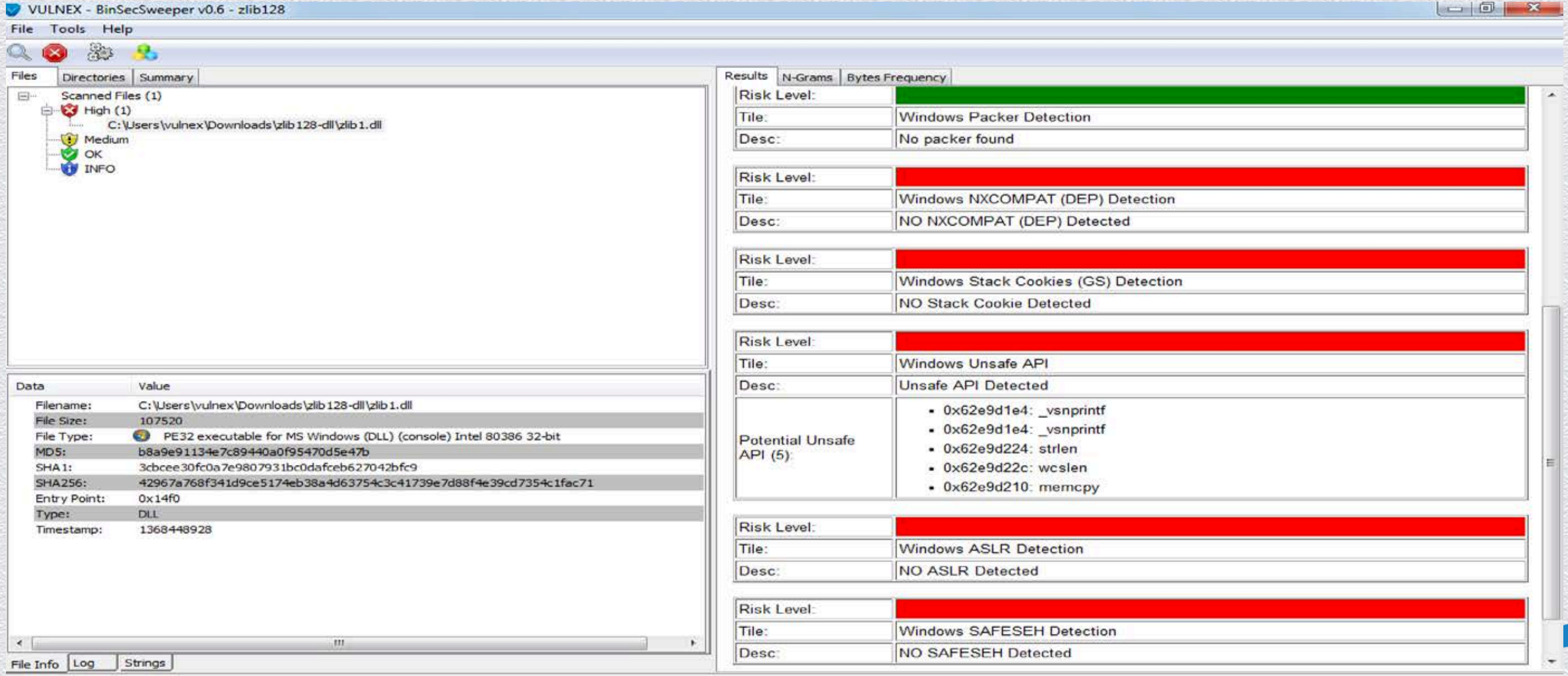
Scan completed!

Missing: ASLR + DEP

Good: Stack Cookies

But was still exploitable!

4. Are you compiling your app with zlib.dll ?



Data	Value
Filename:	C:\Users\vulnex\Downloads\zlib128-dll\zlib1.dll
File Size:	107520
File Type:	PE32 executable for MS Windows (DLL) (console) Intel 80386 32-bit
MD5:	b8a9e91134e7c89440a0f95470d5e47b
SHA1:	3cbcee30fc0a7e9807931bc0dafceb627042bfc9
SHA256:	42967a768f341d9ce5174eb38a4d63754c3c41739e7d88f4e39cd7354c1fac71
Entry Point:	0x14f0
Type:	DLL
Timestamp:	1368448928

Risk Level:	🟩
Tile:	Windows Packer Detection
Desc:	No packer found

Risk Level:	🟥
Tile:	Windows NXCOMPAT (DEP) Detection
Desc:	NO NXCOMPAT (DEP) Detected

Risk Level:	🟥
Tile:	Windows Stack Cookies (GS) Detection
Desc:	NO Stack Cookie Detected

Risk Level:	🟥
Tile:	Windows Unsafe API
Desc:	Unsafe API Detected
Potential Unsafe API (5):	• 0x62e9d1e4: _vsnprintf • 0x62e9d1e4: _vsnprintf • 0x62e9d224: strlen • 0x62e9d22c: wcslen • 0x62e9d210: memcpy

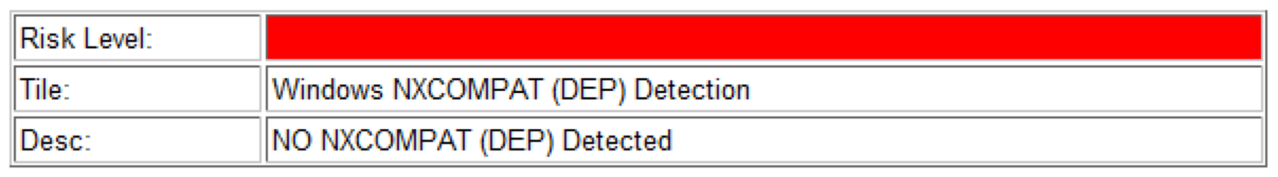
Risk Level:	🟥
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected

Risk Level:	🟥
Tile:	Windows SAFESEH Detection
Desc:	NO SAFESEH Detected

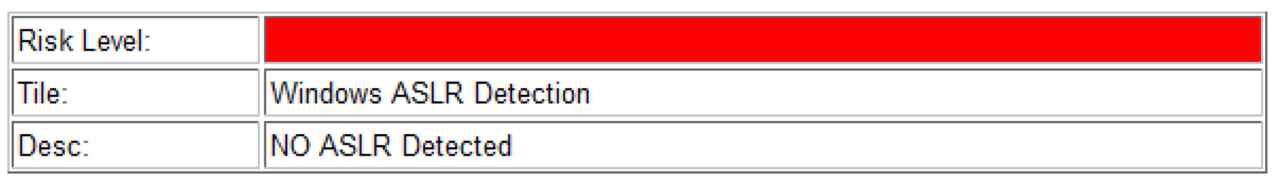
VULNEX
OFFENSIVE & DEFENSIVE CYBER SECURITY

4. Are your 3rd party components improving?

- Python 2.7 -> sqlite3.dll

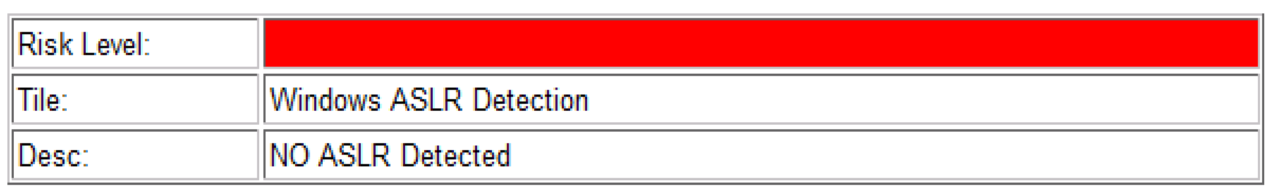


Risk Level:	🟥
Tile:	Windows NXCOMPAT (DEP) Detection
Desc:	NO NXCOMPAT (DEP) Detected



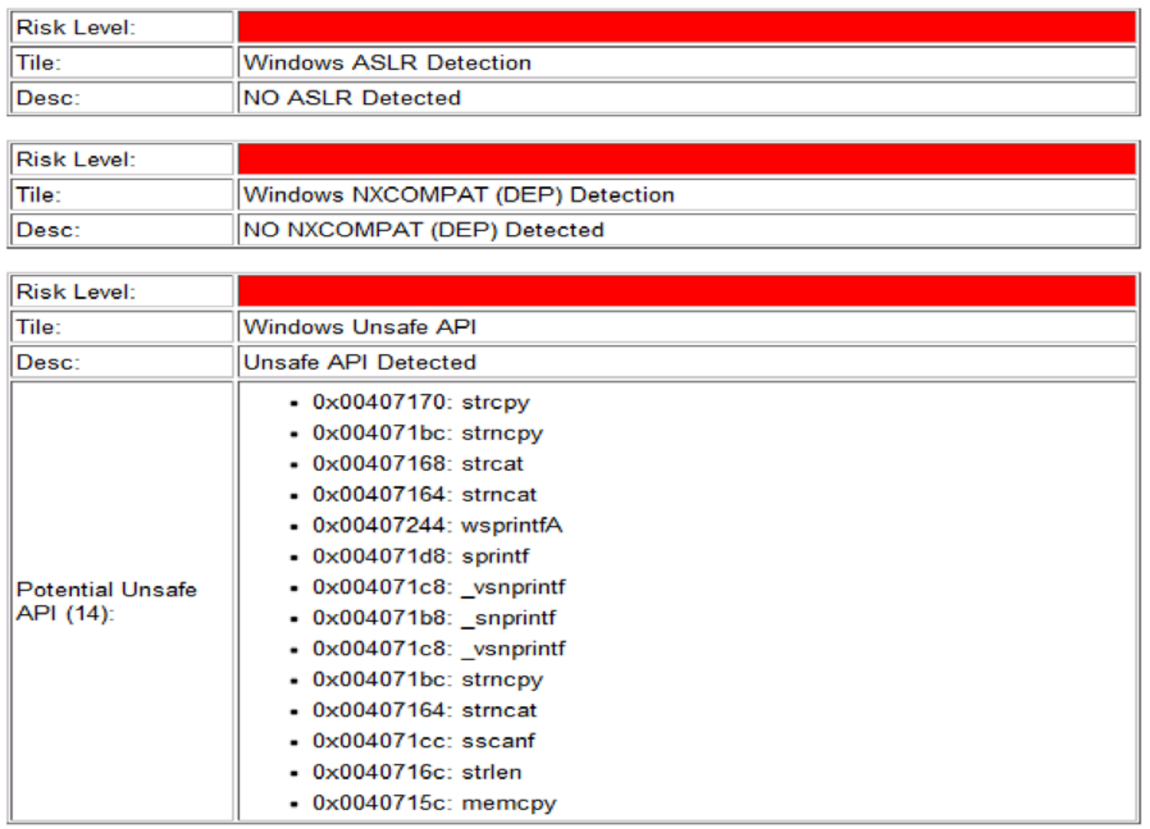
Risk Level:	🟥
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected

- Python 3.3 -> sqlite3.dll



Risk Level:	🟥
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected

4. A DLL inside a well-known software

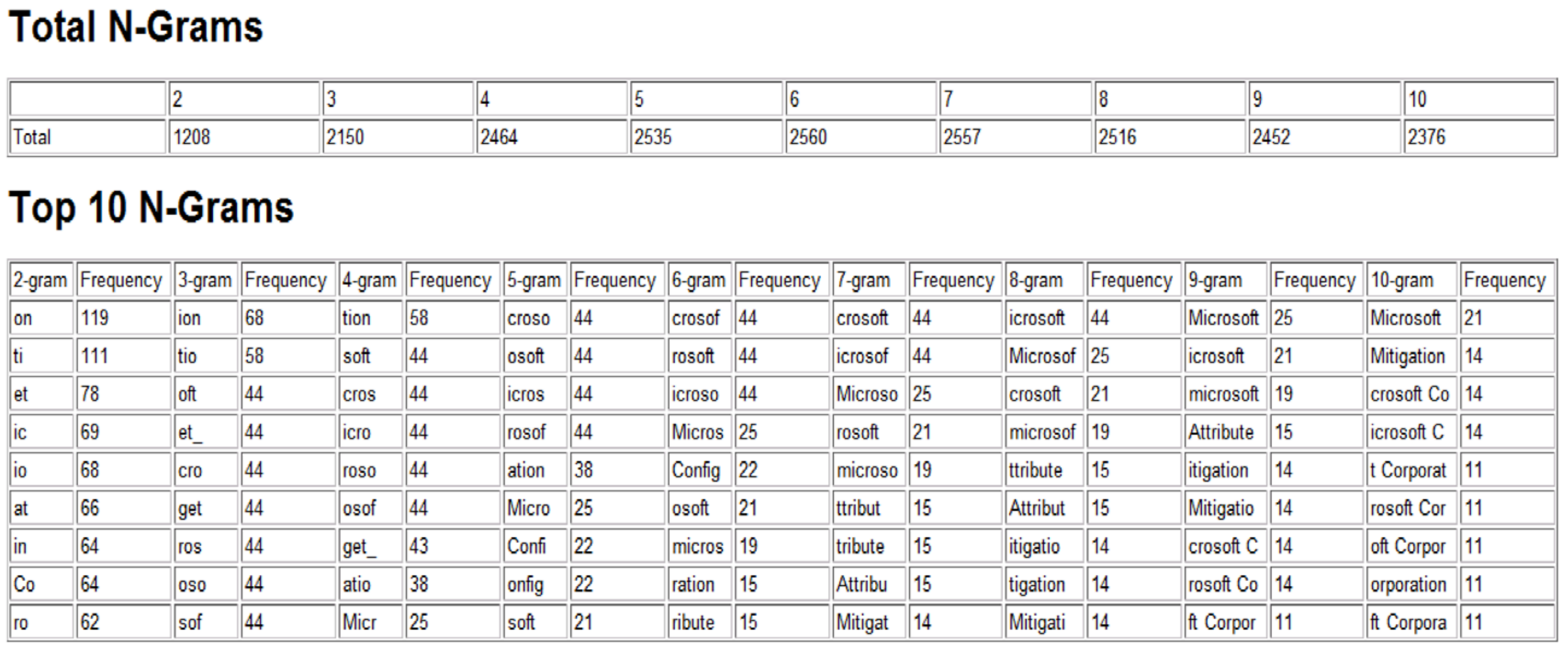


Risk Level:	🟥
Tile:	Windows ASLR Detection
Desc:	NO ASLR Detected

Risk Level:	🟥
Tile:	Windows NXCOMPAT (DEP) Detection
Desc:	NO NXCOMPAT (DEP) Detected

Risk Level:	🟥
Tile:	Windows Unsafe API
Desc:	Unsafe API Detected
Potential Unsafe API (14):	• 0x00407170: strcpy • 0x004071bc: strncpy • 0x00407168: strcat • 0x00407164: strncat • 0x00407244: wsprintfA • 0x004071d8: sprintf • 0x004071c8: _vsnprintf • 0x004071b8: _snprintf • 0x004071c8: _vsnprintf • 0x004071bc: strncpy • 0x00407164: strncat • 0x004071cc: sscanf • 0x0040716c: strlen • 0x0040715c: memcpy

4. The most common word inside a Microsoft binary?



Total N-Grams

	2	3	4	5	6	7	8	9	10
Total	1208	2150	2464	2535	2560	2557	2516	2452	2376

Top 10 N-Grams

2-gram	Frequency	3-gram	Frequency	4-gram	Frequency	5-gram	Frequency	6-gram	Frequency	7-gram	Frequency	8-gram	Frequency	9-gram	Frequency	10-gram	Frequency
on	119	ion	68	tion	58	croso	44	crosof	44	crosoft	44	icrosoft	44	Microsoft	25	Microsoft	21
ti	111	tio	58	soft	44	osoft	44	rosoft	44	icrosof	44	Microsof	25	icrosoft	21	Mitigation	14
et	78	oft	44	cros	44	icros	44	icroso	44	Microso	25	crosoft	21	microsoft	19	crosoft Co	14
ic	69	et_	44	icro	44	rosof	44	Micros	25	rosoft	21	microsof	19	Attribute	15	icrosoft C	14
io	68	cro	44	roso	44	ation	38	Config	22	microso	19	ttribute	15	itigation	14	t Corporat	11
at	66	get	44	osof	44	Micro	25	osoft	21	ttribut	15	Attribut	15	Mitigatio	14	rosoft Cor	11
in	64	ros	44	get_	43	Confi	22	micros	19	tribute	15	itigatio	14	crosoft C	14	oft Corpor	11
Co	64	oso	44	atio	38	onfig	22	ration	15	Attribu	15	tigation	14	rosoft Co	14	orporation	11
ro	62	sof	44	Micr	25	soft	21	ribute	15	Mitigat	14	Mitigati	14	ft Corpor	11	ft Corpora	11

5. Conclusions

5. Verifying Software Security Posture Matters!

- Binaries contain a lot of information!

- The security posture of the software developed by you is important:
  - Security improves Quality
  - Branding (shows you care about security)

- How is the security posture of software vendors you use?

5. Does your Software:

- Has it been compiled with all possible mitigations?
- Use insecure APIs?
- Contain malware?
- Backdoors?

Q&A

- FIN
- Thanks!
- @simonroses