It rather involved being on the other side of this airtight hatchway: Replacing an unsigned writable MSI package

devblogs.microsoft.com/oldnewthing/20180227-00

February 27, 2018

Raymond Chen

In the category of dubious security vulnerability, we have this report:

> I have found a security vulnerability in Windows that permits an attacker to run arbitrary code on a remote system. Prepare the system as follows:
>
> - Set up a domain controller and one or more clients joined to that domain.
> - Create an MSI package and put it on a network share. For example, you can use the popular XYZ package, available for download here.
> - Use Group Policy to deploy the MSI package to the client machines.
>
> Given these starting conditions, the attacker can do the following:
>
> - Replace the MSI package on the network share with a rogue package that contains the malicious payload.
> - Wait for the user to Repair the MSI package. On a large corporate network, this is probably not going to take long.
> - When the user performs a Repair, Windows will automatically download and install the rogue MSI package from the network share. Since MSI packages can install registry keys, shell extensions, and services, the malicious payload will start executing almost immediately.[1]

Okay, let's see what we have here.

First, we have the XYZ package. Closer inspection of the XYZ package reveals that it is not digitally signed. Therefore, anybody can tamper with it undetected, since there is nothing that attests to its authenticity.

Okay, but a rogue copy of the XYZ package is dangerous only if you manage to trick somebody into installing it. The first step in the attack is to convince the domain administrator to deploy the authentic XYZ package to all the machines on the network.

The second step in the attack is that the network administrator must have placed the authentic XYZ package in a location that the attacker has write access to.

Okay, let's stop and take another look at those prerequisites.

We're requiring the domain administrator to deploy an MSI package that is not digitally signed, and put it on an insecure share.

In other words, we're assuming that the domain administrator is incompetent and set up an insecure system. Insecure system is insecure.

If you study this report more carefully, you'll see that you don't even need to wait for the Repair step in order to attack client machines. You just have to wait for a new client computer to join the domain. On a large corporate network, this will not take long. That new client computer will receive instructions from the Group Policy object it received from the domain controller that it should install the XYZ program from the insecure share. You don't even need any of the steps in the second half of the vulnerability report: The new client computer is already attacked the moment it joins the domain!

The finder tried to salvage this report by altering the attack slightly:

> Alternatively, the attacker can replace the `XYZ.DLL` file in its default location of `C:\Program Files\XYZ\XYZ.DLL` with a rogue copy.

Okay, but wait: How did the attacker get write access to `C:\Program Files\XYZ\XYZ.DLL`? By default, you need administrator privileges to write to that file. If an attacker can write to that file, then it means one of three things:

- The attacker already has administrator privileges.
- The security on that file is different from the default.
- The attacker does not have administrator privileges, the security on the file is set to the default, but the attacker found a sneaky way to trick the system into allowing a non-administrator to replace the file.

In the first case, the attacker is already on the other side of the airtight hatchway. Nothing exciting there.

In the second case, the XYZ package created an insecure configuration that allows non-administrators to update one of its files. That also falls into the category of "If you set up an insecure system, then don't be surprised that it's insecure."

The third case is interesting. But the finder provided no details as to what sort of sneaky trick is being used. The finder merely presupposed the existence of a vulnerability, and concluded "If there is a vulnerability, then I found a vulnerability."

Or in the parlance of airtight hatchways: “If there is a passageway that lets me to get to the other side of the airtight hatchway, then I can get to the other side of the airtight hatchway.”

As you might suspect, this is not a particularly interesting statement.

[1] In reality, the finder’s attack was more convoluted than this. In their version, the rogue XYZ package contained a slightly altered version of `XYZ.DLL`, and you had to go through some extra contortions to get the system to start using the rogue version of that file without the user launching the XYZ program. I removed all the style points and went to the direct attack.

Raymond Chen

Follow