Investigating the Security of Nexus 1000V Virtual Switches in VMware ESXi Hypervisors

Raymond A. Hansen
Purdue University
401 N Grant St
West Lafayette, IN 47906
hansenr@purdue.edu

Benjamin Peterson
Purdue University
401 N Grant St
West Lafayette, IN 47906

Timothy Becker
Purdue University
401 N Grant St
West Lafayette, IN 47906
becker43@purdue.edu

ABSTRACT
In this paper, the security posture of two versions of the Cisco Nexus 1000V virtual switch is tested against a set of exploits known to be valid on physical switching infrastructure. Specifically, the Nexus 1000V as implemented with VMware's ESXi hypervisor is examined. The attempted exploits are CAM table overflows, VLAN hopping, Spanning Tree manipulation, ARP poisoning, and Private VLAN attacks. With the exception of Spanning Tree manipulation, the Nexus 1000V is vulnerable to all of the attacks in at least one of the tested release combinations. This leads to a call for additional security considerations when deploying the Nexus 1000V/ESXi combination in data centers and cloud provider networks as intended by their design.

Keywords
Network Security; Virtualization; Network Function Virtualization; Layer 2 Security;

1. INTRODUCTION
Virtualization and cloud technologies are now pervasive in enterprise networks [2] due the many advantages offered. Implementation within enterprises allows for the reduction of underutilized server infrastructure; thereby enabling efficiencies in management and delivery of network applications and services to users. In cloud provider environments, this virtualization is further leveraged for rapid service provisioning, improved disaster recovery and business continuity, and service isolation [13]. As server virtualization has increased to nearly 75% of all x86 implementations [2], a need to effectively, efficiently, and securely manage the network infrastructure that interconnects the virtual machines, host systems, and hypervisors has also emerged.

Network function virtualization has been one of the primary mechanisms by which this has been addressed. That is, implementing a manageable virtual network device with every hypervisor has added a layer of control and management capabilities that are difficult to achieve with dedicated physical switches, firewalls, and routers. VMware has termed their implementation of the virtual network device vSwitch, which provides a virtual switching fabric between virtual NICs, which are connected to virtual machines. However, when virtualization clusters can host hundreds, or thousands, of virtual machines, a simple virtual switch may not accommodate the needs required by such complex demands**.** As such, VMware has enhanced these capabilities further with the vNetwork Distributed Switch, seeking to leverage the ability to manage multiple vSwitch instances from one management console [3,10].

While this is desirable from a service system administration standpoint, the distributed approach remains proprietary and generally incompatible with the physical network infrastructure that is present to physically interconnect the host hardware. Additionally, as this approach is further implemented, the traditional division of responsibilities between network/security administrators and system administrators is blurred. Ultimately, this could lead to a breakdown in the security posture an organization has due to a lack of a single entity that is responsible for the requisite tasks associated with that security posture.

Cisco Systems, while attempting to remediate this potential breakdown and seeking to continue its dominance in the enterprise network, created a virtual switch that is a direct replacement for VMware's vSwitch in ESX. This virtual network device, the Nexus 1000V, is a virtualization of their Nexus fabric extender switches as part of the Nexus Series of data center and cloud provider infrastructure and seamlessly integrates into the ESX/ESXi hypervisor [8,9]. The Nexus 1000V extends the features and functionality available in vNetwork switch while providing an interface identical to the physical Nexus switching hardware.

While there are many known vulnerabilities and attacks against physical infrastructure and switches, it is not currently established if those same vulnerabilities are present in the virtualized switches [2]. Additionally, very little is known concerning if those same attacks that are successful in the physical implementations are portable to the virtualized implementation [17]. Given the massive numbers of data centers and cloud providers implementing virtualization, and VMware holding a significant portion of the market share [2], and Cisco having more than 12,000 implementations of the Nexus 1000V in ESX environments [9], an investigation of the security of this virtualized architecture is necessary. Due to the challenges of securing both data centers and cloud environments, this knowledge could be immediately critical. That is, as multitenancy grows in the data center and cloud service provider networks, the ability to compromise the network and expose data and information from another customer is a dire possibility. Identifying these potential vulnerabilities and understanding their attack profile is immediately necessary.

The remainder of the paper is organized as follows: section two identifies the details of the network architecture; section three indicates the specific attacks and methodologies that were executed in this work. Section four details the results of each attack on the different versions of the Nexus 1000V, and section

RIIT'16, September 28-October 01 2016, Boston, MA, USA
 ISBN 978-1-4503-4453-1/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2978178.2978188

five provides the conclusions of the work and suggestions for future efforts.

2. NETWORK ARCHITECTURE

Each of these attacks has been proven to be exploitable in physical switching environments. As such to create a realistic test platform, these were tested against the virtual switching environment shown in Figure 1 below.

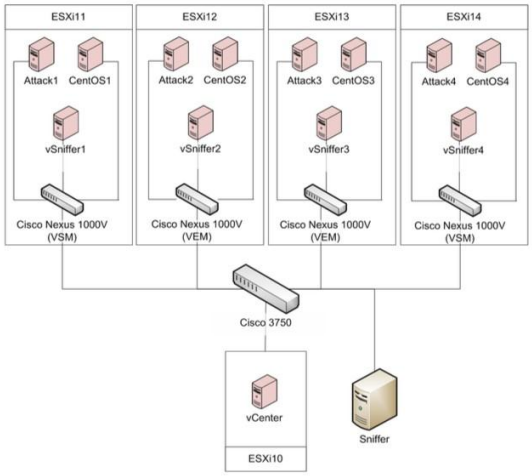

Figure 1 - Network Architecture for Attacks

Each instance of ESXi was installed with vSphere 6.0.0 and were placed in a cluster managed by vCenter version 6.0.0 on the ESXi system. Multiple virtual machines were installed on each system. Minimally, a Kali Linux system was installed to act as an attacker, a CentOS Linux sysem was installed to provide a target, and an Ubuntu Linux client was installed to act as a second target system and to run Wireshark as a monitoring/sniffing client. The physical systems were interconnected with a Cisco 3750G, with the switch uplink connected router, which performed port address translation (PAT). Each ESXi instance had the Cisco Nexus 1000V installed to replace the VMWare vSwitch according to [7]. In the first iteration of the testing, the Nexus 10000V release *4.2(1)sv1(4)* was used. The second iteration used release *5.2(1)sv3(1.15)* in order to evaluate multiple releases of the virtual switch.

Cisco recommends configuration of three VLANs in the Nexus 1000v switching platform: Control, Packet, and Management [5,6]. Additionally, five host VLANs were created for testing purposes within this study.

Table 1 - VLAN & Network Assignment

VLAN ID	Name	IP Address
10	Management	10.96.1.0/24
20	Control	172.28.20.0/24
30	Packet	172.28.30.0/24
110	Private	172.28.110.0/24
120	Public	172.28.120.0/24
153	pvPrimary	172.28.153.0/24
154	pvCommunity	N/A
155	pvIsolated	N/A

Nearly all of the exploits were attempted over VLANs 110 and 120. An ACL was configured that denied any traffic that was sourced from the "Public" VLAN and destined for the "Private" VLAN. Also, VLANs 153, 154, and 155 are members of Private VLANs to limit host-to-host communications. Further, VLAN 155 is set as an isolated network, which only allows connections out to the Internet, with no connections to any other network segments.

3. NETWORK ATTACKS USED

A general categorization of attacks against known and exploited vulnerabilities in switches can be described as such [1]:

CAM Overflow
VLAN Hopping
Spanning Tree Protocol Manipulation
ARP Poisoning
Private VLAN Attack

Specific tools, techniques, and processes were needed to perform each of the attacks. Complex exploits that leveraged the success of one attack to launch a subsequent attack were not attempted. The tools used for each exploit type are shown in Table 2 below. While other tools exist that are able to perform these exploits, these tools were selected based on their ease of use, popularity, documentation, and efficacy.

Table 2 - Vulnerability Exploit Tools

Attack Methodology	Tool(s)
CAM Overflow	macof [18]
VLAN Hopping	Yersinia [15]
STP Manipulation	Yersinia
ARP Poisoning	Ettercap [16]
Private VLAN Attack	Nemesis [14]

The following sections describe each of the individual attacks and the methodologies used to attempt to exploit the potential vulnerabilities.

3.1 CAM Overflow Attack

A CAM table overflow attack takes advantage of the reaction of a switch when its forwarding table reaches its maximum capacity. A CAM table, also called the MAC address table, is a dynamic table located in memory that maps hosts' MAC addresses to physical interfaces on a switch. This table is populated as part of a reactionary process that records MAC addresses as they are received from ingress frames on a specific port. Because the memory assigned to the table is finite, it has a maximum capacity. When the table is full, the switch refuses to add any additional entries into the table until other mappings are removed. Just as traditional physical switches, the default aging time for the Nexus 1000v is 300 seconds.

If a switch receives a frame with a destination MAC address that is not in the forwarding table, then its standard response is to forward the frame out all ports in that broadcast domain. Attackers can manipulate this response to their advantage by intentionally filling the forwarding table to its maximum capacity with counterfeit MAC addresses. Then, when the switch receives a frame with a destination address that it is unaware of, it will forward that packet out all interfaces in the broadcast domain, including the attackers interface. The attacker can then eavesdrop on this traffic and learn potentially sensitive data.

To attempt to exploit this vulnerability, the attack was launched from the Kali Linux VM. The CentOS Linux VM sent ICMP traffic to another host in the same network. A successful exploit would show the ICMP packets being sent through the switch as

being sourced from the CentOS machine, but broadcast out all interfaces on the switch since the CAM table would overflow; resulting the MAC addresses of the VMs being overwritten.

Both of the CentOS Linux VMs were assigned new MAC addresses so that it was certain that the Nexus 1000V had never dynamically learned of those specific MAC addresses. After this change, the *macof* CAM table overflow attack was started by executing the command `macof -i eth0` on the Kali Linux attacker [18]. The *macof* tool instantly started generating random MAC addresses and sending thousands of counterfeit frames with those bogus MAC addresses.

3.2 VLAN Hopping Attack

One method to execute a VLAN Hopping attack is to leverage the ability to place two 802.1Q tags into an Ethernet frame. This exploit sends a specifically crafted Ethernet frame with two VLAN tags into the network to attempt to bypass existing filters, ACLs, etc. The attacker can pass this special frame into an edge interface of a switch, and the switch will then remove the outer-most tag and pass along the rest of the frame, leaving the second tag still in place. [4] This frame must traverse to another switch for this exploit to be successful. This is because when the frame leaves the switch, it should be sent onto the trunk connection with the second VLAN tag still intact. When the second switch receives this frame, it should forward the frame into the segment indicated by the remaining VLAN tag. This method would bypass the layer 3 filter, and "hop" straight onto the other VLAN without any interaction with a layer 3 device. This traffic can only be unidirectional, as the victim will not craft this type of frame for return traffic. However, if there are routes present between the two network segments, traffic may be returned via traditional routing.

The three VMs of ESXi14 were used for this attack. The Kali Linux attacker was configured to be on the Public network, which had an ACL preventing any traffic from the Public network to the Private network from being exchanged. The CentOS target was configured to be a part of the Private network, where it cannot be reached by a device on the Public network.

Yersinia was used to craft double-tagged 802.1Q frames carrying ICMP packets destined for the Private network. Yersinia was operated in interactive mode for this test, which is started by executing the command `yersinia -I` [15]. Figure 2 shows the output of the interactive Yersinia frame construction.

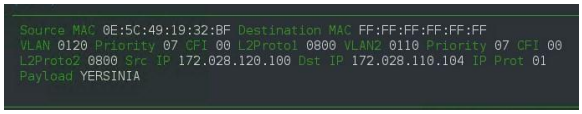

Figure 2 - IEEE 802.1Q Fields Defined via Yersinia

The crafted frames were sent from the attacker. As seen below, the packets sent out from Kali Linux appeared to be properly configured for the attack. These double tagged frames were then verified on ingress to the Nexus 1000V, egress of the Nexus 1000V, and ingress of the Catalyst 3750G

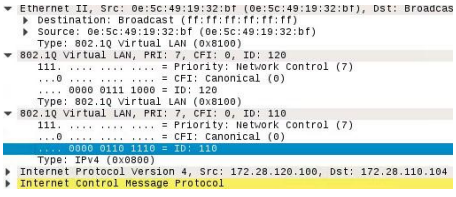

Figure 3 – Double-Tagged Frame from Attacker

3.3 Spanning Tree Protocol Manipulation

The spanning tree protocols utilize specific frames between switches in a network to eliminate switching loops. These specific bridge protocol data units (BPDU) are used to indicate the authoritative switch in the network, the best path to that switch, and which interface(s) should be blocked to keep a loop from forming. The dangers of switching loops in Ethernet environments results in frames being forwarded along inefficient paths, repeatedly forwarded between multiple switches, or generating broadcast storms.

An attacker may craft a specific BPDU that forces the reconvergence of the switches to a new topology. This would potentially allow the attacker to force traffic to flow through a different switch and allow that traffic to be monitored, copied, replayed, etc. Additionally, a crafted BPDU or set of BPDUs could be used to create a denial of service within the network.

Although the Nexus 1000V implemented a loop prevention algorithm that was meant to eliminate the need for STP, there was potential that it would still process STP BPDUs. To attempt to exploit this vulnerability, spoofed STP messages were sent from the attack VM to the Nexus 1000V. Yersinia was used, as it allowed for the creation of falsified STP-compliant frames (Omella, n.d.).

For this exploit, the attack VM was assigned to VLAN 30 and two STP manipulations were attempted. The first manipulation was a Configuration BPDU denial-of-service and the second was a Topology Change Notification BPDU denial-of-service. Both of these attempts were accomplished by again using Yersinia's interactive mode. This time however, the "STP" attack group was selected. The configuration for these packets was based on the information detailed by [19] as shown in Figure 4 below.

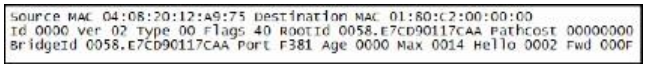

Figure 4 - Yersinia output for STP-compliant Frames

3.4 ARP Poisoning Attack

An ARP poisoning attack leverages the protocol that translates addresses between layer 2 and layer 3, called ARP. The ARP poisoning attack begins by identifying clients inside the network and recording all of the selected clients MAC addresses (layer 2) and IP Addresses (layer 3). ARP is designed in such a way that it will overwrite any older entries in the ARP table that are conflicting. So, the attacker will craft and send unsolicited ARP messages containing the counterfeit IP address associated with the attackers MAC address. After running the exploit, all traffic destined for the victims IP Address will actually be sent to the MAC address of the attacker, which then could potentially be relayed appropriately to the victim so that they are unaware that anything as changed.

To execute the ARP poisoning attack against the Nexus 1000V the three hosts on ESXi14 were placed on the Public network (VLAN 120). For this exploit, it was not necessary to have any hosts on a separate network, as this attack is contained within a layer 2 broadcast domain. The attacker must be able to identify hosts of interest. So, the client VMs were set to indefinitely ping each other. The attacker captured packets to and from its interface connected to the network, but was filtered down to only display this ICMP traffic.

In the figure below, you can see that prior to the Ettercap attack, the ARP table of the Catalyst 3750 switch appears normal, with unique MAC addresses resolved to each IP address in the table. At

this point, the attacker is not capable of seeing any ICMP traffic between the client VMs because the switch is forwarding it to their intended destinations (MAC addresses) and no other interface.

```
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.28.120.104          5  0050.5600.0212  ARPA   Vlan120
Internet  172.28.120.105          0  0050.5612.0102  ARPA   Vlan120
Internet  172.28.120.96          11  0050.56a1.4530  ARPA   Vlan120
Internet  172.28.120.103          7  0050.56a1.03ec  ARPA   Vlan120
Internet  172.28.120.1            -  000f.8fc6.abc4  ARPA   Vlan120
```

Figure 5 - Initial ARP Table for Poisoning Attack

To execute the ARP poisoning exploit, Ettercap was used. The command used to execute the program on the directly connected LAN Segment was: `ettercap -T -M arp ////` [12].

3.5 Private VLAN Attacks

Private VLANs are designed to allow multiple users to access one VLAN, but prevent them from being able to communicate with each other, unless it is to a designated destination (such as a default gateway). This protection mechanism can often be avoided by sending a packet with the destination MAC address of the default gateway (which is allowed in the private network) and a destination IP Address of another host inside the network. The packet will be forwarded across the layer 2 network to the default gateway, and the default gateway will route the packet back onto the same network to the destination IP address, sidestepping the intended security mechanism in place. This exploit typically only functions in a unidirectional method, as the target will not craft this specific packet to forward traffic back to the attacker. Despite the traffic only being unidirectional, the attack can still be leveraged for a denial of service attack.

To attempt to exploit private VLANs on the Nexus 1000V, a private VLAN (VLAN 155) was configured for Isolation mode, which restricts all hosts from communicating with each other. One client VM and the attacker VM were migrated to this isolated private VLAN on ESXi14. The hosts involved were configured with the following IP Addresses and MAC addresses:

Table 3 -

Host	IP Address	MAC Address
Gateway	172.28.153.1	00:0f:8f:c6:ab:c8
Attacker	172.28.153.17	00:50:56:00:02:12
Target	172.28.153.18	00:50:56:a1:03:ec

To verify that the private VLAN was functioning as expected, each host, the attacker and target attempted to ping each other and the gateway. If the private VLAN is functioning properly, both the target and attacker should be able to ping their gateways successfully. However, pings to each other should be unsuccessful as shown in Figure 6 below.

The target VM was capturing packets with Wireshark on its interface connected to the isolated VLAN. While verifying the private VLAN configuration, no standard ICMP packets destined for the target from the attacker were captured with Wireshark, which further validates the configuration of the private VLAN. Nemesis was used to craft the packets needed to exploit this vulnerability [11]. On the attacker machine, the command `sudo nemesis icmp -d eth0 -S 172.28.153.18 -D 172.28.153.17 -M 00:0f:8f:c6:ab:c8` was executed to send the manipulated ICMP packet. The command sends an ICMP packet with a source IP address of 172.28.253.18, a destination IP address of 172.28.153.17 and a destination MAC address of 00:0f:8f:c6:ab:c8.

```
tim@tim-ubuntu-vm:~$ ping 172.28.153.1
PING 172.28.153.1 (172.28.153.1) 56(84) bytes of data.
64 bytes from 172.28.153.1: icmp_seq=1 ttl=255 time=2.77 ms
64 bytes from 172.28.153.1: icmp_seq=2 ttl=255 time=1.18 ms
64 bytes from 172.28.153.1: icmp_seq=3 ttl=255 time=1.20 ms
64 bytes from 172.28.153.1: icmp_seq=4 ttl=255 time=2.71 ms
64 bytes from 172.28.153.1: icmp_seq=5 ttl=255 time=1.99 ms
^C
--- 172.28.153.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.188/1.975/2.775/0.696 ms
tim@tim-ubuntu-vm:~$ ping 172.28.153.18
PING 172.28.153.18 (172.28.153.18) 56(84) bytes of data.
From 172.28.153.17 icmp_seq=1 Destination Host Unreachable
From 172.28.153.17 icmp_seq=2 Destination Host Unreachable
From 172.28.153.17 icmp_seq=3 Destination Host Unreachable
From 172.28.153.17 icmp_seq=4 Destination Host Unreachable
From 172.28.153.17 icmp_seq=5 Destination Host Unreachable
From 172.28.153.17 icmp_seq=6 Destination Host Unreachable
From 172.28.153.17 icmp_seq=7 Destination Host Unreachable
From 172.28.153.17 icmp_seq=8 Destination Host Unreachable
From 172.28.153.17 icmp_seq=9 Destination Host Unreachable
^C
--- 172.28.153.18 ping statistics ---
10 packets transmitted, 0 received, +9 errors, 100% packet loss, time 9047ms
pipe 3
tim@tim-ubuntu-vm:~$
```

Figure 6 – Target Connectivity

4. RESULTS

Table 4 shows a simple summary of the different exploits against the two different architectures. At a rudimentary level, the same vulnerabilities that are present in physical switches are also present in the Cisco Nexus 1000V virtual switch. A majority of these span different releases of the Nexus 1000V as well.

Table 4 - Summary of Results

	CAM Manip	VLAN Hopping	STP	ARP Poison	Private VLAN
ESXi 5.0 with Nexus 1000V release 4.2(1)	Y	Y	N	Y	Y
ESXi 6.0 with Nexus 1000V release 5.2(1)	N	Y	-	Y	Y

In the earlier release of the Nexus 1000V, the CAM overflow attack was successful. That is, the valid MAC addresses were purged from the CAM table and replaced with randomly created MAC addresses that were assigned to valid interfaces on the virtual switch. The two hosts began losing 100% of the frames sent. However, with release 5.2(1), the ICMP packets sent by the victim were successfully sent and received back while the attack was continuously running

Executing the `show mac address-table count vlan 120` command, the Nexus 1000V reported a total number of 20,480 MAC addresses, and never displayed a value greater than that number. Despite Cisco documentation stating that the maximum number of MAC addresses per VLAN within a VEM is 1024 (Cisco Systems, Inc., n.d.), clearly more were present.

The CAM table remained full as the attack continued, but the Wireshark sniffer never received any *broadcast* ICMP packets from the sending machine. In theory, the Nexus 1000v would not have added the new MAC addresses to the forwarding table, and would have been unaware of where to send the ICMP packets. The normal reaction for a switch in this scenario would be to forward the frame out ALL interfaces in the broadcast domain. However this behavior was not observed.

In an effort to understand why the attack was ineffective on the newer release, several factors were analyzed concerning the behavior of the mac address table and an attempt to quantify the

rate at which the attacker could fill the CAM table. Based on these efforts, the Nexus 1000V CAM table could be filled to 20,480 unique addresses in just over 360ms.

To determine if the forwarding table was static or "rolling", I took a sample of the mac-address table after the macof attack had been run against it and was at max capacity of 20,480 mac addresses. After the sample was recorded, I ran the attack again to generate all new mac addresses. I then took a second sample to compare to. If the forwarding table was a "rolling" table, the entries in both tables should be drastically different, and in the very least have entries that are in very different positions from one another. While examining the CAM tables more thoroughly, it was noticed that there were several *static* MAC address mappings, among the thousands of dynamic mappings. All of the dynamic mappings were mapped to the interface that connected to the attacker, which was expected. However, the static mappings pointed to the other virtual hosts in the virtual network.

It was determined that the Cisco Nexus 1000v has a MAC address *auto-learn* feature that is now enabled by default, which uses the integration with VMware to set a static MAC address mapping to all hosts connected to the switch. This feature was introduced in Cisco Nexus 1000v version 4.2(1)sv1(5.1) (Cisco Systems, Inc., 2014). Because all of the virtual machines have permanent MAC address table entries, these mappings can never be pushed out of the forwarding table by dynamic entries and will always supersede dynamic entries. This explains why the CAM table overflow attack was ineffective against the Nexus 1000v. Because of these static entries that have been installed automatically, a CAM table overflow attack becomes ineffective as seen in the release 5.2(1).

Both tested versions of the Nexus 1000V were susceptible to the VLAN hopping exploit. It was noted that in both cases, the double tagged frames were sent through the Nexus 1000V and processed accordingly. That is, the frame was passed into the Nexus 1000v switch, and the outer tag (VLAN 120) was stripped off, while the second tag remained. The Wireshark VM also recorded the packet being sent from the system-uplink of the Nexus 1000v switch to the Catalyst 3750G switch with the inner VLAN tag still present.

Beyond the Nexus 1000v egress interface, the packet is dropped at some point. The target's virtual interface was monitored in an attempt to verify if the traffic had made the "hop" across VLANs, however that packet for an unknown reason would always be dropped before it could be viewed on VLAN 110. These results were the same whether the attacker VM was connected to an interface on the Nexus 1000V set as trunk or access; the packets monitored in the switch appeared the same with both configurations. Monitoring the egress traffic of the uplink port channels on the Nexus 1000v indicated that the Nexus 1000V was forwarding the packets with the second VLAN tag intact to the physical Catalyst 3750G.

The attempt to exploit the spanning tree protocol on the Nexus 1000V was limited to the release 4.2(1). After the packets being sent by Yersinia had been verified, their impact on the Nexus 1000V was assessed. The most apparent observation was that the switch's functionality had persisted despite the attempted manipulations and no further STP BPDUs were identified on the network. Had the attack been successful, the switch would have been overwhelmed with CPU intensive computations and its forwarding functions would have been impacted. Because of this, the CPU usage of the VSM in vCenter was checked to further verify the impact of the attempts. It was found that the CPU usage had remained at a level consistent with normal usage, indicating that the spanning tree BPDUs were discarded without processing.

ARP Poisoning

Regardless of Nexus 1000V release tested, the ARP poisoning exploit was successful. Shortly after executing the ARP poisoning exploit, the ARP table of the Catalyst 3750G quickly changed and showed that all IP addresses, except for the gateway pointed to the MAC address of the attacker, Kali Linux. This indicates that the attack against the Nexus 1000V was successful and propagated those spoofed MAC addresses to the upstream switch as well. Figure 7 below shows a portion of the ARP table on the Catalyst 3750G after the attack was executed.

Protocol	Address	Age (min)	Hardware Addr	Type	Interface
Internet	172.28.120.104	0	0050.56a1.4530	ARPA	Vlan120
Internet	172.28.120.105	0	0050.56a1.4530	ARPA	Vlan120
Internet	172.28.120.96	1	0050.56a1.4530	ARPA	Vlan120
Internet	172.28.120.103	0	0050.56a1.4530	ARPA	Vlan120
Internet	172.28.120.1	-	000f.8fc6.abc4	ARPA	Vlan120

Figure 7 - Ending ARP Table for Poisoning Attack

After the ARP table was successfully poisoned, the ICMP traffic between client VMs was successfully redirected to the attacker VM as a man in the middle.

Finally, both tested releases of the Nexus 1000V proved vulnerable to an attack on Private VLANs. Upon executing the attack command, the target VM immediately captured ICMP packets with a destination IP address of itself and a source IP address of the attacker, indicating the ICMP packet was successfully sent to the target from the attacker across the private VLAN. However, the attacker did not receive the ICMP response packets as the client didn't craft special replies and the attacker didn't create an attack on the gateway that would forward messages back in a bi-directional manner.

5. CONCLUSIONS

The Cisco Nexus 1000v is a robust and well-implemented virtual switching solution for large virtual computing systems. However, it is clearly susceptible to many of the same vulnerabilities that other physical switches are vulnerable to as well. In a multi-tenant virtual environment, it appears that layer 2 exploits such as ARP poisoning, VLAN Hopping, and private VLAN exploits are a real threat, if they are not effectively mitigated. It appears that more recent versions of the software have used the integration with VMware to allow for a more secure and dynamic solution. While all of these attacks require the ability to craft frames and packets within specific VLANs, some would argue that these types of attacks are of little concern. However, with the growth of cloud services like IaaS and Paas, these attacks may become plausible from remote systems that have direct access to the VLANs. This causes significant concern, as successful exploitations at the lower layers can be leveraged for higher layer access (Layer 7 data for instance).

Many of these vulnerabilities can be mitigated with additional security and configuration, but if left exposed could create substantial problems. The popularity of virtualization has made solutions like this significantly more common, and because of this users of these VMware and Cisco products should to be aware of these potential vulnerabilities. As stated at the beginning of this paper, the lines between network administrator and systems administrator are becoming blurry with the usage of this technology. It is possible that a systems administrator with little networking experience or insight into these vulnerabilities could be the primary care taker of the solution, and could unknowingly leave gaping holes into the network without realizing it. This

could leave the data center, cloud provider, or other clients and customers exposed to potential threats.

It was also possible that the Nexus 1000V introduced new vulnerabilities. In particular, it was possible that the means of communication used to facilitate distributed switching had created vulnerabilities that were not present in physical switches. There was also potential that the mere fact the Nexus 1000V resided as a virtual machine could also pose as a security implication. Regardless of whether the potential vulnerabilities had previously existed in physical switches or were introduced with the transition to the virtual realm, each posed a potential security implication. Yet, only the vulnerabilities known in the physical architecture were considered here. In the future, additional insight into this vector should be examined.

Future work certainly should address many of the issues presented here. The Advanced (for pay) versions of the Nexus 1000V were not tested here, only the Essential versions. It is assumed that they will behave similarly against the attacks shown here. Cisco indicates, "the software image is the same for both the editions." [7] However, for factual verification of functionality, the 1000V Advanced must be tested. Additionally, more complex exploits could be crafted to identify limits of the Nexus 1000V. The deployment of the Nexus 1000V into the ESXi environment was not examined in depth during this study. There is the potential for a compromised version of the virtual switch to be deployed that could leak data or provide back door access. The potential exploitation of the platform itself is certainly of concern for the more than 10K deployed systems utilizing the Nexus switching platform. Lastly, the testing of multiple Nexus 1000Vs at data center scale should be evaluated and its inter-switch and switch-controller communications for passing of configuration information.