BinSecSweeper: Binary Security Posture Verification

OWASP Spain
Barcelona 2014
ME & VULNEX

Simon Roses Fenerling

- Founder & CEO, VULNEX [www.vulnex.com](http://www.vulnex.com)
- @simonroses | @vulnexsl
- Former Microsoft, PwC, @Stake
- Black Hat, RSA, OWASP, SOURCE, AppSec, DeepSec, TECHNET, HITB
- CISSP, CSSLP & CEH

VULNEX

- CyberSecurity Startup
- Services & Training
- Product development: BinSecSweeper and Computer Forensics Solution

TALK OBJECTIVES

• Secure development
• Verification technologies
• Assess software security posture
AGENDA

1. Secure Development: Verification
2. BinSecSweeper
3. Case Studies
4. Conclusions
1. Secure Development: Verification
1. SOFTWARE == HOUSE
1. SECURE DEVELOPMENT: VERIFICATION

• MS SDL
  – “This phase involves a comprehensive effort to ensure that the code meets the security and privacy tenets established in the previous phases.”

• Software Assurance Maturity Model (SAMM)
  – “Verification is focused on the processes and activities related to how an organization checks and tests artifacts produced throughout software development. This typically includes quality assurance work such as testing, but it can also include other review and evaluation activities.”
1. OPENSAMM
1. MICROSOFT SDL

- **Training**
  - Core training

- **Requirements**
  - Define quality gates/bug bar
  - Analyze security and privacy risk

- **Design**
  - Attack surface analysis
  - Threat modeling

- **Implementation**
  - Specify tools
  - Enforce banned functions
  - Static analysis

- **Verification**
  - Dynamic/Fuzz testing
  - Verify threat models/attack surface

- **Release**
  - Response plan
  - Final security review
  - Release archive

- **Response**
  - Response execution
1. IT’S ABOUT SAVING MONEY!

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

- Microsoft BinScope

- RECX Binary Assurance for Windows
  [http://www.recx.co.uk/products/exeaudit.php](http://www.recx.co.uk/products/exeaudit.php)

- ErrataSec Looking Glass
  [http://blog.erratasec.com/search/label/LookingGlass#.UodWXJ2DN9A](http://blog.erratasec.com/search/label/LookingGlass#.UodWXJ2DN9A)
1. BINScope
1. CURRENT VERIFICATION TOOLS

- Platform specific
  - Windows: BinScope, Looking Glass & Binary Assurance
  - Linux: checksec.sh and custom scripts

- Limited set of checks
  - Check for defenses but what about:
    - Compiler used
    - External libs used
    - Malware
    - You name it...

- Not easy to extend
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. BinSecSweeper
2. WHY BINSEC SWEEPER?

- 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 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.
BINSECWEEPER FEATURES

• 100% in Python

• 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

• Visualizations

• Reporting

• Perform mass binary scanning

Supported files: exe, dll, sys, msi, so, a, scr, cpl, ocx, dry
2. BINSECSWEEPER IN ACTION (I)
2. BINSECSWEEPER IN ACTION (II)
2. BINSECSWEEPER IN ACTION (III)
## 2. CURRENT WINDOWS CHECKS

<table>
<thead>
<tr>
<th>CHECK</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address space layout randomization (ASLR)</td>
<td>Checks if binary has opted the ASLR. Link with /DYNAMICBASE</td>
</tr>
<tr>
<td>Stack Cookies (GS)</td>
<td>Verifies if binary was compiled with Stack Cookies protection. Compile with /GS</td>
</tr>
<tr>
<td>HotPatch</td>
<td>Checks if binary is prepared for hot patching. Compile with /hotpatch</td>
</tr>
<tr>
<td>Compatible with Data Execution Prevention</td>
<td>Validates if binary has opted hardware Data Execution Prevention (DEP). Link with /NXCOMPAT</td>
</tr>
<tr>
<td>Structured Exception Handling (SEH)</td>
<td>Checks if binary was linked with SafeSEH. Link with /SAFESEH</td>
</tr>
<tr>
<td>Adobe Malware Classifier</td>
<td>Analyzes binary for malware behavior using machine learning algorithms</td>
</tr>
<tr>
<td>Visual Studio Compiler Fingerprinting</td>
<td>Identifies if binary was compiled with Visual Studio and version (2008, 2010 &amp; 2012)</td>
</tr>
<tr>
<td>Packer</td>
<td>Checks if binary has been packed</td>
</tr>
<tr>
<td>Insecure API</td>
<td>Check if binary uses banned API</td>
</tr>
<tr>
<td>VM Detection</td>
<td>Check if binaries contains VM detection code</td>
</tr>
</tbody>
</table>
## 2. CURRENT LINUX CHECKS

<table>
<thead>
<tr>
<th>CHECK</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortify Source</td>
<td>Checks if binary was compiled with buffer overflow protection (bounds checking). Compile with <code>-D_FORTIFY_SOURCE=X</code></td>
</tr>
<tr>
<td>Never eXecute (NX)</td>
<td>Verifies if binary was compiled with NX to reduce the area an attacker can use to perform arbitrary code execution.</td>
</tr>
<tr>
<td>Position Independent Code (PIE)</td>
<td>Checks if binary was compiled with PIE to protects against &quot;return-to-text&quot; and generally frustrates memory corruption attacks. Compile with <code>-fPIE -pie</code></td>
</tr>
<tr>
<td>RELocation Read-Only (RELRO)</td>
<td>Validates if binary was compiled with RELRO (partial/full) to harden data sections. Compile with <code>-z,relro,-z,now</code></td>
</tr>
<tr>
<td>Stack Canary</td>
<td>Checks if binary was compiled with stack protector to protect against stack overflows. Compile with <code>-fstack-protector</code></td>
</tr>
</tbody>
</table>
2. 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

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

        self.SetPluginResultsNew(dl)
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))
    else:
        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)
2. BINSECSWEEPER: WHERE?

- Sorry, not yet available!

3. Case Studies & Demos
**BINSEC SWEeper Use Cases**

**Developers**
- Verify product complies with Software Assurance policies before releasing

**IT Pros**
- Can assess software in systems for the security posture

**InfoSec & Researchers**
- Perform file forensics & analyze malware
3. TIME FOR SOME ACTION

- Case Study I: Verify your own software
- Case Study II: Software Security Posture, ACME inc
- Case Study III: Misc.
3. CASE STUDY I: VERIFY YOUR OWN SOFTWARE

• Is your in-house software following a secure development framework?

• Is your software being checked for:

  1. Compiled with a modern compiler?

  2. Security defenses enabled for Windows or Linux?

  3. No malware included in product?

  4. Using external libraries (DLL, etc.) and what is their security?
3. CASE STUDY I: VERIFY YOUR OWN SOFTWARE

- BinSecSweeper can verify that product (used by development teams):
  - What Visual Studio version has been used? (Windows Only) (MS SDL)
  - What defenses have been enabled?:
    | Windows                | Linux                  |
    |------------------------|------------------------|
    | Stack Cookies          | Stack Canary           |
    | ASLR                   | NX                     |
    | DEP                    | Fortify Source         |
    | SAFESEH                | PIE                    |
    | HotPacthing            | RELRO                  |
  - Will audit all files in the project?

- Program security posture: will it Pass / Fail?
3. CASE STUDY II: SOFTWARE SECURITY POSTURE, AMCE INC

• Do IT know the security posture of all software? You can assess your vendors...

• Now you know where EMET is needed!
3. CASE STUDY II: SOFTWARE SECURITY POSTURE, AMCE INC

VLC

- Total Files (305)
  - High (298)
  - Medium (7)
  - OK (7)

SKYME

- Total Files (7)
  - High (2)
  - Medium (5)
  - OK (5)

iTunes

- Total Files (120)
  - High (1)
  - Medium (2)
  - OK (117)

Dropbox

- Total Files (10)
  - High (1)
  - Medium (4)
  - OK (2)
3. CASE STUDY III: ARE YOU COMPILING YOUR APP WITH ZLIB.DLL?

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Title</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Windows Packer Detection</td>
<td>No packer found</td>
</tr>
<tr>
<td>Medium</td>
<td>Windows NXCOMPAT (DEP) Detection</td>
<td>NO NXCOMPAT (DEP) Detected</td>
</tr>
<tr>
<td>INFO</td>
<td>Windows Stack Cookies (GS) Detection</td>
<td>NO Stack Cookie Detected</td>
</tr>
<tr>
<td></td>
<td>Windows Unsafe API</td>
<td>Unsafe API Detected</td>
</tr>
<tr>
<td></td>
<td>Windows ASLR Detection</td>
<td>NO ASLR Detected</td>
</tr>
<tr>
<td></td>
<td>Windows SAFESEH Detection</td>
<td>NO SAFESEH Detected</td>
</tr>
</tbody>
</table>

VULNEX - BinSecSweeper v0.6 - zlib128

**Data:**
- **Filename:** `C:\Users\vuln\Downloads\zlib128-dll\zlib1.dll`
- **File Size:** 107520
- **File Type:** PE32 executable for MS Windows (DLL) (console) Intel 80386 32-bit
- **MD5:** b3a9e911347e7c89440af95470d6e4f7b
- **SHA1:** 3dc9ee3bc0a7e7809131bc0ada0be6527042bfc9
- **SHA256:** 42967a768f341dc9e5182eb38a4867354c3c41379e7d88f4e39cd7354c1fac71
- **Entry Point:** 0x1410
- **Type:** DLL
- **Timestamp:** 1368449328
### 3. CASE STUDY III: ARE YOUR 3\textsuperscript{RD} PARTY COMPONENTS IMPROVING?

**Python 2.7 -> sqlite3.dll**

<table>
<thead>
<tr>
<th>Risk Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile:</td>
<td>Windows NXCOMPAT (DEP) Detection</td>
</tr>
<tr>
<td>Desc:</td>
<td>NO NXCOMPAT (DEP) Detected</td>
</tr>
</tbody>
</table>

**Python 3.3 -> sqlite3.dll**

<table>
<thead>
<tr>
<th>Risk Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile:</td>
<td>Windows ASLR Detection</td>
</tr>
<tr>
<td>Desc:</td>
<td>NO ASLR Detected</td>
</tr>
</tbody>
</table>
# 3. CASE STUDY III: A DLL INSIDE A WELL-KNOWN SOFTWARE

<table>
<thead>
<tr>
<th>Risk Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile:</td>
<td>Windows ASLR Detection</td>
</tr>
<tr>
<td>Desc:</td>
<td>NO ASLR Detected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile:</td>
<td>Windows NXCOMPAT (DEP) Detection</td>
</tr>
<tr>
<td>Desc:</td>
<td>NO NXCOMPAT (DEP) Detected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile:</td>
<td>Windows Unsafe API</td>
</tr>
<tr>
<td>Desc:</td>
<td>Unsafe API Detected</td>
</tr>
</tbody>
</table>

**Potential Unsafe API (14):**
- 0x00407170: `strcpy`
- 0x004071bc: `strncpy`
- 0x00407168: `strcat`
- 0x00407164: `strncat`
- 0x00407244: `wprintfA`
- 0x004071d8: `printf`
- 0x004071c8: `_vsnprintf`
- 0x004071b8: `_snprintf`
- 0x004071c8: `_vsnprintf`
- 0x004071bc: `strncpy`
- 0x00407164: `strncat`
- 0x004071cc: `sscanf`
- 0x0040716c: `strlen`
- 0x0040715c: `memcpy`
3. CASE STUDY III: THE MOST COMMON WORD INSIDE A MICROSOFT BINARY?

### Total N-Grams

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1208</td>
<td>2150</td>
<td>2464</td>
<td>2535</td>
<td>2560</td>
<td>2557</td>
<td>2516</td>
<td>2452</td>
<td>2376</td>
</tr>
</tbody>
</table>

### Top 10 N-Grams

<table>
<thead>
<tr>
<th>2-gram</th>
<th>Frequency</th>
<th>3-gram</th>
<th>Frequency</th>
<th>4-gram</th>
<th>Frequency</th>
<th>5-gram</th>
<th>Frequency</th>
<th>6-gram</th>
<th>Frequency</th>
<th>7-gram</th>
<th>Frequency</th>
<th>8-gram</th>
<th>Frequency</th>
<th>9-gram</th>
<th>Frequency</th>
<th>10-gram</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>119</td>
<td>ion</td>
<td>68</td>
<td>tion</td>
<td>58</td>
<td>croso</td>
<td>44</td>
<td>crosof</td>
<td>44</td>
<td>crosof</td>
<td>44</td>
<td>icrsoft</td>
<td>44</td>
<td>Microsoft</td>
<td>25</td>
<td>Microsoft</td>
<td>21</td>
</tr>
<tr>
<td>ti</td>
<td>111</td>
<td>tio</td>
<td>58</td>
<td>soft</td>
<td>44</td>
<td>osoft</td>
<td>44</td>
<td>rosoft</td>
<td>44</td>
<td>icrsoft</td>
<td>44</td>
<td>icrsoft</td>
<td>44</td>
<td>Microsoft</td>
<td>25</td>
<td>icrsoft</td>
<td>21</td>
</tr>
<tr>
<td>et</td>
<td>78</td>
<td>oft</td>
<td>44</td>
<td>cros</td>
<td>44</td>
<td>icros</td>
<td>44</td>
<td>icrsoft</td>
<td>44</td>
<td>Microso</td>
<td>25</td>
<td>crosoft</td>
<td>21</td>
<td>microsoft</td>
<td>19</td>
<td>crossoft</td>
<td>14</td>
</tr>
<tr>
<td>ic</td>
<td>69</td>
<td>et_</td>
<td>44</td>
<td>icrco</td>
<td>44</td>
<td>rosof</td>
<td>44</td>
<td>Micros</td>
<td>25</td>
<td>rosoft</td>
<td>21</td>
<td>microsf</td>
<td>19</td>
<td>Attribute</td>
<td>15</td>
<td>icrsoftC</td>
<td>14</td>
</tr>
<tr>
<td>io</td>
<td>68</td>
<td>cro</td>
<td>44</td>
<td>roso</td>
<td>44</td>
<td>ation</td>
<td>38</td>
<td>Config</td>
<td>22</td>
<td>microso</td>
<td>19</td>
<td>attribute</td>
<td>15</td>
<td>mitigation</td>
<td>14</td>
<td>t Corporat</td>
<td>11</td>
</tr>
<tr>
<td>at</td>
<td>66</td>
<td>get</td>
<td>44</td>
<td>osot</td>
<td>44</td>
<td>Microc</td>
<td>25</td>
<td>osot</td>
<td>21</td>
<td>tribut</td>
<td>15</td>
<td>Attribute</td>
<td>15</td>
<td>Mitigation</td>
<td>14</td>
<td>rosoft Cor</td>
<td>11</td>
</tr>
<tr>
<td>in</td>
<td>64</td>
<td>ros</td>
<td>44</td>
<td>get_</td>
<td>43</td>
<td>Confi</td>
<td>22</td>
<td>micros</td>
<td>19</td>
<td>tribute</td>
<td>15</td>
<td>itigati</td>
<td>14</td>
<td>crossoftC</td>
<td>14</td>
<td>oft Corpor</td>
<td>11</td>
</tr>
<tr>
<td>Co</td>
<td>64</td>
<td>oso</td>
<td>44</td>
<td>atio</td>
<td>38</td>
<td>config</td>
<td>22</td>
<td>ration</td>
<td>15</td>
<td>Attribu</td>
<td>15</td>
<td>mitigation</td>
<td>14</td>
<td>rosoft Co</td>
<td>14</td>
<td>corporation</td>
<td>11</td>
</tr>
<tr>
<td>ro</td>
<td>62</td>
<td>sof</td>
<td>44</td>
<td>Micr</td>
<td>25</td>
<td>soft</td>
<td>21</td>
<td>tribut</td>
<td>15</td>
<td>Mitigat</td>
<td>14</td>
<td>Mitigation</td>
<td>14</td>
<td>ft Corpor</td>
<td>11</td>
<td>ft Corpora</td>
<td>11</td>
</tr>
</tbody>
</table>
4. Conclusions
4. 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 (show you care about security)

• How is the security posture of software vendors you use?
4. Q&A

- Thanks!

- @BinSecSweeper

- @simonroses | @vulnexsl

- www.vulnex.com