Where is my Code Vulnerable: Matching CVEs and Source Code

David A. Barrett & Peter Shin, Canvass Labs

david.barrett@canvasslabs.com  https://www.canvasslabs.com
peter@canvasslabs.com
Motivation

• Modern software depends upon third-party or open source software (or prior versions of their own proprietary software)

• Introduces maintenance costs for developers (or security officers)
  • Tracking dependencies
  • Monitor dependencies for newly-discovered vulnerabilities
  • Applying patches for updates
Solution

• Create Databases:
  - Find vulnerable code in open source
  - Find package names/versions in NVD

• Scan your source code to find vulnerabilities

• Report the vulnerable code and its fix.

1. GitHub
   - Extract Source Code Signatures
   - NVD
   - Extract Package Names & versions

2. Software Library
3. Dependency Management System
4. Match Source Code
5. Match Vulnerabilities
6. Vulnerability List & Fixes
Map NVD Description to Package Names and Versions
Mapping NVD Description (example)

CVE-2003-1045 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

votes.cgi in Bugzilla 2.16.3 and earlier, and 2.17.1 through 2.17.4, allows remote attackers to read a user’s voting page when that user has voted on a restricted bug, which allows remote attackers to read potentially sensitive voting information by modifying the who parameter.

Source: MITRE

View Analysis Description
votes.cgi in **Bugzilla 2.16.3 and earlier, and 2.17.1 through 2.17.4**, allows remote attackers to read a user's voting page when that user has voted on a restricted bug, which allows remote attackers to read potentially sensitive voting information by modifying the who parameter.
Keyword Tagging

Bugzilla 2.16.3 and earlier, and 2.17.1 through 2.17.4

product name (pn)
version range end (vre)

= < separator (sp)

separator (sp) version range start (vrs)

<= version range end (vre)
Keyword Interpretation

**Bugzilla <= 2.16.3**

and

**2.17.1 <= Bugzilla <= 2.17.4**
Example 2

CVE-2009-1232 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

Mozilla Firefox 3.0.8 and earlier 3.0.x versions allows remote attackers to cause a denial of service (memory corruption) via an XML document composed of a long series of start-tags with no corresponding end-tags. NOTE: it was later reported that 3.0.10 and earlier are also affected.

Source: MITRE

View Analysis Description

QUICK INFO

CVE Dictionary Entry:
CVE-2009-1232
NVD Published Date:
04/02/2009
NVD Last Modified:
09/28/2017
Mozilla Firefox 3.0.8 and earlier 3.0.x versions allows remote attackers to cause a denial of service (memory corruption) via an XML document composed of a long series of start-tags with no corresponding end-tags. NOTE: it was later reported that 3.0.10 and earlier are also affected.
Keyword Tagging (example 2)

Mozilla Firefox 3.0.8 and earlier 3.0.x

\[ pn \quad pn \quad vre = < vr \]

... 

3.0.10 and earlier

\[ vre = < \]
3.0.x <= Mozilla Firefox <= 3.0.8

and

Mozilla Firefox <= 3.0.1
Example 3

CVE-2017-5948 Detail

Current Description
An issue was discovered on OnePlus One, X, 2, 3, and 3T devices. OxygenOS and HydrogenOS are vulnerable to downgrade attacks. This is due to a lenient ‘updater-script’ in OTAs that does not check that the current version is lower than or equal to the given image’s. Downgrades can occur even on locked bootloaders and without triggering a factory reset, allowing for exploitation of now-patched vulnerabilities with access to user data. This vulnerability can be exploited by a Man-in-the-Middle (MITM) attacker targeting the update process. This is possible because the update transaction does not occur over TLS (CVE-2016-10370). In addition, a physical attacker can reboot the phone into recovery, and then use ‘adb sideload’ to push the OTA (on OnePlus 3/3T ‘Secure Start-up’ must be off).

Source: MITRE

View Analysis Description
An issue was discovered on OnePlus One, X, 2, 3, and 3T devices. OxygenOS and HydrogenOS are vulnerable to downgrade attacks. This is due to a lenient 'updater-script' in OTAs that does not check that the current version is lower than or equal to the given image's. Downgrades can occur even on locked bootloaders and without triggering a factory reset, allowing for exploitation of now-patched vulnerabilities with access to user data. This vulnerability can be exploited by a Man-in-the-Middle (MiTM) attacker targeting the update process. This is possible because the update transaction does not occur over TLS (CVE-2016-10370). In addition, a physical attacker can reboot the phone into recovery, and then use 'adb sideload' to push the OTA (on OnePlus 3/3T 'Secure Start-up' must be off).
Keyword Tagging (example 3)

OnePlus One, X, 2, 3, and, 3T

OxygenOS and HydrogenOS
Keyword Interpretation (example 3)

Vulnerable products:

- OnePlus One X
- OnePlus One 2
- OnePlus One 3
- OnePlus One 3T
- OxygenOS
- HydrogenOS
CVE-2012-1463 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

The ELF file parser in AhnLab V3 Internet Security 2011.01.18.00, Bitdefender 7.2, Quick Heal (aka Cat QuickHeal) 11.00, Command Antivirus 5.2.11.5, Comodo Antivirus 7424, eSafe 7.0.17.0, F-Prot Antivirus 4.6.2.117, F-Secure Anti-Virus 9.0.16160.0, McAfee Anti-Virus Scanning Engine 5.400.0.1158, Norman Antivirus 6.06.12, nProtect Anti-Virus 2011-01-17.01, and Panda Antivirus 10.0.2.7 allows remote attackers to bypass malware detection via an ELF file with a modified endianness field. NOTE: this may later be SPLIT into multiple CVEs if additional information is published showing that the error occurred independently in different ELF parser implementations.

Source: MITRE
+View Analysis Description
The ELF file parser in Command Antivirus 5.2.11.5, Comodo Antivirus 7424, eSafe 7.0.17.0, F-Prot Antivirus 4.6.2.117, F-Secure Anti-Virus 9.0.16160.0, McAfee Anti-Virus Scanning Engine 5.400.0.1158, Norman Antivirus 6.06.12, nProtect Anti-Virus 2011-01-17.01, and Panda Antivirus 10.0.2.7 allows remote attackers to bypass malware detection via . . .
Keyword Tagging (example 4)

Command Antivirus 5.2.11.5, Comodo Antivirus

7424, eSafe 7.0.17.0, F-Prot Antivirus 4.6.2.117,

F-Secure Anti-Virus 9.0.16160.0, McAfee Anti-Virus

Scanning Engine 5.400.0.1158, Norman Antivirus

6.06.12, nProtect Anti-Virus 2011-01-17.01, and

Panda Antivirus 10.0.2.7
Keyword Interpretation (example 4)

- Command Antivirus 5.2.11.5
- Comodo Antivirus 7424
- eSafe 7.0.17.0
- F-Prot Antivirus 4.6.2.117
- F-Secure Anti-Virus 9.0.16160.0
- McAfee Anti-Virus Scanning Engine 5.400.0.1158
- Norman Antivirus 6.06.12
- nProtect Anti-Virus 2011-01-17.01
- Panda Antivirus 10.0.2.7
The original distribution of MyBulletinBoard (MyBB) to update from older versions to 1.0.2 omits or includes older versions of certain critical files, which allows attackers to conduct (1) SQL injection attacks via an attachment name that is not properly handled by inc/functions_upload.php (CVE-2005-4602), and possibly (2) other attacks related to threadmode in usercp.php.
Finding Vulnerable Code
# A Vulnerability and its Fix

## Patch code:

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>Hunk: Lines 88-94 (previously 88-91)</td>
<td>*/ remember the value we stored into this reg */</td>
</tr>
<tr>
<td>89</td>
<td><code>regs[insn-&gt;dst_reg].type = SCALAR_VALUE;</code></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>- <code>__mark_reg_known(regs + insn-&gt;dst_reg, insn-&gt;imm);</code></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>+ if (BPF_C+LASS(insn-&gt;code) == BPF_ALU64) {</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td><code>__mark_reg_known(regs + insn-&gt;dst_reg, insn-&gt;imm);</code></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>} else {</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>+ <code>__mark_reg_known(regs + insn-&gt;dst_reg, (u32)insn -&gt;imm);</code></td>
<td></td>
</tr>
</tbody>
</table>

**Package:** linux

**File:** `kernel/bpf/verifier.c`

**CVE ID:** CVE-2017-16995

**Repository:** `/kernel`

**Function:** `check_alu_op`

**CWE_ID:** CWD-119
int check_alu_op(struct bpf_verifier_env* env, struct bpf_insn*_env insn) {
    struct bpf_reg_state *regs = cur_regs(env)
    ...

    regs[insn->dst_reg].type = SCALAR_VALUE;
    __mark_reg_known(
        regs + insn->dst_reg, insn->imm);
}
We Use its Abstract Syntax

```c
TYPE check_alu_op(TYPE PARM,
    TYPE PARM) {
    TYPE LOCAL_VAR = FUNC_CALL(PARM)
    ...
    LOCAL_VAR[PARM] = SCALAR_VALUE;
    FUNC_CALL(
        LOCAL_VAR + PARM, PARM);
}
```
Vulnerable Source Code

```c
int check_alu_op(struct bpf_verifier_env* env, struct bpf_insn*_env insn) {
    struct bpf_reg_state *regs = cur_regs(env)
    ...

    regs[insn->dst_reg].type = SCALAR_VALUE;
    __mark_reg_known(
        regs + insn->dst_reg, insn->imm);
}
```
int check_alu_op(struct bpf_verifier_env* env, struct bpf_insn*_env my_insn) {
    struct bpf_reg_state *regs = cur_regs(env)
    ...
    regs[my_insn->dst_reg].type = SCALAR_VALUE;
    __mark_reg_known(
        regs + my_insn->dst_reg, my_insn->imm);
}

And show you it is vulnerable and how to fix it
Example Result
Example Result: Code Matched 3 Vulnerabilities

Source Code:
/* ssl/t1_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscape SSL
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to. The following conditions
 * apply to all code found in this distribution.
 * 
 * Redistributions of this distribution are made available subject to the
 * conditions listed below.
 * 
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 
 * 1. Redistributions of source code must retain the copyright
 * notices, this list of conditions and the following disclaimer.
 * 
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in the
 * documentation and/or other materials provided with the distribution.
 * 
 * 3. All advertising materials mentioning features or use of this
 * software must display the following acknowledgement:
 * "This product includes software developed by the Free Software
 * Foundation; see http://www.fsf.org for more information.""
Example Result: A Matched Vulnerability And Its Fix

**Package name:** openssl

**Repository path:** /ssl/t1_lib.c

**File name:** t1_lib.c

**Function name:** tls1_process_heartbeat

**CVE ID:** CVE-2014-0160

**CVSS Score:** 5.0

**CWE ID:** CWE-119

**Patch Code:**

```c
16 11  
17 12  
18 13  
14 +  /* Read type and payload length first */
15 +  if (1 + 2 + 16 > s->s3->rrec.length)
16 +  return 0; /* silently discard */
17 +  hbtype = *p++;
18 +  n2s(p, payload);
19 +  if (1 + 2 + payload + 16 > s->s3->rrec.length)
20 +  return 0; /* silently discard per RFC 6520 sec. 4 */
21 +  pl = p;
22 +
19 23  
```

**Hunk:** Lines 6-25 (previously 6-21)

<table>
<thead>
<tr>
<th>Line</th>
<th>Original Code</th>
<th>Modified Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>unsigned int payload;</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>unsigned int padding = 16; /* Use minimum padding */</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>/* Read type and payload length first */</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>hbtype = *p++;</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>n2s(p, payload);</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>pl = p;</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>if (s-&gt;msg_callback)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>s-&gt;msg_callback(0, s-&gt;version, TLS1_RT_HEARTBEAT,</td>
<td></td>
</tr>
</tbody>
</table>

**Patch Code:**

```c
16 11  
17 12  
18 13  
14 +  /* Read type and payload length first */
15 +  if (1 + 2 + 16 > s->s3->rrec.length)
16 +  return 0; /* silently discard */
17 +  hbtype = *p++;
18 +  n2s(p, payload);
19 +  if (1 + 2 + payload + 16 > s->s3->rrec.length)
20 +  return 0; /* silently discard per RFC 6520 sec. 4 */
21 +  pl = p;
22 +
19 23  
```
Example Result: Versions That Are Vulnerable

Package name: openssl
Repository path: /
File name: ssl/tls1_lib.c
Function name: tls1_process_heartbeat
CVE ID: CVE-2014-0160
CVSS Score: 5.0
CWE ID: CWE-119

<table>
<thead>
<tr>
<th>Version</th>
<th>Function lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenSSL_1_0_1c</td>
<td>2436 2503</td>
</tr>
<tr>
<td>OpenSSL_1_0_2-beta1</td>
<td>3789 3856</td>
</tr>
<tr>
<td>OpenSSL_1_0_1b</td>
<td>2436 2503</td>
</tr>
<tr>
<td>OpenSSL_1_0_1f</td>
<td>2553 2620</td>
</tr>
<tr>
<td>OpenSSL_1_0_1e</td>
<td>2481 2548</td>
</tr>
<tr>
<td>OpenSSL_1_0_1</td>
<td>2436 2503</td>
</tr>
<tr>
<td>OpenSSL_1_0_1d</td>
<td>2473 2540</td>
</tr>
<tr>
<td>OpenSSL_1_0_1a</td>
<td>2436 2503</td>
</tr>
</tbody>
</table>
## Popular repositories

<table>
<thead>
<tr>
<th>Repository Name</th>
<th>Description</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mavenDemo</td>
<td>Maven demo project</td>
<td>Java</td>
</tr>
<tr>
<td>npmDemo</td>
<td>NPM demo project</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Bert-on-CVE</td>
<td>Jupyter Notebook</td>
<td></td>
</tr>
<tr>
<td>canvass_for_security-dependency_checker</td>
<td>Finds vulnerabilities in dependent software packages</td>
<td>Python</td>
</tr>
<tr>
<td>canvass_for_security-sample_vuin_db</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Where is my Code Vulnerable: Matching CVEs and Source Code

David A. Barrett & Peter Shin, Canvass Labs

david.barrett@canvasslabs.com  https://www.canvasslabs.com
peter@canvasslabs.com