OpenSSL – Multiple Vulnerabilities

MSS-SIEM

Prepared By: Managed Service / Revision Number: 1.1

Date: 06/9/2014

Table of Contents

Technical Summary

OpenSSL.org has issued an advisory for multiple vulnerabilities. The most significant of these is the **SSL/TLS MITM vulnerability (CVE-2014-0224)**. An attacker using a carefully crafted handshake can force the use of weak keying material in OpenSSL SSL/TLS clients and servers. This can be exploited by a Man-in-the-middle (MITM) attack where the attacker can decrypt and modify traffic from the attacked client and server.

Google's Adam Langley has written an analysis of the bug. He notes: "... these attacks need man-in-the-middle position against the victim and that ***non-OpenSSL clients (IE, Firefox, Chrome on Desktop and iOS, Safari etc) aren't affected.*** None the less, all OpenSSL users should be updating." He adds (on Twitter) that **Chrome on Android does use OpenSSL**, but he has not confirmed that it is vulnerable.

Mitre.org has assigned the following CVEs:

- SSL/TLS MITM vulnerability (CVE-2014-0224)
- DTLS recursion flaw (CVE-2014-0221)
- SSL_MODE_RELEASE_BUFFERS NULL pointer dereference (CVE-2014-0198)
- DTLS invalid fragment vulnerability (CVE-2014-0195)
- Anonymous ECDH denial of service (CVE-2014-3470)

Impact

Unlike Heartbleed, the most recent OpenSSL vulnerabilities are indirect attack vectors and would require some degree of privileged network access. With Heartbleed an attacker can simply target a vulnerable system and extract data from memory with a high level of consistency. A remote attacker with a man-in-the-middle vantage point on the network may be able to decrypt or modify traffic between a client and server. An attacker may also be able to perform denial of service attacks against users. This attack vector is highly dependent upon network access, the ciphers in use and system configuration.

Affected Versions

OpenSSL provides the SSL implementation in many mainstream products and applications including the following that may be affected by this vulnerability. The vulnerability of the individual product will depend on the linked version of OpenSSL used to build the application or the installed library version.

- Apache web server (see table below to confirm SSL version)
- Tomcat (see table below to confirm SSL version)
- Nginx (see table below to confirm SSL version)
- “Appliances” based on Apache web server and other services using OpenSSL
- “Management Interfaces” based on Apache web server and other services using OpenSSL
- A wide array of Linux-based products and systems

Per OpenSSL, the following are versions of OpenSSL and their status in relation to exploitation:

- All ***CLIENT*** versions of OpenSSL are vulnerable.
- ***SERVER*** versions 1.0.1 and 1.0.2-beta1 are vulnerable.
- Users of ***SERVER*** versions earlier than 1.0.1 are advised to upgrade as a precaution.

Several operating systems were distributed with affected versions, they are:

- Debian
- Fedora
- RedHat
- Ubuntu

- FreeBSD

Additional vulnerable products are listed below.

The vulnerability **DOES NOT** affect the following major platforms:

- Microsoft IIS (all versions)
- Apple IOS

Technical Details

Original OpenSSL.org Advisory:

> OpenSSL Security Advisory [05 Jun 2014]
> ==
>
> SSL/TLS MITM vulnerability (CVE-2014-0224)
> ==
>
> An attacker using a carefully crafted handshake can force the use of weak
> keying material in OpenSSL SSL/TLS clients and servers. This can be exploited
> by a Man-in-the-middle (MITM) attack where the attacker can decrypt and
> modify traffic from the attacked client and server.
>
> The attack can only be performed between a vulnerable client *and*
> server. OpenSSL clients are vulnerable in all versions of OpenSSL. Servers
> are only known to be vulnerable in OpenSSL 1.0.1 and 1.0.2-beta1. Users
> of OpenSSL servers earlier than 1.0.1 are advised to upgrade as a precaution.
>
> OpenSSL 0.9.8 SSL/TLS users (client and/or server) should upgrade to 0.9.8za.
> OpenSSL 1.0.0 SSL/TLS users (client and/or server) should upgrade to 1.0.0m.
> OpenSSL 1.0.1 SSL/TLS users (client and/or server) should upgrade to 1.0.1h.
>
> Thanks to KIKUCHI Masashi (Lepidum Co. Ltd.) for discovering and
> researching this issue. This issue was reported to OpenSSL on 1st May
> 2014 via JPCERT/CC.
>
> The fix was developed by Stephen Henson of the OpenSSL core team partly based
> on an original patch from KIKUCHI Masashi.

DTLS recursion flaw (CVE-2014-0221)
==========

By sending an invalid DTLS handshake to an OpenSSL DTLS client the code
can be made to recurse eventually crashing in a DoS attack.

Only applications using OpenSSL as a DTLS client are affected.

OpenSSL 0.9.8 DTLS users should upgrade to 0.9.8za
OpenSSL 1.0.0 DTLS users should upgrade to 1.0.0m.
OpenSSL 1.0.1 DTLS users should upgrade to 1.0.1h.

Thanks to Imre Rad (Search-Lab Ltd.) for discovering this issue. This
issue was reported to OpenSSL on 9th May 2014.

The fix was developed by Stephen Henson of the OpenSSL core team.

DTLS invalid fragment vulnerability (CVE-2014-0195)
==========

A buffer overrun attack can be triggered by sending invalid DTLS fragments
to an OpenSSL DTLS client or server. This is potentially exploitable to
run arbitrary code on a vulnerable client or server.

Only applications using OpenSSL as a DTLS client or server affected.

OpenSSL 0.9.8 DTLS users should upgrade to 0.9.8za
OpenSSL 1.0.0 DTLS users should upgrade to 1.0.0m.
OpenSSL 1.0.1 DTLS users should upgrade to 1.0.1h.

Thanks to Jüri Aedla for reporting this issue. This issue was
reported to OpenSSL on 23rd April 2014 via HP ZDI.

The fix was developed by Stephen Henson of the OpenSSL core team.

SSL_MODE_RELEASE_BUFFERS NULL pointer dereference (CVE-2014-0198)
==========

A flaw in the do_ssl3_write function can allow remote attackers to
cause a denial of service via a NULL pointer dereference. This flaw
only affects OpenSSL 1.0.0 and 1.0.1 where SSL_MODE_RELEASE_BUFFERS is
enabled, which is not the default and not common.

OpenSSL 1.0.0 users should upgrade to 1.0.0m.
OpenSSL 1.0.1 users should upgrade to 1.0.1h.

This issue was reported in public. The fix was developed by
Matt Caswell of the OpenSSL development team.

SSL_MODE_RELEASE_BUFFERS session injection or denial of service (CVE-2010-5298)
==========

A race condition in the ssl3_read_bytes function can allow remote
attackers to inject data across sessions or cause a denial of service.
This flaw only affects multithreaded applications using OpenSSL 1.0.0
and 1.0.1, where SSL_MODE_RELEASE_BUFFERS is enabled, which is not the
default and not common.

OpenSSL 1.0.0 users should upgrade to 1.0.0m.
OpenSSL 1.0.1 users should upgrade to 1.0.1h.

This issue was reported in public.

Anonymous ECDH denial of service (CVE-2014-3470)
==========

OpenSSL TLS clients enabling anonymous ECDH ciphersuites are subject to a
denial of service attack.

OpenSSL 0.9.8 users should upgrade to 0.9.8za
OpenSSL 1.0.0 users should upgrade to 1.0.0m.
OpenSSL 1.0.1 users should upgrade to 1.0.1h.

Thanks to Felix Gröbert and Ivan Fratrić at Google for discovering this
issue. This issue was reported to OpenSSL on 28th May 2014.

The fix was developed by Stephen Henson of the OpenSSL core team.

Other issues
==========

OpenSSL 1.0.0m and OpenSSL 0.9.8za also contain a fix for
CVE-2014-0076: Fix for the attack described in the paper "Recovering
OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack"
Reported by Yuval Yarom and Naomi Benger. This issue was previously
fixed in OpenSSL 1.0.1g.

References
==========

URL for this Security Advisory:
http://www.openssl.org/news/secadv_20140605.txt

Note: the online version of the advisory may be updated with additional
details over time.

Determining Vulnerability

A number of tools and signatures have been developed to address this vulnerability.

Commercial Vulnerability Scanning Tools

The following scanning vendors have released checks for this vulnerability:

Rapid7 (Nexpose)

Rapid7 has released an additional check within the Nexpose to identify these vulnerabilities. This can also be achieved by analyzing package versions using authenticated scans on some systems.

Tenable (Nessus)

Tenable has released a check for the Nessus product line. This can also be achieved by analyzing package versions using authenticated scans on some systems.

Qualys

Qualys has released a check for their scanner.

Other Methods

Below is a set of steps for the Linux command line to assist in assessing if hosts are vulnerable. These steps apply to most Linux and Apple operating systems.

OpenSSL Command Line

Running the command "openssl version -a" will return the version information. If the version is ***SERVER*** 1.0.1 or 1.0.2-beta1, it IS vulnerable. **ALL *CLIENT*** versions ARE vulnerable.

NMAP

Seclists.org has introduced a custom NSE check for use with NMAP. The output in this case is a bit difficult to parse but NMAP is a great tool for testing multiple sites.

```
# turn off pings, turn off dns resolution pull version banners, run the test script, output in all formats and pull in a
# list of targets

nmap –T3 –n –vvv –PN -sV –open --script=ssl-ccs-injection –web-xml –oA CVE-2014-0224 –iL targets.txt
```

The custom NSE script is available on seclists.org:

```
local bin = require('bin')
local match = require('match')
local nmap = require('nmap')
local shortport = require('shortport')
local sslcert = require('sslcert')
local stdnse = require('stdnse')
local string = require('string')
local table = require('table')
local vulns = require('vulns')
local have_tls, tls = pcall(require,'tls')

assert(have_tls,
  "This script requires tls.lua from http://nmap.org/nsedoc/lib/tls.html")

description = [[
Detects whether a server is vulnerable to the SSL/TLS "CCS Injection"
vulnerability (CVE-2014-0160), first discovered by Masashi Kikuchi.
The script is based on the ccsinjection.c code authored by Ramon de C Valle
(https://gist.github.com/rcvalle/71f4b027d61a78c42607)

In order to exploit the vulnerablity, a MITM attacker would effectively
do the following:

   o Wait for a new TLS connection, followed by the ClientHello
     ServerHello handshake messages.

   o Issue a CCS packet in both the directions, which causes the OpenSSL
     code to use a zero length pre master secret key. The packet is sent
```

```
      to both ends of the connection. Session Keys are derived using a
      zero length pre master secret key, and future session keys also
      share this weakness.

   o Renegotiate the handshake parameters.

   o The attacker is now able to decrypt or even modify the packets
     in transit.

The script works by sending a 'ChangeCipherSpec' message out of order and
checking whether the server returns an 'UNEXPECTED_MESSAGE' alert record
or not. Since a non-patched server would simply accept this message, the
CCS packet is sent twice, in order to force an alert from the server. If
the alert type is different than 'UNEXPECTED_MESSAGE', we can conclude
the server is vulnerable.
]]

---
-- @usage
-- nmap -p 443 --script ssl-ccs-injection <target>
--
-- @output
-- PORT    STATE SERVICE
-- 443/tcp open  https
-- | ssl-ccs-injection:
-- |   VULNERABLE:
-- |   SSL/TLS MITM vulnerability (CCS Injection)
-- |     State: VULNERABLE
-- |     Risk factor: High
-- |     Description:
-- |       OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before
-- |       1.0.1h does not properly restrict processing of ChangeCipherSpec
-- |       messages, which allows man-in-the-middle attackers to trigger use
-- |       of a zero-length master key in certain OpenSSL-to-OpenSSL
-- |       communications, and consequently hijack sessions or obtain
-- |       sensitive information, via a crafted TLS handshake, aka the
-- |       "CCS Injection" vulnerability.
-- |
```

```
-- |     References:
-- |       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224
-- |       http://www.cvedetails.com/cve/2014-0224
-- |_      http://www.openssl.org/news/secadv_20140605.txt

author = "Claudiu Perta <claudiu.perta@gmail.com>"
license = "Same as Nmap--See http://nmap.org/book/man-legal.html"
categories = { "vuln", "safe" }



portrule = function(host, port)
  return shortport.ssl(host, port) or sslcert.isPortSupported(port)
end

--- Receives an alert record from the server
--      Byte  0    = SSL record type (21 - '0x15' in case of an alert)
--      Bytes 1-2  = SSL version (major/minor)
--      Bytes 3-4  = Length of data in bytes, excluding the header itself
--                   (big-endian convention is used)
--      Byte  5    = Alert level (1: Warning, 2: Fatal)
--      Byte  6    = Alert value
local function receive_alert(s)
  local status, data = s:receive_buf(match.numbytes(7), true)
  if not status then
    stdnse.print_debug(1, 'Error: could not read data from socket')
    return nil, nil, nil
  end

  -- Unpack the record
  local pos, r_type, v1, v2, r_len, alert_level, alert_type = bin.unpack(
    'CCC>SCC', data)

  -- Not an alert record
  if r_type ~= 21 then
    stdnse.print_debug(1, 'Error: not an alert record')
    return nil, nil, nil
  end
```

```
  stdnse.print_debug(1, string.format(
    "%x %x %x", r_type, alert_level, alert_type))

  return status, alert_level, alert_type
end

local function keys(t)
  local ret = {}
  for k, _ in pairs(t) do
    ret[#ret+1] = k
  end
  return ret
end

local function test_ccs_injection(host, port, version)
  local hello = tls.client_hello({
      ["protocol"] = version,
      -- Claim to support every cipher
      -- Doesn't work with IIS, but IIS isn't vulnerable
      ["ciphers"] = keys(tls.CIPHERS),
      ["compressors"] = {"NULL"},
      ["extensions"] = {
        -- Claim to support every elliptic curve
        ["elliptic_curves"] = tls.EXTENSION_HELPERS["elliptic_curves"](
          keys(tls.ELLIPTIC_CURVES)),
        -- Claim to support every EC point format
        ["ec_point_formats"] = tls.EXTENSION_HELPERS["ec_point_formats"](
          keys(tls.EC_POINT_FORMATS)),
      },
    })

  local s
  local specialized = sslcert.getPrepareTLSWithoutReconnect(port)
  if specialized then
    local status
    status, s = specialized(host, port)
    if not status then
      stdnse.print_debug(1, "Connection to server failed")
```

```
    return
  end
else
  s = nmap.new_socket()
  local status = s:connect(host, port)
  if not status then
    stdnse.print_debug(1, "Connection to server failed")
    return
  end
end

s:set_timeout(5000)

-- Send Client Hello to the target server
local status, err = s:send(hello)
if not status then
  stdnse.print_debug(1, "Couldn't send Client Hello: %s", err)
  s:close()
  return false
end

-- Read response
local done = false
local i = 1
local response
repeat
  status, response, err = tls.record_buffer(s, response, i)
  if err == "TIMEOUT" then
    -- Timed out while waiting for server_hello_done
    -- Could be client certificate required or other message required
    -- Let's just drop out and try sending the heartbeat anyway.
    done = true
    break
  elseif not status then
    stdnse.print_debug(1, "Couldn't receive: %s", err)
    s:close()
    return false
  end
```

```
  local record
  i, record = tls.record_read(response, i)
  if record == nil then
    stdnse.print_debug("%s: Unknown response from server", SCRIPT_NAME)
    s:close()
    return false
  elseif record.protocol ~= version then
    stdnse.print_debug("%s: Protocol version mismatch", SCRIPT_NAME)
    s:close()
    return false
  end

  if record.type == "handshake" then
    for _, body in ipairs(record.body) do
      if body.type == "server_hello_done" then
        stdnse.print_debug(1, "Handshake completed!")
        done = true
      end
    end
  end
until done

-- Send the change_cipher_spec message twice to
-- force an alert in the case the server is not
-- patched.

-- change_cipher_spec message
local ccs = tls.record_write(
  "change_cipher_spec", version, bin.pack("C", 0x01))

-- Send the first ccs message
status, err = s:send(ccs)
if not status then
  stdnse.print_debug(1, "Couldn't send first ccs message: %s", err)
  s:close()
  return false
end
```

```
  -- Send the second ccs message
  status, err = s:send(ccs)
  if not status then
    stdnse.print_debug(1, "Couldn't send second ccs message: %s", err)
    s:close()
    return false
  end

  -- Read the alert message. We expect alert level = 2 (fatal)
  -- and alert type = 10 (UNEXPECTED MESSAGE)
  local alert_level, alert_type
  status, alert_level, alert_type = receive_alert(s)

  if status and alert_level == 2 and alert_type == 10 then
    stdnse.print_debug(
      1, 'Server returned UNEXPECTED_MESSAGE, not vulnerable')
    s:close()
    return false
  else
    stdnse.print_debug(
      1, 'Vulnerable - response is not UNEXPECTED_MESSAGE')
    s:close()
    return true
  end
end

action = function(host, port)
  local vuln_table = {
    title = "SSL/TLS MITM vulnerability (CCS Injection)",
    state = vulns.STATE.NOT_VULN,
    risk_factor = "High",
    description = [[
OpenSSL before 0.9.8za, 1.0.0 before 1.0.0m, and 1.0.1 before 1.0.1h
does not properly restrict processing of ChangeCipherSpec messages,
which allows man-in-the-middle attackers to trigger use of a zero
length master key in certain OpenSSL-to-OpenSSL communications, and
consequently hijack sessions or obtain sensitive information, via
```

```
a crafted TLS handshake, aka the "CCS Injection" vulnerability.
    ]],
    references = {
      'https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224',
      'http://www.cvedetails.com/cve/2014-0224',
      'http://www.openssl.org/news/secadv_20140605.txt'
    }
  }

  local report = vulns.Report:new(SCRIPT_NAME, host, port)
  local tls_version = 'TLSv1.0'
  local status = test_ccs_injection(host, port, tls_version)
  if (status) then
    vuln_table.state = vulns.STATE.VULN
  end

  return report:make_output(vuln_table)
end
```

Recommendations

Overview

In situations where the build of OpenSSL and other related applications such as Apache web server are controlled by the customer, it is possible to update or rebuild the applications to use a more current version of OpenSSL. Since many appliance-based solutions and third party packages are built on this platform.

Patching

OpenSSL has provided a patched version of the software that will prevent exploitation. The patched versions are OpenSSL 0.9.8za, 1.0.0m and 1.0.1h. It is available for download through OpenSSL and available through OS vendor patching portals as well.

Third-Party Vendors

Follow up with third-party vendors and service providers for official recommendations. Many third-party products and appliances implement OpenSSL that require updates. As a result many of the workarounds may not be possible without support from the vendor.

Workaround (Manual Builds of OpenSSL)

No known workarounds.

Known Vulnerable Vendors

Vendor Information (Learn More)

Vendor	Status	Date Notified	Date Updated
Rapid7	Multiple products affected.	6/5/2014	6/6/2014
Aruba	Multiple products affected.	6/5/2014	6/5/2014
Cisco	Multiple products affected.	6/5/2014	6/6/2014
Checkpoint	Multiple products affected.	6/6/2014	6/6/2014
Debian	Multiple versions affected.	6/5/2014	6/5/2014
Fedora	Multiple versions affected.	6/5/2014	6/5/2014
RedHat	Multiple versions affected.	6/5/2014	6/5/2014
Ubuntu	Multiple versions affected.	6/5/2014	6/5/2014
FreeBSD	Multiple versions affected.	6/5/2014	6/5/2014

Monitoring/Detection

The following vendors have released signatures for detecting the attack at the time of writing:

Palo Alto Networks

To address this vulnerability, Palo Alto Networks has released an emergency content update that provides prevention of the exploitation of the SSL/TLS MITM with IPS vulnerability signature ID 36476 ("OpenSSL SSL/TLS MITM vulnerability") with critical severity and default action of alert. Palo Alto Networks customers with a Threat Prevention subscription are advised to verify that they are running the latest content version on their devices.

Sourcefire

To address some possible attacks related to the reported vulnerabilities, Sourcefire has released some content to help identify and prevent activity related to the reported vulnerabilities. Sourcefire has released GID 31180 and 31181.

Accuvant MSS Recommendations

Accuvant Managed Security Solutions is currently developing custom content to identify any related activity across all of the managed platforms and service lines. Accuvant clients with any of the devices listed in the Monitoring/Detection section of this document with updated signatures will have coverage for this issue.

Strategic Recommendations

To ensure thorough mitigation Accuvant strongly recommends the following additional steps:

Use one of the vulnerability detection options listed above to map and inventory all instances of potentially vulnerable OpenSSL deployments. At a minimum, all critical systems with a client IT environment should be assessed. This includes, but is not limited to the following device or system types:

- Internet Facing OpenSSL based Web Servers
- Firewall and security device HTTPS Management Interfaces
- OpenSSL based VPN Services
- Wireless Controller Technologies
- Linux based “Appliances”

Update asset intelligence databases within client SIEM environments to include all known instances of any discovered vulnerabilities. This is will assist in effectively prioritizing response efforts in the event of a wide spread attack involving multiple targets within a client IT environment.

Assure that OpenSSL and other mission critical third-party applications are included in routine patch management efforts.

References

1. https://www.openssl.org/news/secadv_20140605.txt
2. http://www.zdnet.com/openssl-fixes-another-severe-vulnerability-7000030253/
3. https://www.imperialviolet.org/2014/06/05/earlyccs.html
4. http://ccsinjection.lepidum.co.jp/blog/2014-06-05/CCS-Injection-en/index.html
5. https://access.redhat.com/site/articles/904433
6. https://isc.sans.edu/diary/Critical+OpenSSL+Patch+Available.+Patch+Now!/18211
7. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0224
8. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0221
9. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0198
10. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0195
11. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3470
12. http://cve.mitre.org/index.html

Revisions

Release Version:	1.1 – Initial Release
Date:	6/9/2014
Summary of Changes:	Initial release