Owning the datacenter using Cisco NX-OS

By George Hedfors at Cybercom dot com

Abstract

Cisco's 4000, 5000 and 7000-series switches have a Linux based operating system. The first variation was called SAN-OS and is implemented on the 4000 and 5000 series. Thereafter it was changed into NX-OS (Nexus) for the 7000-series, which also is called "Next Generation Switches".

The slightly modified Linux that serves as backend of the switch is changed to look like an out-of-the-box Cisco hardware at the first glance. However, it also has all kinds of vulnerabilities that can be associated with any Linux distribution.

Introduction

The "Nexus" is usually used in large high performance networks where you need close to 100% uptime. It can also be considered a cloud solution due to the fact that it allows for creation of virtual switch devices called VDC (Virtual Device Context). Each VDC serves as a layer 1 separated switch device, which may interact, with other VDC's in the same Nexus. This is ideal for high availability solutions as the Nexus also has dual hardware built into the same chassis with seamless fail over. Typically the Nexus would serve as the central switching device in a large datacenter allowing virtual networks and virtual switches to be created in order to design the network.

However, owning the Nexus would give you the whole datacenter on a silver plate…

Brief History

The research began as George Hedfors was assessing the security of a newly designed high availability network controlled by the Nexus. During the assessment, a core vulnerability was discovered as the whole Nexus suddenly failed over during a routine layer 2 attack. The attack consisted of FX's old CDP (Cisco Discovery Protocol) flooding and denial of service attack, first demonstrated in the year 2001. Thereafter George continued to research potential vulnerabilities, as the Nexus also is available for VMware as a virtual switch virtualization..

Vulnerabilities

As the Nexus no longer consists of the classic Cisco IOS kernel-only operating system, they've redesigned everything. Protocols and such are now handled by the Linux operating system as operating system daemons that "obviously" need root-level system access.

CDP heap overflow

The CDP daemon is designed to receive packets on layer 2 in order to register other Cisco devices in the same network. Devices transmit a number of details using CDP, including the device name. The name may be up 255 characters long according to the protocol specification.

The vulnerability consists of a classic integer truncation error where the device name length is specified by an integer (32-bit), which is thereafter converted into a byte (8-bit). This means that if the size is larger than 8-bits, it will be too long. If then one specifies a size larger then 8-bits, it will consequently be truncated into an 8-bit number and you have your heap overflow once the original content size is read into the much too small allocated space.

GDB server CLi escape

There is a "hidden" command, which one can use to trigger the execution of a GDB server. This is a function generally used by support technicians from Cisco to debug problems that may occur. Once knowing about the hidden gdb command, one can list the current processes ("show processes") and choose any process ID to debug using GDB. It's then possible to connect a remote computer to the GDB server in order to debug the running process.

As everyone who has some knowledge of GDB knows, it's possible to "call" any function in the process that is being debugged. This means that one could call on "/bin/id" for example or any other binary of choosing. Doing that and some other untold details, it's possible to escape from the Cisco CLi into the backend Linux gaining the privilege of the process that executed the shell.

VSH parser CLi escape

VSH, not sure what it really stands for, is however the Cisco CLi that faces users once connecting using SSH or …telnet. Anyway, some 20(?) odd years ago, people discovered that redirecting pipes might be a security problem. Thereafter, people in general has considered this possibility and programmed with proper input validation.

Never the less, VSH still seem to be unaware of those issues, hence in some cases it's possible to invoke external commands. Commands will be executed in the Linux backend using pipe redirections. It's therefore possible to escape shell by simply executing bash.

```
nexus# ssh `/bin/bash -i`
```

VSH privilege escalation

Once upon a time when the Nexus was created, the architecture design didn't expect anyone to break out of the VSH CLi. As previously discussed, this is entirely possible. Therefore, the role separation functionality was simply programmed as a feature of the VSH shell. These features are invoked once an admin for example chooses to access a different VDC than the current. VSH will then launch with a parameter specifying the VDC to access.

However, given the fact that it's possible to break out of the CLi, it may also be restarted using any parameters of the users choice. Giving the user the choice of choosing any VDC to access there is also another parameter worth investigating. It's described as "Disable all roles" using the VSH help…