Matthew Moltzau

Ryan Vacca

Julia Vrooman

**Final Project Progress Report - ARP spoofing detector**

Project Description:

Spoofed ARP Detector. The goal of this project is to develop a tool that detects ARP spoofing in LANs and WLANs. You must demo your tool in a testbed LAN.

**Topics**

**- A more in-depth definition of your project, based on your new findings.**

Based on our research into this topic and labs in class, we have formalized methods to initiate and detect ARP spoofing. ARP spoofing occurs when an attacker tells its target that has an IP address that actually doesn't belong to itself. The spoofing can be set up in such a way that data passed to the router will then be intercepted by the attacker and can be read, allowing for session hijacking and other man-in-the-middle attacks. When spoofing, ARP requests are continuously sent out in order to keep the poisoned cache fresh and when requests are no longer sent, the targeted machine is able to correct the poisoned cache. In our initial testing, we have used the tool `arpspoof` in order to create a spoofing, but in the future we will dissect data from wireshark in order to model ARP packets sent out using our own python program. The `arpspoof` tool seems to only make a man-in-the-middle attack between a router and target, however with our program we hope to drop the internet connection of the target as well by not forwarding requests to the router.

One easy way to detect ARP spoofing is to check the system ARP table for duplicate MAC addresses. On linux, this is accessible through both the `arp` command and `/proc/net/arp`. On other operating systems the ARP table may also be stored in RAM, but in either case, the table is developed by adding entries from received ARP packets. The attacking machine may not be listed in the table though, so our implementation will seek a proactive approach that reaches out to all machines on the network. In order to mitigate this, the attacker may have a firewall that blocks ARP replies, so this would be interesting to experiment with. Outside of detecting duplicates, a suspicious number of ARP requests should be considered in logging an alert. At the moment we don't know what our baseline is, so this is something we must record and track in both a normal network and actively-spoofed network. Regarding the changing of an ARP table entry, we expect the IP address of a machine on a network will change eventually, however it would be suspicious if it changed frequently or went back to an old address. Our program will seek to retain knowledge of changes up to a certain amount in order to detect more anomolies.

After these measures have been implemented, we may inspect other tools such as `arpwatch`, `arptables`, or `arpon` for a closer look into how they work. Some of these not only detect changes, but also block packets as well. The prevention of ARP spoofing is currently out of scope for this project, but if we have time we will consider adding it. ARP packets can be blocked with a firewall such as `iptables`, though it is important to note without some form of state, the `iptables` by itself is not enough since it operates on layers 3 and 4 of the OSI Network Model, which doesn't include MAC address information. Other forms of detection would be limited as well. The blocking of packets may be done for the computer running the program itself, however when listening over the network, we will not be able to drop the packets for another machine.

**- A progress report, in accordance with the initial guidelines you submitted before.**

*Timeline:*

*SEP: Discuss program architecture1, decide language of choice2.*

*OCT: Configure Virtual network(s)3, research ARP python spoofing4, develop prototype5.*

Up to this point we have remained on schedule that we proposed in the timeline of our project proposal.

1. We have discussed the program architecture and decided on a number of relevant implementation details, including constraints on dependencies. The primary working component will include a sniffer that dissects packets and keeps information in a table -- similar to the port scanner detector made for a previous homework. The spoofing itself will be done in a program that will make continuous ARP requests with the victim's IP address as one of the input parameters.
2. We have decided on python3 for our programming language version, though bash will be used for some quick prototyping using already-created tools for testing and comparison.
3. We have configured the virtual network to contain an attacker and multiple victims, one of which runs on Kali Linux for our detection program. The virtual network created includes a virtual router that will be used as well.

* Machine #1: This machine will serve as the attacker that runs the ARP spoofing program.
* Kali VM #2: This virtual machine will serve as both the target of the attack and the detector. The first working defense will be for itself, but it will also be responsible for detecting ARP spoofing on other machines by listening promiscuously over the network.
* Windows Server VM / Kali VM #3: This virtual machine will serve as the secondary target machine on the LAN, to simulate the victim with no protection and test whether or not promiscuous mode works.

1. We have researched the ARP spoofing techniques and have found that we can manually make an ARP request to hijack the victims IP-Address. Refer to the first section of this document for a more in-depth description of ARP spoofing.
2. As of now, we have performed ARP spoofing using the `arpspoof` tool and detected it using both by using `arpwatch` and by manually looking at the contents of the ARP table. We have a sniffer implementation we can port for our needs and being to dissect ARP packets, though we have not done this yet. We have run wireshark in order to demonstrate the effect of ARP spoofing, but have not formed a baseline from the data yet.

**- Remaining tasks and your detailed plans toward them.**

NOV: Develop prototype into final product, testing, and documentation.

DEC: Prepare products for turn-in, practice delivery mechanism for class presentation.

Moving forward there are a number of elements that need to be created, however the project is well defined and understood. We need to dissect ARP packets and store information in a table. Because we expect duplicates, the table entry will contain a list of additional information for us to sort through. After we define a baseline for expected behavior, we will add a few simple rules. The arp spoofer itself needs to be created, and for that we will inspect wireshark data to give us information in order to create a packet.