How Can You Trust Formally Verified Software?

Alastair Reid

Arm Research
@alastair_d_reid
Software

US aviation authority: Boeing 787 bug could cause 'loss of control'

More trouble for Dreamliner as Federal Aviation Administration warns glitch in control unit causes generators to shut down if left powered on for 248 days

The Boeing 787 has four generator-control units that, if powered on at the same, could fail simultaneously and cause a complete electrical shutdown. Photograph: Elaine Thompson/AP

Buffer over-read vulnerabilities

 CVE-2014-0160 Detail

Modified

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis, which may result in further details.

Current Description

The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

Source: MITRE    Last Modified: 04/07/2014    View Analysis Description
Buffer over-read vulnerabilities

Current

The (1) TLS and (2) SSL protocol implementation in Cisco IOS 12.0 through 12.4.1 allows remote attackers to execute arbitrary code via a crafted XML file, related to "error handling logic." CVE-2008-4059

CVE-2008-3803

A "logic error" in Cisco IOS 12.0 through 12.4, when a Multiprotocol Label Switching (MPLS) VPN with extended communities is configured, sometimes causes a corrupted route target (RT) to be used, which allows remote attackers to read traffic from other VPNs in opportunistic circumstances.

CVE-2007-4613

A buffer over-read vulnerability exists due to the improper handling of input that may result in a stack buffer overflow in the XMLRPC function that is present in the XML-RPC WebLogic Server 6.0.1 Gold through SP7. CVE-2007-2998

CVE-2007-2998

The libkrl library in Sun Solaris 9 before 20070617, in Solaris 10 through 8, and in SunOS 5.9 through SP5 might allow remote attackers to cause a denial of service (linked daemon deadlock) because of an unhandled assertion, which might overlap CVE-2006-2298.

CVE-2007-2444

Logic error in the Samba 3.0.23d through 3.0.23pre2 allows local users to gain temporary privileges and execute commands as root.

CVE-2007-0414

Logic error in the Samba 3.0.23d through 3.0.23pre2 allows local users to gain temporary privileges and execute commands as root.

CVE-2006-4394

Logic error in LogWindow in Apple Mac OS X 10.4 through 10.4.7, allows network accounts without GUIDs to bypass service access controls and log into the system using loginwindow using unknown vectors.

CVE-2006-0381

A logic error in the IP fragment cache functionality in pf in FreeBSD 5.3, 5.4, and 6.0, and OpenBSD, when a "scrub fragment drop-off" rule is being used, allows remote attackers to cause a denial of service (crash) via crafted packets that cause a packet fragment to be inserted twice.

CVE-2005-2380

A logic error in FreeBSD kernel 5.4-STABLE and 6.0 causes the kernel to calculate an incorrect buffer length, which causes more data to be copied to userspace than intended, which could allow local users to read portions of kernel memory.

CVE-2005-0198

A logic error in the CRAM-MD5 code for the University of Washington IMAP (UW-IMAP) server, when Challenge-Response Authentication Mechanism with MD5 (CRAM-MD5) is enabled, does not properly enforce all the required conditions for successful authentication, which allows remote attackers to authenticate as arbitrary users.

Source: MITRE       Last Modified: 04/07/2014  + View Analysis Description

MTA: Mail transfer agent (MTA)
MTA: Mail transfer agent (MTA)
MTA: Mail transfer agent (MTA)
MTA: Mail transfer agent (MTA)
MTA: Mail transfer agent (MTA)
Buffer over-read vulnerabilities

CVE-2008-3803
A "logic error" in Cisco IOS 12.0 through 12.4, when a Multiprotocol Label Switching (MPLS) VPN with extended communities is configured, sometimes causes a corrupted route target (RT) to be used, which allows remote attackers to read traffic from other VPNs in opportunistic circumstances.

CVE-2008-0059
Race condition in NSXML in Foundation for Apple Mac OS X 10.4.11 allows context-dependent attackers to execute arbitrary code via a crafted XML file, related to "error handling logic."

CVE-2007-4613
SSL libraries in BEA WebLogic Server 6.1 Gold through 7.0 Gold through 8.1.10 before patches to the BEA-02-05 vulnerability might overlap a buffer over-read vulnerability.

CVE-2007-2998
The libkine in Sun Solaris 9 before 20070501 might cause a denial of service (invalid daemon access) via the in_kequeue() function.

CVE-2007-2444
Logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2007-2414
Logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2006-4934
Logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2006-2989
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2005-014
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2005-824
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9988
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9989
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9996
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9998
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9995
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

CVE-2002-9994
A logic error in the Solaris kernel's service (shell out) function causes a denial of service (null pointer dereference) attack.

Null pointer dereference

The readBufInt32 function in util/read.c in libming 0.4.8 mishandles memory allocation. A crafted input will lead to a remote denial of service (null pointer dereference) attack.

The ltm function in service (shell out) function causes a denial of service (null pointer dereference) attack.

The snd_msgqueue() function in service (shell out) function causes a denial of service (null pointer dereference) attack.

The snd_msgqueue() function in service (shell out) function causes a denial of service (null pointer dereference) attack.

The snd_msgqueue() function in service (shell out) function causes a denial of service (null pointer dereference) attack.

The snd_msgqueue() function in service (shell out) function causes a denial of service (null pointer dereference) attack.

The snd_msgqueue() function in service (shell out) function causes a denial of service (null pointer dereference) attack.

Null pointer dereference vulnerabilities

SAP NetWeaver AS ABAP 7.40 allows remote authenticated users with certain privileges to cause a denial of service (process crash) via vectors involving disp-work.exe, aka SAP Security Note 2406241.

In all Qualcomm products with Android releases from CAF using the Linux kernel, if a pointer argument coming from userspace is invalid, a driver may use an uninitialized structure to log an error message.

A Null Pointer Dereference issue was discovered in Schneider Electric Wonderware ArchestrA Logger, versions 17.20.426.2307.1 and prior. The null pointer dereference vulnerability could allow an attacker to crash the logger service.

A denial of service for logging and log-viewing (applications that use the Wonderware ArchestrA Logger service is unavailable).
There is an invalid free in Image::printFDStructure that leads to a Segmentation fault in Exiv2 0.26. A crafted argument input will lead to a remote denial of service attack.

In LibTIFF 4.0.8, there is a heap-based buffer overflow in the tiff_write_pdf function in tools/tiff2pdf.c. This heap overflow could lead to different damages. For example, a crafted TIFF document can lead to an out-of-bounds read in TIFFCleanup, an invalid free in TIFFClose or tiff_free, memory corruption in 12p_readwrite_pdf_image, or a double free in 12p_free. Given these possibilities, it probably could cause arbitrary code execution.

When under stress, closing many connections, the HTTP/2 handling code in Apache httpd 2.4.26 would result in potentially erratic behaviour.

In all Qualcomm products with Android releases from CAF using the Linux kernel, a race condition in a WLAN driver can lead to a Use After Free condition.

In all Qualcomm products with Android releases from CAF using the Linux kernel, a race condition in a USB driver functions can lead to a Use After Free condition.

K3Vault MySQL Free Knowledge Base application package 0.16a comes with a FileExplorer/Explorer.aspx script to Upload/Download/Uploads file-management component. An unauthenticated user can access the file upload and deletion functionality. Through this functionality, a user can upload an ASPX script to Uploads/Documents/ to run any arbitrary code.

The lfs_setlink function (over denial of service) of the message queue head pointer can lead to a Use After Free condition by changing the value of a message queue head pointer. In a process crash via vectors involving disp-work.exe, aka SAP Security Note 2406041, SAP NetWeaver AS ADAB 7.40 allows remote authenticated users with certain privileges to cause a denial of service (over boundary access). A file pointer Dereference Vulnerability was discovered in Schneider Electric Wonderware ArchestrA Logger, versions from 2017.426.3307.1 and prior. The null pointer Dereference vulnerability could allow an attacker to crash the logger using a denial of service for logging and log-viewing (applications that use the Wonderware ArchestrA Logger service is unavailable).
Buffer overflow vulnerabilities

- **CVE-2017-9953**: There is an invalid free in Image::printFFDS::Structure that leads to a segmentation fault in Exiv2 0.26. A crafted, unconfgured, arbitrary code execution via a

- **CVE-2017-9935**: In LibTIFF 4.0.8, there is a heap-based buffer overflow in the t2p_write_pdf function in tools/tiffpdf.c. This

- **CVE-2017-9788**: A Logic error vulnerability in TiffClose or t2p_free, memory corruption in the boundaries read in TIFFCleanup, an invalid free in TIFFClose or t2p_free, memory corruption in t2p_readwrite_pdf_image, or a double free in t2p_free. Given these possibilities, it's probably could cause

- **CVE-2017-9685**: When under stress, closing many connections, the HTTP/2 handler driver can lead to

- **CVE-2017-9684**: In all Qualcomm products, the driver can lead to

- **CVE-2017-9682**: In all Qualcomm products, the driver can lead to

- **CVE-2017-9992**: Heap-based buffer overflow in the decode_ddsl function in libavcodec/dfa.c in FFmpeg

- **CVE-2017-9991**: Heap-based buffer overflow in the xwd_decode_frame function in libavcodec/xwddec.c in FFmpeg 3.3 before 3.3.1 allows remote attackers to cause a denial of service (application crash or possibly

- **CVE-2017-9990**: Stack-based buffer overflow in FFmpeg 3.3 before 3.3.1 allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file.

- **CVE-2017-9987**: There is a heap-based buffer overflow in the function hpe1_motion in mpegvideo_motion.c in libav 12.1. A crafted input can lead to a remote denial of service attack.

- **CVE-2017-9948**: A stack buffer overflow vulnerability has been discovered in Microsoft Skype 7.2, 7.35, and

- **CVE-2017-9947**: A stack buffer overflow vulnerability has been discovered in Microsoft Skype 7.2, 7.35, and

- **CVE-2017-9668**: In all Qualcomm products with Android releases from QPL 2.2 to QPL 2.6.1, an unconfgured, arbitrary code execution via a

- **CVE-2017-9631**: A Null Pointer Dereference issue was discovered in Schneider Electric Wonderware ArchestrA Logger, versions from userspace is invalid, a driver may use an uninitialized structure to log an error message.
Formal verification

- Of libraries and apps
  - Verified Software Toolchain
  - deepspec cryptos

- Of compilers
  - COMPCERT
  - Vellvm
  - CAKEML

- Of operating systems
  - seL4
  - CertiKOS
Fonseca et al., An Empirical Study on the Correctness of Formally Verified Distributed Systems, Eurosys ‘17
Formally Verified Software

- Verification Tool
- Shim Code
- Formal Specifications

Fonseca et al., An Empirical Study on the Correctness of Formally Verified Distributed Systems, Eurosys ‘17
Takeaway #1: 3 key questions to ask

1. What specifications does your proof rely on?
2. Why do you trust those specifications?
3. Does anybody else use these specifications?
Takeaway #2: Specifications must have multiple uses
Takeaway #2: Specifications must have multiple uses
How can you trust formally verified software?

How can you trust formally verified software?

Specifications are part of the TCB

3 key questions

Specifications must have multiple users

How can you trust formal specifications?

Testing specifications

Verifying processors

Verifying specifications

How can you trust formally verified software?
Creating trustworthy specifications
The state of most processor specifications

Large (1000s of pages)
Broad (10+ years of implementations, multiple manufacturers)
Complex (exceptions, weak memory, …)
Informal (mostly English prose)

We are all just learning how to (retrospectively) formalize specifications
## Arm Processor Specifications

**A-class** *(phones, tablets, servers, ...)*  
6,000 pages  
40,000 line formal specification  
- Instructions (32/64-bit)  
- Exceptions / Interrupts  
- Memory protection  
- Page tables  
- Multiple privilege levels  
- System control registers  
- Debug / trace  

**M-class** *(microcontrollers, IoT)*  
1,200 pages  
15,000 line formal specification  
- Instructions (32-bit)  
- Exceptions / Interrupts  
- Memory protection  
- Page tables  
- Multiple privilege levels  
- System control registers  
- Debug / trace
Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.

Entry to lockup from an exception causes:

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending or active.
- The PC to be set to 0xFFFFFFFF.
- EPSR.IT to be become UNKNOWN.

In addition, HFSR.FORCED is not set to 1.
Encoding A1
ARMv4*, ARMv5T*, ARMv6*, ARMv7
ADC{S}<c>, <Rd>, <Rn>, <Rm>{, <shift>}

<table>
<thead>
<tr>
<th>cond</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>S</th>
<th>Rn</th>
<th>Rd</th>
<th>imm5</th>
<th>type</th>
<th>0</th>
<th>Rm</th>
</tr>
</thead>
</table>

if Rd == '1111' && S == '1' then SEE SUBS PC, LR and related instructions;
d = Uint(Rd);  n = Uint(Rn);  m = Uint(Rm);  setflags - (S == '1');
(shift_t, shift_n) = DecodeImmShift(type, imm5);

if ConditionPassed() then
  EncodingSpecificOperations();
shfted - Shift(R[m], shift_t, shift_n, APSR.C);
(result, carry, overflow) = AddWithCarry(R[n], shifted, APSR.C);
if d == 15 then  // Can only occur for ARM encoding
  ALUWritePC(result);  // setflags is always FALSE here
else
  R[d] = result;
  if setflags then
    APSR.N = result<31>;
    APSR.Z = IsZeroBit(result);
    APSR.C = carry;
    APSR.V = overflow;
Arm Architecture Specification Language (ASL)

Indentation-based syntax

Imperative

First-order

Strongly typed (type inference, polymorphism, dependent types)
  - Bit-vectors
  - Unbounded integers
  - Infinite precision reals
  - Arrays, Records, Enumerations

Exceptions
Architectural Conformance Suite

Processor architectural compliance sign-off

Large

• v8-A 11,000 test programs, > 2 billion instructions
• v8-M 3,500 test programs, > 250 million instructions

Thorough

• Tests dark corners of specification
Testing Pass Rate
(Artists Impression)
Measuring architecture coverage of tests

Untested: op1*op2 == -3.0, FPCR.RND=-Inf

```c
bits(N) FPRSqrtStepFused(bits(N) op1, bits(N) op2)
    assert N IN (32, 64);
    bits(N) result;
    op1 = FPNeg(op1); // per FMSUB/FMLS
    (type1,sign1,value1) = FPUnpack(op1, FPCR);
    (type2,sign2,value2) = FPUnpack(op2, FPCR);
    (done,result) = FPProcessNaNs(type1, type2, op1, op2, FPCR);
    if !done then
        inf1 = (type1 == FPType_Infinity);
        inf2 = (type2 == FPType_Infinity);
        zero1 = (type1 == FPType_Zero);
        zero2 = (type2 == FPType_Zero);
        if (inf1 && zero2) || (zero1 && inf2) then
            result = FPOnePointFive("0");
        else if inf1 || inf2 then
            result = FPInfinity(sign1 EOR sign2, N);
        else
            // Fully fused multiply-add and halve
            result_value = (3.0 + (value1 * value2)) / 2.0;
            if result_value == 0.0 then
                // Sign of exact zero result depends on rounding mode
                sign = if FPCR.Rounding() == FPRounding_NEGINF then '1' else '0';
                result = FPZero(sign, N);
            else
                result = FPRound(result_value, FPCR.Rounding());
            return result;
    return result;
```
Formal verification of processors
Checking an instruction

ADD
Checking an instruction

Context

CMP LDR  ADD  STR BNE
Memory

R0 - R15
ARM Research

25

Architecture Specification

ASL to Verilog

Combinational Verilog

Specialize
Monomorphize
Constant Propagation
Width Analysis
Exception Handling
...

The Architecture for the Digital World®
Arm CPUs verified with ISA-Formal

<table>
<thead>
<tr>
<th>A-class</th>
<th>R-class</th>
<th>M-class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-A53</td>
<td>Cortex-R52</td>
<td>Cortex-M4</td>
</tr>
<tr>
<td>Cortex-A32</td>
<td>Next generation</td>
<td>Cortex-M7</td>
</tr>
<tr>
<td>Cortex-A35</td>
<td></td>
<td>Cortex-M33</td>
</tr>
<tr>
<td>Cortex-A55</td>
<td></td>
<td>Next generation</td>
</tr>
<tr>
<td>Next generation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rolling out globally to other design centres

Sophia, France - Cortex-A75 (partial)
Austin, USA - TBA
Chandler, USA - TBA

Cambridge Projects
Formal validation of specifications
Suppose…

Last year: audited all accesses to privileged registers
- Specification: Added missing privilege checks
- Testsuite: Added new tests to test every privilege check
- Formal testbench: Verify every check

This year: add new instruction but accidentally omit privilege check

How many tests in the test suite will fail on new specification?
Executable Specification

Defines what *is* allowed

- Animation → Check spec matches expectation
- Testable → Compare spec against implementation
Executable Specification

Defines what *is* allowed
  - Animation  →  Check spec matches expectation
  - Testable   →  Compare spec against implementation

Does *not* define what *is not* allowed
  - e.g., Impossible states, impossible actions/transitions, security properties
  - No redundancy
  - Problem when extending specification
Creating a specification of disallowed behaviour

Where to get a list of disallowed behaviour?
How to formalise this list?
How to formally validate specification against spec of disallowed behaviour?
Rule JRJC

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority processor exception.
Rule R

State Change X by any of the following:

- Event A
- Event B
- State Change C
- Event D
Rule $R$

State Change $X$ by any of the following:

- Event A
- Event B
- State Change C
- Event D

And cannot happen any other way
Rule R

State Change X by any of the following:

• Event A
• Event B
• State Change C
• Event D

And cannot happen any other way

Rule R:  \[ X \rightarrow A \lor B \lor C \lor D \]
<table>
<thead>
<tr>
<th>State Change X</th>
<th>Exit from lockup</th>
<th>Fell(LockedUp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event A</td>
<td>A Cold reset</td>
<td>Called(TakeColdReset)</td>
</tr>
<tr>
<td>Event B</td>
<td>A Warm reset</td>
<td>Called(TakeReset)</td>
</tr>
<tr>
<td>State Change C</td>
<td>Entry to Debug state</td>
<td>Rose(Halted)</td>
</tr>
<tr>
<td>Event D</td>
<td>Preemption by a higher priority processor exception</td>
<td>Called(ExceptionEntry)</td>
</tr>
</tbody>
</table>
Rule JRJC

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority processor exception.
Rule JRJC
Exit from lockup is by any of the following:

• A Cold reset.
• A Warm reset.
• Entry to Debug state.
• Preemption by a higher priority processor exception.

\[
\text{Fell(LockedUp)} \rightarrow \text{Called(TakeColdReset)} \\
\quad \lor \text{Called(TakeReset)} \\
\quad \lor \text{Rose(Halted)} \\
\quad \lor \text{Called(ExceptionEntry)}
\]
Rule JRJC

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority processor exception.

\[
\text{Fell}(\text{LockedUp}) \rightarrow \text{Called}(\text{TakeColdReset}) \\
\quad \lor \text{Called}(\text{TakeReset}) \\
\quad \lor \text{Rose}(\text{Halted}) \\
\quad \lor \text{Called}(\text{ExceptionEntry})
\]

\[
\begin{align*}
\text{__Called}_\text{TakeColdReset} &= \text{FALSE}; \\
\text{__Called}_\text{TakeReset} &= \text{FALSE}; \\
\text{__Called}_\text{TakeExceptionEntry} &= \text{FALSE}; \\
\text{__Past}_\text{LockedUp} &= \text{LockedUp}; \\
\text{__Past}_\text{Halted} &= \text{Halted}; \\
\text{assert}(\text{__Past}_\text{LockedUp} > \text{LockedUp}) &\implies (\text{__Called}_\text{TakeColdReset} \\
&\quad \lor \text{__Called}_\text{TakeReset} \\
&\quad \lor \text{__Past}_\text{Halted} < \text{Halted} \\
&\quad \lor \text{__Called}_\text{ExceptionEntry});
\end{align*}
\]
Rule VGNW

Entry to lockup from an exception causes

• Any Fault Status Registers associated with the exception to be updated.

• No update to the exception state, pending or active.

• The PC to be set to 0xFFFFFFFFE.

• EPSR.IT to become UNKNOWN.

In addition, HFSR.FORCED is not set to 1.
Rule VGNW

Entry to lockup from an exception causes

• Any Fault Status Registers associated with the exception to be updated.

• No update to the exception state, pending or active.

• The PC to be set to 0xFFFFFFFFF.

• EPSR.IT to become UNKNOWN.

In addition, HFSR.FORCED is not set to 1.
Arithmetic operations
Boolean operations
Bit Vectors
Arrays
Functions
Local Variables
Statements
  Assignments
  If-statements
Loops
Exceptions

Arithmetic operations
Boolean operations
Bit Vectors
Arrays
Functions
Local Variables
Statements
  Assignments
  If-statements
Loops
Exceptions
Formally Validating Specifications

- v8-M Spec
- Property

Verification

- CEX
- Bug in Spec

- Proof
Formally Validating Specifications

v8-M Spec → Verification → CEX → Proof

Property

12 Bugs Found so far
Public release of machine readable Arm specification

Enable formal verification of software and tools

Releases
  April 2017: v8.2
  July 2017: v8.3

Working with Cambridge University REMS group to convert to SAIL
  Backends for HOL, OCaml, Memory model, (hopefully Coq too)

Tools: https://github.com/alastairreid/mra_tools

Talk to me about how I can help you use it
Potential uses of processor specifications

Verifying compilers
Verifying OS page table / interrupt / boot code
Verifying processor pipelines
Verification and discovery of peephole optimizations
Automatic generation of binary translators
Automatic generation of test cases
Decompilation of binaries
Abstract interpretation of binaries
etc.
How can you trust formally verified software?
How can you trust formal specifications?

Test the specifications you depend on

Ensure specifications have multiple uses

Create meta-specifications

https://xkcd.com/1416/
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!

“Trustworthy Specifications of the ARM v8-A and v8-M architecture,” FMCAD 2016
“End to End Verification of ARM processors with ISA Formal,” CAV 2016
“Who guards the guards? Formal Validation of ARM v8-M Specifications,” OOPSLA 2017

@alastair_d_reid
arm