VLAN hopping, ARP poisoning and Man-In-The-Middle Attacks in Virtualized Environments

Ronny L. Bull*, Jeanna N. Matthews†, Kaitlin A. Trumbull‡
*†Clarkson University {bullrl, jnm}@clarkson.edu
*‡Utica College {rlbull, katrumbu}@utica.edu

***Abstract*—Cloud service providers offer their customers the ability to deploy virtual machines in a multi-tenant environment. These virtual machines are typically connected to the physical network via a virtualized network configuration. This could be as simple as a bridged interface to each virtual machine or as complicated as a virtual switch providing more robust networking features such as VLANs, QoS, and monitoring. At DEF CON 23, we presented how attacks known to be successful on physical switches apply to their virtualized counterparts. Here, we present new results demonstrating successful attacks on more complicated virtual switch configurations such as VLANs. In particular, we demonstrate VLAN hopping, ARP poisoning and Man-in-the-Middle attacks across every major hypervisor platform. We have added more hypervisor environments and virtual switch configurations since our last disclosure, and have included results of attacks originating from the physical network as well as attacks originating in the virtual network.**

Keywords*—*Virtualization, Networking, Network Security, Cloud Security, Layer 2 Attacks.

I. Introduction

With the growing popularity of Internet-based cloud service providers, many businesses are turning to these services to host their mission critical data and applications. Cloud customers often deploy virtual machines to shared, remote, physical computing resources. Virtual machines running in cloud capacity are connected to the physical network via a virtualized network within the host environment. Typically, virtualized hosting environments will utilize either a bridged network interface or a virtualized switch such as Open vSwitch[1], [2] for Xen and KVM based environments, or for VMware vSphere and Microsoft Hyper-V, the built-in virtual switch options or the Cisco Nexus 1000V[3] series virtual switch. These virtual switches are designed to emulate their physical counterparts, however, the majority of them do not provide any of the Layer 2 protection mechanisms found in modern enterprise grade hardware switches.

It is important for users of multi-tenant cloud services to understand how secure their network traffic is from other users of the same cloud services, especially given that virtual machines from many customers share the same physical resources. If another tenant can launch a Layer 2 network attack and capture all the network traffic flowing from and to their virtual machines, this poses a substantial security risk. By understanding which virtual switches are vulnerable to which attacks, users can evaluate the workloads they run in the cloud, consider additional security mechanisms such as increased encryption and/or increased monitoring and detection of Layer 2 attacks.

In this paper, we present the results of a systematic study to evaluate the effects of VLAN hopping and ARP poisoning attacks across five major hypervisor environments with seven different virtual network configurations. This can be considered a continuation of the work we presented at DEF CON 23 evaluating layer 2 network security[4], but this year we include testing of more sophisticated network configurations including VLANs and mixed physical/virtual environments. We begin by providing some basic background information on the general network configuration options available to virtualized environments. We then present the details of our test environment and then give detailed descriptions of the attack methodology used for each of our VLAN hopping and ARP poisoning scenarios, discuss the results, and provide mitigation strategies that could help to prevent the attacks from being successful. We conclude the paper by discussing related work and summarizing our results.

II. Basic Network Configuration Options

There are two types of networking configurations that are typically used in virtualized environments; bridging and switching. In this section, we describe both options and discuss how each one is applied within a virtualized network.

A. Bridging

Bridged mode is the simplest of configurations providing an interface dedicated to virtual machine use. A bridge connects two or more network segments at Layer 2 in order to extend a broadcast domain and separate each of the segments into their own individual collision domains[5]. A forwarding table[5], [6] is used to list the MAC addresses associated with devices located on each network segment connected to the bridge *(Figure 1)*. Requests are forwarded based upon contents of this table and the destination MAC address located in the Ethernet frame. A frame is forwarded across the bridge only if the MAC address in the destination block of the frame is reachable from a different segment attached to the bridge. Otherwise, the frame is directed to a destination address located on the same segment as the transmitting device or dropped.

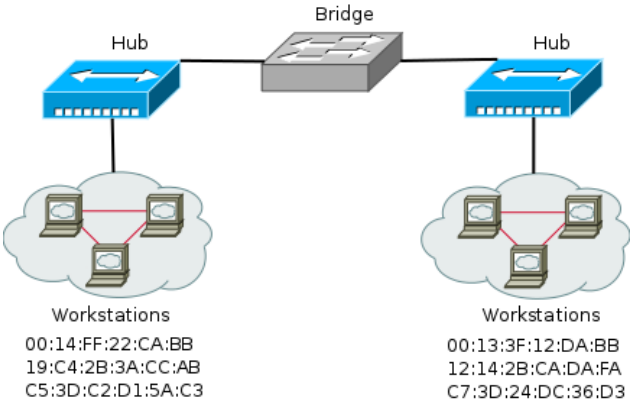


Fig. 1. A basic bridge using a forwarding table to pass requests between two network segments.

In virtualized environments, guest machines utilize user-space virtual network interfaces that simulate a Layer 2 network device in order to connect to a virtual bridge. Typically, the virtual bridge is configured and bound to a physical interface on the host machine that is dedicated solely to virtual machine traffic.

B. Switching

Physical switches have the capability of operating at Layer 2 or higher of the OSI model. Switches can be thought of as multi-port bridges[5] where each port of the switch is considered as its own isolated collision domain. Instead of a forwarding table, switches employ a CAM (content addressable memory) table[5] . Content addressable memory is specialized memory hardware located within a switch that allows for the retention of a dynamic table or buffer that is used to map MAC addresses of devices to the ports they are connected to *(Figure 2)*. This allows a switch to intelligently send traffic directly to any connected device without broadcasting frames to every port on the switch. The switch reads the frame header for the destination MAC address of the target device, matches the address against its CAM table, then forwards the frame to the correct device.

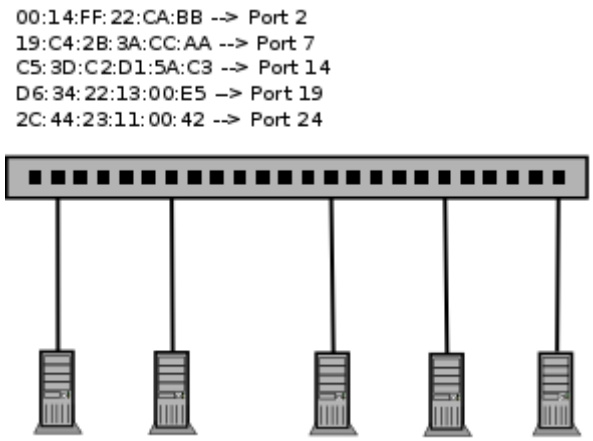


Fig. 2. A switch and its CAM table.

Virtual switches emulate their physical counterparts and are capable of providing features such as VLAN traffic separation, performance and traffic monitoring, as well as quality of service *(QoS)* solutions. Virtual machines are connected to a virtual switch by the way of virtual network interfaces *(VIF)* that are similar to the Layer 2 network devices used in conjunction with virtual bridges.

III. Test Environment

In this section, we provide details about our test environment which consisted of eight server class systems all located on a test network isolated from local production networks to avoid impacting them. Each new 1U SuperMicro server system consists of a quad core Intel Xeon X3-1240V3 processor running at 3.4GHz, 32GB of memory, a 500GB Western Digital Enterprise 7400 RPM SATA hard drive, and four on-board gigabit Ethernet ports. Having four Ethernet ports on each system allowed us to dedicate a port to the hypervisor operating system for management purposes, and also gave us the flexibility to use the other three ports for different virtual machine network configurations within each environment. This especially became useful when conducting the VLAN hopping experiments which will be discussed in more detail later in the paper.

Table I provides a list of the hypervisor environments and operating systems that were installed to the new hardware along with the virtual switch configuration used within each system.

TABLE I. Summary of hypervisor platforms and virtual switch configurations installed to the new hardware.

Hypervisor Platform	Virtual Switch
Gentoo OS Xen 4.5.1	Linux 802.1d Bridging
Gentoo OS Xen 4.5.1	Open vSwitch 2.4.0
VMWare vSphere ESXi 6.0.0	Standard ESXi Virtual Switch
MS Server 2012 R2 DataCenter w/Hyper-V	Standard Hyper-V Virtual Switch
MS Server 2012 R2 DataCenter w/Hyper-V	Cisco Nexus 1000v 5.2(1)SM3(1.1a)
ProxMox 3.4 *(KVM)*	Linux 802.1d Bridging
Citrix XenServer 6.5.0	Open vSwitch 2.1.3
Kali 2.0 Standalone System	No virtual switch

IV. Attacks Performed

Three new Layer 2 networking attacks were explored and thoroughly tested across all of the hypervisor environments specified in *Table I*: VLAN hopping via Switch Spoofing, VLAN hopping via Double Tagging and an ARP poisoning Man-in-the-Middle attack. Each attack was performed identically on all platforms in order to analyze the differences between the environments when subjected to the different attack scenarios.

A. VLAN Hopping via Switch Spoofing

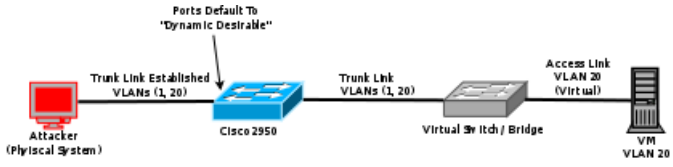
Switch spoofing is an attack that leverages a vulnerability[7] in physical Cisco switches that utilize the proprietary Dynamic Trunking Protocol *(DTP)* in order to automatically negotiate trunk links between switches. The majority of modern Cisco switches have DTP enabled by default on all ports out of the box so that trunk links

can easily be formed automatically. If physical ports on a Cisco switch are left in *dynamic desirable mode*, then an attacker can connect a system via any free switch port and fool the switch into thinking that their system is another switch looking to negotiate a trunk link. If the attack is successful and a trunk link is formed, the attacker will have access to all of the VLANs associated with the trunk thereby giving their system access to any system located on any of the corresponding VLANs. This attack has been well documented against physical networks in previous work performed by Cisco[8] and the SANS Institute[9]. There is also a powerful open source Layer 2 networking security auditing tool available called Yersinia[10] that can automate such a DTP attack against a switched Cisco network.

In this section, we discuss our evaluation of the effectiveness of executing a similar switch spoofing attack from a virtual machine. We attempt this attack within each of seven test environments connected to a Cisco 2950 switch on the physical network. For comparison, we begin with a control scenario in which we connected our physical Kali 2.0 system to a port on the Cisco 2950 switch to verify that the switch port can be successfully changed from dynamic desirable mode to trunking mode, and then moved on to evaluating if the same attack works when executed from a virtual machine connected to a virtual switch with an uplink to the same physical Cisco 2950 switch.

We adhere to the best practices guides[11], [12], [13] offered by the hypervisor manufacturers when setting up the physical switch ports connected to each virtual switch environment. Each of these guides suggests that the switch port be manually setup as a trunk port with access to each of the VLANs required for the virtual machines hosted within the environment. When testing the attack from each of the virtual networks, we made sure to convert the port that was connected to the system hosting the attacking virtual machine back to dynamic desirable mode from trunk mode in order to see if the virtual machine could successfully convert the physical switch port into trunk mode from the virtual network. In this case, we are suggesting that when the administrator connected the hypervisor environment to the physical switch they neglected to follow the best practices guide and never actually changed the switch port leaving it at its default setting of dynamic desirable. *Figure 3* illustrates the control scenario using the physical Kali 2.0 system, and *Figure 4* illustrates the scenario where the attacker is using a Kali 2.0 virtual machine located within one of the seven virtual test environments.

Fig. 3. Switch spoofing control scenario using a physical Kali 2.0 system to perform a DTP attack on a physical Cisco 2950 switch in order to gain unauthorized access to virtual machines on restricted VLANs.



We utilized the Yersinia tool via SSH in command line mode on each of the attacking systems in order to perform the attack. The attack process was straight forward and consisted of the following steps:

Fig. 4. Switch spoofing scenario where the attack is generated from a virtual machine connected to a virtual switched environment that has a physical uplink to a Cisco 2950 switch in order to gain unauthorized access to other virtual machines located on restricted VLANs within other hypervisor environments.

1) First the Yersinia application was loaded at the command line with *yersinia -I*.
2) Then the proper network interface was selected to use for the attack, in all cases the default network interface was used.
3) Yersinia was then changed to DTP mode by pressing *'g'* and selecting *'DTP Mode'*.
4) The attack was then conducted by pressing the *'x'* key and selecting option *'1'* to enable trunking mode.

If the attack was successful, the Yersinia application displayed *TRUNK/AUTO* in the DTP mode interface, otherwise if the attack failed *ACCESS/DESIRABLE* was displayed. We also verified if the attack worked by observing the interface and trunk status for the respective port associated with the attacking system on the Cisco switch by using the commands *sh int status* and *sh int trunk* from the console. This allowed us to see if the switch port was successfully converted into trunking mode or not. If the port was converted into trunking mode then the word *trunk* would be displayed under the VLAN column in the output of *sh int status*, and the interface would also appear in the trunk list with the word *auto* next to it in the output of *sh int trunk*.

The results of this attack varied across the different virtual network environments as shown in *Table II*. The control test from the physical Kali 2.0 system worked as expected and the port was put into trunking mode from dynamic desirable mode thus granting access to all of the virtual machines that were associated with VLANs available on the trunk. We simply loaded the 8021q kernel module on the attacking system, associated the target VLAN to the network interface and provided a valid IP address to the newly created VLAN tagged interface on the system. The following commands were used in order to set up this interface. Note that in this example the VLAN being targeted has the VLAN ID of *20*.

```
modprobe 8021q
vconfig add eth0 20
ifconfig eth0.20 192.168.1.10 netmask 255.255.255.0 up
```

This created a new network interface on the system labeled *eth0.20* which could be used to access the target systems located within the isolated VLANs on each of the virtual networks. The same process was used when testing from the virtual machines in order to validate the attack. *Table II* provides a summary of the results of the switch spoofing experiments.

The attack worked in the control scenario as well as three out of the seven virtual network environments. We see that if a virtual environment utilized a virtual bridged interface for virtual machine network connectivity the attack was successful,

TABLE II. SWITCH SPOOFING ATTACK RESULTS ACROSS THE SEVEN VIRTUAL TEST ENVIRONMENTS AND A PHYSICAL CONTROL SYSTEM. ✓INDICATES THE ATTACK WAS SUCCESSFUL.

	Results of Attack	
Platform	**Negotiate Trunk Link**	**Unauthorized VLAN Access**
Physical Kali 2.0 Control System	✓	✓
OS Xen w/ Linux Bridging	✓	✓
OS Xen w/ Open vSwitch		
VMWare vSphere ESXi	✓	✓
MS Hyper-V Standard vSwitch		
MS Hyper-V Cisco Nexus 1000v		
Proxmox	✓	✓
Citrix XenServer		

while environments that utilized a virtual switch for network connectivity prevented the attack from occurring. It can also be seen that the ESXi standard virtual switch allowed the attack to occur, indicating that this virtual switch is acting more like bridge than a switch. We have posted a demo video[14] of the successful attack from the ESXi environment to YouTube in order to document the process that was used on each of the seven environments for this experiment.

These results were a bit surprising since this attack is specific to a Cisco proprietary protocol and one would think that the attack would not be allowed to be passed from the virtual network to the physical switch as the DTP probes should be blocked. This was the case for each of the virtual switched environments since they were not compatible with the DTP protocol. However, the bridged interfaces also acted as a pass through allowing the attack to traverse through the virtual network and affect the physical switch.

We attempted to perform the attack directly against the Cisco Nexus 1000v switch to see if its virtual interfaces could be converted to trunking mode. When configuring the Nexus 1000v per the deployment guides[15], [16], we found that even connecting a virtual machine to the virtual switch required virtual subnets and policies that restricted which networks the virtual machines could access. This prevented the establishment of a trunk connection between the virtual machine and the Cisco Nexus 1000v virtual switch.

Switch spoofing attacks can be mitigated on physical Cisco switches by following a few best practices such as disabling unused switch ports to prevent unauthorized physical access to the switch as well as disabling the Dynamic Trunking Protocol on all ports. Limiting VLAN access on trunk connections is also a wise preventative action to reduce the likelihood of an attacker gaining unauthorized access to all of the VLANs on the network. Because DTP is a Cisco proprietary protocol, another way to mitigate this attack is to not use Cisco switches in the physical network.

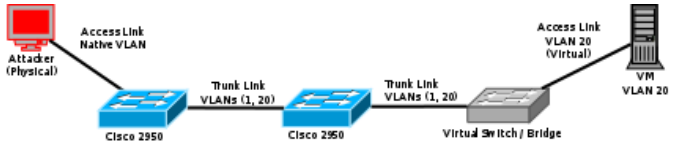
In terms of virtual networks connected to physical Cisco switches within a data center, it is important to recognize that this attack will work if the virtual network uses a bridged interface for virtual machine connectivity. In order to prevent this from occurring, administrators could either convert the virtual network to a secure virtual switched environment or lock down the physical switch to which the virtual platform is connected. The port could be secured by following best practices and ensuring that it is in trunk mode, and only has access to the specific VLANs that are required for the virtual network. Access to the native VLAN within the physical environment should also be blocked by removing it from the trunk VLAN access list on that specific port.

B. VLAN Hopping - Double Tagging

The VLAN hopping Double-Tagging or Double-Tagging VLAN jumping attack is an attack that leverages an inherent vulnerability in the 802.1q VLAN protocol[17] which allows an attacker to bypass network segmentation and spoof VLAN traffic by manipulating an Ethernet frame so that it contains two 802.1q VLAN tags. This attack requires two switches with a trunk connection established between them to be present in between the attacking system and the target system. When the Ethernet frame is pushed through the first switch the first 802.1q VLAN tag is stripped from the frame leaving only the second 802.1q VLAN tag. This tricks the second switch into thinking that the frame is destined for the target VLAN and it allows the frame to be forwarded on to the destination. One thing to note about this attack is that it is one-way unlike the switch spoofing attack described previously which allows for two-way communication between the attacking system and the target. By leveraging this vulnerability an attacker can send frames to target systems on isolated VLANs in order to perform denial of service attacks or create a one-way covert channel between the attacker and the target system.

We explored three different scenarios in order to evaluate the effectiveness of this attack within virtualized environments. All three scenarios require the use of at least one physical switch located between the attacking system and the virtualized network that is being targeted. *Figure 5* depicts the attack scenario where an attacker is using a physical Kali 2.0 system attached to a physical switch which has a trunk connection to a second physical switch with an established trunk link to each of the hypervisor environments.

Fig. 5. Double tagging scenario where the attack is generated from a physical Kali 2.0 system connected to a Cisco 2950 switch with a second Cisco 2950 switch located in between the first Cisco switch and the connected hypervisor environments.

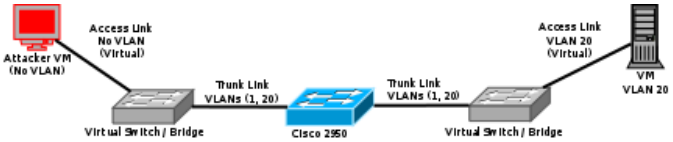


The second scenario as depicted in *Figure 6* still uses a physical Kali 2.0 system for the attack, however only a single physical switch sits in between the attacker and the virtual network.

In the third scenario, as illustrated in *figure 7*, the attacking system is a virtual machine connected to one of the seven virtual test networks that is trying to send a frame to a target system on another one of the virtual networks with a physical switch positioned in between both host systems.

Fig. 6. Double tagging scenario where the attack is generated from a physical Kali 2.0 system connected to a Cisco 2950 switch in order to gain unauthorized access to virtual machines located on restricted VLANs within connected hypervisor environments.

Fig. 7. Double tagging scenario where the attack is generated from a Kali 2.0 virtual machine within one of the connected virtual networks. A physical Cisco 2950 switch acts as the physical connectivity device located between each of the connected virtual networks.



To verify that the double-tagging VLAN hopping attack would work across the two Cisco 2950 switches we performed a test using two physical systems which served as a control for the rest of the experiments. The scenario depicted in *figure 5* was used with the virtual machine target being replaced with a CentOS 7 physical system connected to an access port on VLAN 20 on the second switch. The attack worked as expected, and we were able to send a frame from the attacking system located on VLAN 1 on the first switch to the target system located on VLAN 20 on the second switch.

The process that we used to evaluate the double-tagging attack throughout each of the scenarios remained the same. The only difference between the scenarios was the configuration of the network devices. We used the Yersinia tool on the attacking system in order to craft an ICMP request frame which consisted of two VLAN tags that was sent across the network. If the attack was successful we could view the ICMP request in tcpdump[18] on the target system which was located on a different VLAN than the attacker. The following process was used against all seven hypervisor environments in each scenario:

1) Connect to attacker system. *(SSH was used to access the physical attacking system, and in the case of a virtual attacker the console connection was used).*
2) Connect to the target system via the virtual machine console.
3) Connect to the console on each Cisco 2950 switch used in the experiment.
4) Verify switch port settings to confirm trunk and access port configurations.
5) Run *yersinia -I* on attacking system.
6) Select the network interface to be used by pressing *'i'*.
7) Select 802.1Q mode by pressing *'g'*.
8) Edit the IP address and VLAN information used for the attack by pressing *'e'*.
9) Run *tcpdump* in a terminal window on the target system and filter for ICMP traffic.
10) Launch the attack from the attacker system by pressing *'x'* then *'1'* to send the double tagged packet to the target.

We have documented the process for each of the three scenarios in a series of demo videos that have been posted to YouTube. The first video[19] highlights the attack as depicted in *figure 5*, where the attacker is using a physical Kali system with two Cisco 2950 switches located in between the attacking system and the target virtual machine. The target virtual machine in this case is a system located within the ProxMox hypervisor environment. The second video[20] illustrates the scenario where there is only a single Cisco 2950 switch located in between the physical attacking system and the target virtual machine as shown in *figure 6*. In this scenario the target virtual machine is located within the Microsoft Server 2012 Hyper-V environment using the Cisco Nexus 1000v virtual switch. In the third video[21] the attack is originated from a virtual machine located in the Citrix XenServer virtual environment and the target system is a virtual machine in the ProxMox environment. Both of the virtual networks are connected to trunk ports on a single Cisco 2950 switch as depicted in *figure 7*.

The results of the first two scenarios that utilized a physical attacking system are summarized in *table III*. The attack worked in both scenarios against every hypervisor environment other than the Microsoft Server 2012 Hyper-V environment which used the standard Hyper-V virtual switch. This was expected since once the double tagged frame passed through the trunk connection between the two physical switches the first VLAN tag was stripped and the frame was forwarded on to the second Cisco 2950 switch with only the second VLAN tag. At this point, the manipulation of the frame was complete, and any hypervisor environment connected to the second switch running a virtual machine on the target VLAN should have been able to see the frame.

In the single switch scenario, the virtual switch was the second switching device trunked with the Cisco 2950. The attack depends on the trunk link supporting the same native VLAN on both switches as well as both switches using 802.1q encapsulation for trunking. For that reason, it is especially interesting to see that the attack was effective against the majority of the virtual networks tested. The Microsoft Hyper-V environment configured with the standard virtual switch, however, was unaffected in both scenarios due to the same reason that prevented the MAC flooding attacks from working in our previous white paper[4]. The Hyper-V virtual switch also provided some minimal protection for virtualized network traffic which included protection against MAC address spoofing[22]. Since the Yersinia tool uses MAC address spoofing for the double-tagging attack the protection offered by the virtual switch prevented the traffic from entering the virtual network and reaching the target virtual machine.

Table IV summarizes the results of the third scenario where the attack is launched from a virtual machine located within one of the seven test environments, and the target system is another virtual machine located within a different hypervisor environment. In this scenario, we are testing to see if the attack can be successfully launched from within a virtual network. As can be seen by the results, the attack was successful in four out of the seven test environments. Any hypervisor using either 802.1d Linux bridging or Open vSwitch for virtual networking was vulnerable.

These experiments provide strong evidence that double-

TABLE III. PHYSICAL DOUBLE-TAGGING ATTACK SCENARIO RESULTS ACROSS THE SEVEN VIRTUAL TEST ENVIRONMENTS. ✓INDICATES THAT A FRAME WAS SUCCESSFULLY SENT FROM THE PHYSICAL ATTACKING SYSTEM TO A TARGET VIRTUAL MACHINE LOCATED ON VLAN 20 WITHIN THE CORRESPONDING HYPERVISOR ENVIRONMENT.

	Results of Attack	
Platform	**Single Switch**	**Double Switch**
OS Xen w/ Linux Bridging	✓	✓
OS Xen w/ Open vSwitch	✓	✓
VMWare vSphere ESXi	✓	✓
MS Hyper-V Standard vSwitch		
MS Hyper-V Cisco Nexus 1000v	✓	✓
Proxmox	✓	✓
Citrix XenServer	✓	✓

TABLE IV. VIRTUAL DOUBLE-TAGGING ATTACK SCENARIO RESULTS ACROSS THE SEVEN VIRTUAL TEST ENVIRONMENTS. ✓INDICATES THAT A FRAME WAS SUCCESSFULLY SENT FROM THE VIRTUAL ATTACKING SYSTEM TO A TARGET SYSTEM LOCATED WITHIN A SEPARATE VIRTUAL NETWORK ON VLAN 20.

	Results of Attack
Platform	**Virtual Switch**
OS Xen w/ Linux Bridging	✓
OS Xen w/ Open vSwitch	✓
VMWare vSphere ESXi	
MS Hyper-V Standard vSwitch	
MS Hyper-V Cisco Nexus 1000v	
Proxmox	✓
Citrix XenServer	✓

tagging VLAN hopping attacks should be considered a serious threat to virtualized environments. In order to protect the virtual machines located within these environments, we have some specific suggestions for configuring the physical switches to which the hypervisors are connected. Administrators should avoid assigning any hosts to the native VLAN *(typically VLAN 1)* on any physical switches that are serving as uplinks for virtual networks. If VLANs are to be used within hypervisor environments for virtual machines, it is necessary to connect the virtual switch to a trunk port on the physical switch. This trunk port should not be configured to carry native VLAN traffic since the double-tagging attack depends on having access to the native VLAN in order to get that first 802.1q tag stripped out of the frame. Our results show that even though the double-tagging attack requires two switches to be successful a virtual switch could easily act as the second switch allowing the attack to reach the target destination. As of right now, it is not possible to configure the virtual switch to stop these attacks so it is important to focus on making sure that the switches that connect the virtual networks to the physical world are secure.

C. ARP Poisoning Man-in-the-Middle Attack

The Address Resolution Protocol *(ARP)* is a Layer 2 networking protocol that is used to map the physical MAC addresses of connected devices within a broadcast domain to their logical Layer 3 IP addresses. Each device on the network maintains an ARP cache which is a table that is dynamically updated when a device discovers other devices located within the same Layer 2 network. When a system is initially placed on a network, the ARP cache is empty and is filled with new entries as the system begins to communicate with other systems either directly or via broadcast transmissions. Typically, the first entry added to the ARP cache is the default gateway for the network. The process of updating the ARP table is rather simple. If a system on a network does not know the physical MAC address of another system within the broadcast domain, it will send out a broadcast transmission to every connected device asking who has that specific Layer 3 IP address. Once the system that is assigned the target Layer 3 IP address receives the Layer 2 ARP broadcast, it sends a unicast reply back to the requesting system with its physical Layer 2 MAC address. The requesting system then updates its ARP cache so that it does not need to send out the broadcast request again when it needs to establish future connections to that particular system.

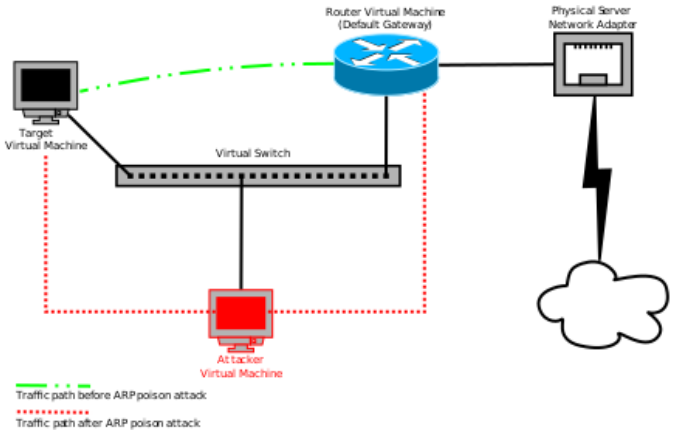
The ARP protocol has been proven to be vulnerable to Man-in-the-Middle attacks[23], [24] where an attacker can manipulate the ARP cache on a target system in order to place themselves in the middle of the communication stream to either sniff or manipulate the traffic going between the systems. This attack is so well known that open source tools[25], [26] have been developed to make it very easy for an attacker to take advantage of the vulnerability.

We tested the effects of an ARP poisoning Man-in-the-Middle attack on each of the virtual network configurations within our test environment. In order to conduct the experiments, each hypervisor environment was allocated two Kali 2.0 virtual machines and a CentOS 7 router system that acted as the default gateway for the virtual network providing access to the Internet. One of the Kali 2.0 virtual machines was setup as the attacking system and the other was the target system. The goal of the experiment was to poison the ARP cache of both the target Kali 2.0 virtual machine and the default gateway in order to place the attacking Kali 2.0 system in the middle of the communication stream and sniff the traffic going from the target system through the default gateway to the Internet. *Figure 8* provides a network diagram of the attack scenario illustrating the traffic paths from the target system before and after the attack.

In order to streamline the attacks across our seven test environments we opted to use a modified version of the ARP cache poisoning Python/Scapy script found in the Black Hat Python[27] book, and a simple custom BASH script using tcpdump to monitor the sniffed traffic. The scripts allowed us to effectively automate the experiments through SSH within each of the hypervisor environments. The following procedure was performed within each virtualized environment in order to evaluate the effects of the attack within the respective virtual network:

1) Open an SSH terminal connection to each virtual machine *(router, target, and attacker)*.
2) Run *arp -a* on each virtual machine in order to document the initial ARP cache state.
3) Enable IP forwarding on the attacker system *(echo 1 >/proc/sys/ipv4/ip_forward)*.

Fig. 8. ARP poisoning Man-in-the-Middle attack scenario diagram.



4) Run the Python/Scapy script on the attacker system to poison the ARP cache of both the router and target systems.
5) Run *arp -a* again on each virtual machine in order to document the modified ARP cache state.
6) Run a continuous ping from the target system to www.google.com.
7) The Python/Scapy script sets up a sniffer to collect the traffic and dumps it to a pcap file, then when finished it restores the ARP cache back to normal on the router and target systems.
8) Run *arp -a* again on each virtual machine in order to document the restored ARP cache state.
9) Run the tcpdump script on the pcap file to view the results.

We have posted narrated demo videos of the ARP poisoning Man-in-the-Middle attack experiments within the VMWare ESXi[28] and the Microsoft Hyper-V/Cisco Nexus 1000v[29] environments to YouTube to document the process. *Table V* summarizes the results of the experiment across each of the virtualized platforms that were tested. As can be seen by the results, the attack was successful in each of the seven environments allowing an attacker to manipulate the ARP cache tables of any virtual machine located within the same broadcast domain on the virtual network.

TABLE V. ARP POISONING MAN-IN-THE-MIDDLE ATTACK RESULTS ACROSS THE SEVEN VIRTUAL TEST ENVIRONMENTS. ✓INDICATES THE PLATFORM WAS AFFECTED.

	Results of Attack	
Platform	**Manipulate ARP Cache**	**Eavesdropping Allowed**
OS Xen w/ Linux Bridging	✓	✓
OS Xen w/ Open vSwitch	✓	✓
VMWare vSphere ESXi	✓	✓
MS Hyper-V Standard vSwitch	✓	✓
MS Hyper-V Cisco Nexus 1000v	✓	✓
Proxmox	✓	✓
Citrix XenServer	✓	✓

Out-of-the-box, all the virtual network environments that we tested provide no protection against this type of attack. In order to mitigate an ARP cache poisoning attack on a physical network, specifically within Cisco switches, an administrator may make use of DHCP Snooping and Dynamic ARP Inspection *(DAI)*, with DHCP Snooping being a prerequisite for enabling DAI[24]. Dynamic ARP Inspection is effective at mitigating ARP based attacks because it intercepts all ARP requests and responses, and verifies their authenticity prior to forwarding the traffic to the destination[24]. Currently none of the virtual networks that were tested provide this level of functionality, though it is available in the advanced *(*non-free) version of the Cisco Nexus 1000v virtual switch. There are however utilities available that could be run as a service on a separate system running on the virtual network to monitor for changes in ARP activity on the network. An open source Linux service called *arpwatch* can be setup to monitor the network for changes in MAC address and IP address pairings and alert a network administrator via email when changes occur[30].

V. RELATED WORK

There has already been a substantial amount of work studying the vulnerability of physical networks to Layer 2 attacks [8], [9], [31], [32], [33], but the impact on virtual networks has not received as much attention. This is beneficial in the fact that published research previously performed on physical networks can serve as a model for testing in virtual environments and comparisons can be made based upon the physical baselines. For instance, *Yeung et al.*[31] provide an overview of the most popular Layer 2 networking attacks as well as descriptions of the tools used to perform them. This work was very helpful in identifying possible attack vectors that could be emulated within a virtualized environment. *Altunbasak et al.*[32] also describe various attacks that can be performed on local and metropolitan area networks, as well as the authors' idea of adding a security tag to the Ethernet frame for additional protection. Cisco also published a white paper[8] regarding VLAN security in their Catalyst series of switches. The paper discloses testing that was performed on the switches in August of 2002 by an outside security research firm @stake which was acquired by Symantec in 2004. In the white paper, they discussed many of the same attacks that were mentioned by *Yeung et al.*[31], however the authors also went into detail about best practices and mitigation techniques that could be implemented on the physical switches in order to prevent the attacks from being successful.

VI. FUTURE WORK

Going forward, we are especially interested in working with cloud service providers and data center operators to assess the vulnerability of their environments to the Layer 2 attacks that we have discussed this paper as well as in our previous work[4], [34]. Understandably, it is unacceptable to run such experiments without the permission and cooperation of the service provider. We hope that these results highlight that users should have the right to ask service providers to document what additional defenses - either prevention or detection - if any they are providing to protect users from these types of attacks on their systems.

VII. Conclusion

This study and the work we presented at DEF CON 23 demonstrates the degree to which virtual switches are vulnerable to Layer 2 network attacks, as well as the effect that these attacks could have on the physical network infrastructure to which the virtual switches are connected. The Layer 2 vulnerabilities described in this paper are directed towards the virtual networking devices and not the hypervisor and without additional mitigation or preventive measures, could be performed on any host running a virtual switch including in a multi-tenant environment.

We have performed an extensive Layer 2 security assessment on the state of virtual networking devices. In their current state, virtual switches pose the same liability as their physical counterparts in terms of network security. However, the lack of sophisticated Layer 2 security controls like those present on enterprise grade physical switches increase the level of difficulty in securing these environments. One malicious virtual machine performing any one of these Layer 2 attacks against the virtual switch could be able to sniff, redirect, or prevent traffic from passing over that virtual switch, potentially compromising the confidentiality, integrity, and availability of co-located clients.

References

[1] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby, "Virtual switching in an era of advanced edges," in *ITC 22 2nd Workshop on Data Center - Converged and Virtual Ethernet Switching (DC-CAVES)*, 2010.

[2] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending networking into the virtualization layer," in *HotNets-VIII*, 2009.

[3] Cisco Systems, Inc. Cisco nexus 1000v series switches for vmware vsphere data sheet. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9902/data_sheet_c78-492971.html

[4] R. L. Bull and J. N. Matthews. Exploring layer 2 network security in virtualized environments. [Online]. Available: https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEFCON-23-Ronny-Bull-Jeanna-Matthews-Exploring-Layer-2-Network-Security-In-Virtualized-Enviroments-WP.pdf

[5] R. Seifert and J. Edwards, *The All-New Switch Book*. Indianapolis, Indiana: Wiley Publishing, Inc., 2008.

[6] LAN MAN Standards Committee, *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges*. New York, NY: The Institute of Electrical and Electronics Engineers, Inc., 2004.

[7] National Vulnerability Database. Vulnerability summary for cve-1999-1129. [Online]. Available: http://www.cvedetails.com/cve/CVE-1999-1129/

[8] Cisco Systems, Inc. Vlan security white paper [cisco catalyst 6500 series switches]. [Online]. Available: http://www.cisco.com/en/US/products/hw/switches/ps708/products_white_paper09186a008013159f.shtml#wp39211

[9] S. Rouiller. Virtual lan security: weaknesses and countermeasures. [Online]. Available: http://www.sans.org/reading-room/whitepapers/networkdevs/virtual-lan-security-weaknesses-countermeasures-1090

[10] A. A. Omella and D. B. Berrueta. Yersinia. [Online]. Available: http://www.yersinia.net/

[11] Citrix. Xenserver vlan networking. [Online]. Available: http://support.citrix.com/article/CTX123489

[12] VMWare. Sample configuration of virtual switch vlan tagging. [Online]. Available: https://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1004074

[13] A. Fazio. Understanding hyper-v vlans. [Online]. Available: https://blogs.msdn.microsoft.com/adamfazio/2008/11/14/understanding-hyper-v-vlans/

[14] R. L. Bull. Switch spoofing attack against a cisco 2950 switch from the vmware esxi 6.0 hypervisor environment. [Online]. Available: https://youtu.be/mMGezerlg9c

[15] K. Holman. Scvmm 2012 r2 quickstart deployment guide. [Online]. Available: https://blogs.technet.microsoft.com/kevinholman/2013/10/18/scvmm-2012-r2-quickstart-deployment-guide/

[16] Cisco Systems, Inc. Installing the cisco nexus 1000v. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus1000/hyperv/sw/5_2_1_s_m_3_1_1/install-and-upgrade/guide/n1000v_gsg/n1000v_gsg_1_HV_install.html

[17] CVE Details. Vulnerability details: Cve-2005-4440. [Online]. Available: http://www.cvedetails.com/cve/CVE-2005-4440/

[18] Tcpdump Project. Tcpdump & libpcap. [Online]. Available: http://tcpdump.org

[19] R. L. Bull. Double tagging vlan hopping attack against the proxmox virtual network using two physical switches. [Online]. Available: https://youtu.be/V2Ht-GB4NbE

[20] ——. Double tagging vlan hopping attack against the hyper-v cisco nexus 1000v virtual network using a single physical switch. [Online]. Available: https://youtu.be/np46KuXpk9c

[21] ——. Double tagging vlan hopping attack between two virtual networks connected to a cisco 2950 switch. [Online]. Available: https://youtu.be/jJDBJRoukIo

[22] Microsoft. Hyper-v virtual switch overview. [Online]. Available: http://technet.microsoft.com/en-us/library/hh831823.aspx

[23] Department of Homeland Security. Alert (ta15-120a) securing end-to-end communications. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA15-120A

[24] Cisco Systems, Inc. Arp poisoning attack and mitigation techniques. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603839.html

[25] C. M. Sheilds and M. M. Toussain. Subterfuge: The automated man-in-the-middle framework. [Online]. Available: https://www.defcon.org/images/defcon-20/dc-20-presentations/Toussain-Shields/DEFCON-20-Toussain-Shields-Subterfuge-WP.pdf

[26] A. Ornaghi and M. Valleri. The ettercap project. [Online]. Available: https://ettercap.github.io/ettercap/

[27] J. Seitz, *Black Hat Python*. San Francisco, California: No Starch Press, 2015.

[28] R. L. Bull. Arp poisoning attack in the vmware esxi 6.0 hypervisor environment. [Online]. Available: https://www.youtube.com/watch?v=1h-pbTktCwI&feature=youtu.be

[29] ——. Arp poisoning attack in ms server 2012 hyperv using the cisco nexus 1000v virtual switch. [Online]. Available: https://www.youtube.com/watch?v=F6X9GsmOwbY&feature=youtu.be

[30] Lawrence Berkeley National Laboratory. arpwatch linux man page. [Online]. Available: http://linux.die.net/man/8/arpwatch

[31] K.-H. Yeung, D. Fung, and K.-Y. Wong, "Tools for attacking layer 2 network infrastructure," in *IMECS '08 Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2008, pp. 1143–1148.

[32] H. Altunbasak, S. Krasser, H. L. Owen, J. Grimminger, H.-P. Huth, and J. Sokol, "Securing layer 2 in local area networks," in *ICN'05 Proceedings of the 4th international conference on Networking - Volume Part II*, 2005, pp. 699–706.

[33] K. Lauerman and J. King. Stp mitm attack and l2 mitigation techniques on the cisco catalyst 6500. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_605972.pdf/

[34] R. L. Bull. Derbycon 4.0: Exploring layer 2 network security in virtualized environments. [Online]. Available: http://youtu.be/tLrNh-34sKY