What's lurking in the Ether?

Security in an Ethernet LAN environment

Nicholas J. Smith
July 4, 2000

Introduction

Tools that have network sniffing capabilities have become increasingly popular over the last few years for legitimate uses, such as networking troubleshooting, as well as for unethical uses such as password sniffing. Given the ease with which these tools can be obtained and installed, they can represent a significant risk to the security of an organization from both internal and external attackers.

One of the countermeasures that is often cited to help reduce the effectiveness of sniffing by attackers is the use of switched Ethernet LANs. However, unless adequate attention is paid to the configuration of these switches and the systems connected to them, it is still possible to compromise connections with the use of the appropriate tools.

The purpose of this paper is to get you thinking about security in an Ethernet LAN environment. In order to convince you that it is necessary to pay more attention to this area, a number of tools and techniques will be considered, including an increasingly popular session hijacking tool that is capable of compromising connections on switched Ethernet LANs.

Shared medium vs. switched Ethernet networks

Ethernet is a popular LAN technology that uses a broadcast medium to enable a network device to transmit to and receive data from other network devices. Because the network segment is a shared medium, only one device is allowed to transmit at a time and *all* other devices listen to the transmission to determine if the data is intended for them. Although this is a very simplistic description of Ethernet operation (more details can be found here), it serves to illustrate that data transmissions on an Ethernet segment are not private.

Because it is only possible for one device to transmit at a time, devices on an Ethernet segment contend for the right to transmit data. As the number of users on an Ethernet network increases, the likelihood of collisions occurring also increases. When collisions occur as a result of two or more devices trying to transmit data, network performance tends to suffer. One technique that can be used to increase network performance is to segment a network into smaller collision domains so that there are less devices contending for the bandwidth. Taken to the extreme, a network can be segmented so that each network device has its own segment. This is the basis of an Ethernet switch (more information on LAN switching can be found here).

Devices, such as servers, workstations, routers and firewalls, are connected to their own ports on an Ethernet switch. The switch maintains a table that maps the physical address

(MAC) information of network devices to port numbers on the switch so that when a device transmits data, the switch can send the data directly from the port of the transmitting device to the port of the receiving device. A switched Ethernet connection operates like a network with only two devices, so as well as increasing network performance, data is kept away from devices that are not intended to receive it.

Sniffers and beyond

Sniffers are software tools that are capable of gathering traffic from a network. In order to gather network traffic, the sniffer must put the Ethernet interface into promiscuous mode. As we have seen already, traditional Ethernet is inherently vulnerable to sniffing due to its broadcast nature. An interface in promiscuous mode will not only see all the traffic on that network segment, but it will gather all the traffic and pass it to the sniffer software. However, if the device running the sniffer program is moved to a switched Ethernet connection, it will only gather data that is going to or from the device on which the sniffer is running. Thus, deploying an Ethernet switch in the LAN environment can be a useful in reducing the effectiveness of most sniffers.

Sniffers are among the most common of hacker tools and are particularly useful in what is known as an "Island Hopping Attack". These attacks typically involve an attacker gaining control of a single machine through some exploit and then installing a sniffer. The sniffer enables the attacker to observe users and administrators logging into other systems and collect plain text passwords that travel along the network during the authentication process. In this way, an attacker can quickly gain access to and possibly take over many other systems.

There are many examples of sniffers. The better ones, such as snoop, sniffit, and tcpdump, provide a variety of features for filtering, capturing and recording network traffic. dsniff is also a very capable tool that has the ability to forge ARP replies (more on ARP spoofing later), which enables it to capture plain text passwords in a switched Ethernet LAN environment. Network Intrusion Detection (NID) tools, such as snort, can be thought of as sniffers with sophisticated filtering and recording features. There are also tools available that have a sniffing capability that augments the primary purpose of the tool. A good example is L0phtCrack, which features packet capture to collect LANMAN password hashes as they travel over the network.

However, there is a class of tools that use network sniffer functionality to achieve loftier goals. These are called session hijacking tools and include Hunt, Juggernaut, T-sight and IP-Watcher. These tools enable an attacker to steal an interactive login session, such as telnet, issue commands as if they were the trusted user, and even give the session back to the original user when the attacker is done.

It should be noted that if encrypted connections are used, the effectiveness of current tools that incorporate sniffer functionality is reduced to zero.

The basics of session hijacking

Session hijacking is a fairly sophisticated form of attack that can involve a number of techniques to achieve its goal. These techniques are briefly described in their own right and in the context of session hijacking.

IP spoofing. IP spoofing exploits implied trust between systems that use authentication based on IP addresses. IP spoofing works by fooling a target system into thinking that an attacker's machine is a trusted system. It is achieved by faking the IP source address of the trusted system and predicting the TCP sequence number of the target system (more details can be found here). These days, IP spoofing is old hat, although Kevin Mitnick gained much notoriety for his use of this technique (see The Kevin Mitnick / Tsutomu Shimomura affair). Countermeasures against IP spoofing include the application of ingress and egress filters on packet forwarding devices to prevent packets with spoofed source IP addresses from reaching their target and the use of TCP/IP software that generates hard to predict sequence numbers. The difference between 'classic' IP spoofing and the use of IP spoofing in session hijacking is that the former just exploits the trust relationship between a target host and a trusted system in order to remotely execute a command on the target system, whereas the latter takes over an existing connection to the target. This means that even if strong authentication, such as a hardware token, is used, it is still possible to hijack the session.

DoS attack. A DoS attack is one that prevents any part of an information system from functioning in accordance with its intended purpose. Often this takes the form of flooding the resources of a system to prevent it from servicing normal and legitimate requests. DoS attacks often involve the attacker using a spoofed source IP address. Therefore, the use of ingress and egress filters can also mitigate against a DoS attack by preventing packets with spoofed source IP addresses from reaching their target. During session hijacking, when the attacking system intrudes on the connection and starts spoofing packets, the TCP sequence numbers between the trusted system and the target host will get out of synchronization. Unfortunately for the attacker, the responses from the target system will still reach the trusted system. If the attacker does nothing about this, an ACK storm will result as the trusted system and the target host try to resynchronize their connection by sending SYNs and ACKs back and forth. The attacking system can prevent the ACK storm from occurring by launching a DoS attack against the trusted system.

Network sniffing. The basics of network sniffers have already been described. For session hijacking, a sniffer serves two purposes. Firstly, it enables existing connections to be observed prior to the hijack. Secondly, it enables the hijack to become interactive by observing the responses from the target system. This second point deserves some emphasis because a factor that is often overlooked in IP spoofing is the fact that the attack is blind because packets from the target will be routed back to the trusted system and not to the attacking system. However, if the attacking system is located on a network segment that is passing traffic between the trusted system and the target host, a sniffer can gather the responses from the target system.

Let's put the above techniques together to describe a basic session hijacking attempt in a step-by-step fashion. The players are Alice (trusted system), Bob (target host), and Eve (attacking system). Alice, Bob and Eve are all located on shared medium Ethernet LANs, but are not necessarily on the same LAN segment. Eve could be located on Alice's LAN, Bob's LAN or an intermediate LAN, provided that the LAN segment is passing traffic between Alice and Bob. The steps are:

1. Alice opens a telnet session to Bob and starts doing some work.
2. Eve observes the connection between Alice and Bob using a sniffer that is integrated into her hijacking tool. Eve makes a note of Alice's IP address and her hijacking software samples the TCP sequence numbers of the connection between Alice and Bob.
3. Eve launches a DoS attack against Alice to stop Alice doing further work on Bob and to prevent an ACK storm from interfering with her attack.
4. Eve generates spoofed packets with the correct TCP sequence numbers and connects to Bob.
5. Bob thinks that he is still connected to Alice.
6. Alice notices a lack of response from Bob and blames it on the network.
7. Eve finds herself at a root prompt on Bob. She issues some commands to make a backdoor and uses the sniffer to observe the responses from Bob.
8. After covering her tracks, Eve logs out of Bob and ceases the DoS attack against Alice.
9. Alice notices that her connection to Bob has been dropped.
10. Eve uses her backdoor to get directly into Bob. While looking around the system, the attacker on Eve discovers that Bob is already 'owned' by another attacker and that part of the root kit includes a neat session hijacking tool called Hunt. The attacker on Eve decides to do some reading to find out more about this tool.

Hunt

Hunt is considered by many to be one of the best session hijacking tools available because it is well written and has a comprehensive feature set. Hunt may not have the nice GUI of similar tools such as IPWatcher and T-sight, but the text based user interface is fairly easy to use and has the benefit of enabling Hunt to be used over a telnet session.

Hunt was developed by Pavel Krauz and is freely available for download from ftp://ftp.gncz.cz/pub/linux/hunt/. In the words of its author, "the main goal of the HUNT project is to develop [a] tool for exploiting well known weaknesses in the TCP/IP protocol suite". Pavel Krauz's Home Page does not provide much information on Hunt, but the README file that accompanies the source code is more enlightening.

Hunt's hijacking capabilities are primarily aimed at telnet and rlogin traffic and enable an attacker to view active sessions on an Ethernet LAN and then select one of them to hijack. The features of version 1.5 of Hunt include:

- Detection and watching of active connections.
- Insertion of commands into a session.
- Total takeover of a session.

- Synchronization of the original client with the server after a hijack.
- Connection reset.
- Network sniffing with the ability to search for a particular string.
- Handling of ACK storms with ARP (Address Resolution Protocol) spoofing.

The ARP spoofing capability of Hunt is of particular interest to us.

ARP spoofing with Hunt

ARP enables systems to map IP addresses to the machine's physical addresses. For systems connected to Ethernet LANs, ARP maps IP addresses to MAC addresses (more details can be found here). Returning to the scenario with Alice and Bob, when Alice wants to send some data to Bob, the ARP program on Alice will first look in ARP cache to see if there is already an entry for Bob's IP address. If there is a mapping, Alice can address the data with Bob's MAC address and send it to him. If no mapping exists, the ARP program on Alice will send out an ARP request to all machines on the Ethernet segment. Under normal conditions, only Bob will answer. Alice can now send her data to Bob and will store Bob's MAC address in her ARP cache.

A weakness of ARP is that a machine can send out an ARP reply without there having been an ARP request. Most systems will accept this forged answer and will update their ARP cache to accommodate it. This weakness allows ARP spoofing to take place.

To demonstrate how Hunt uses ARP spoofing, the steps in our original hijack scenario have been re-written with Hunt being used as the session hijacking tool. For simplicity, it is assumed that Alice, Bob and Eve are on the same network segment. Now the steps are:

1. Alice opens a telnet session to Bob and starts doing some work.
2. Eve uses Hunt to observe all connections passing her location on the network. Seeing the connection between Alice and Bob, Eve selects it for hijacking.
3. Eve sends an ARP reply to Alice, mapping Bob's IP address to a MAC address that does not exist on the LAN segment.
4. Eve sends an ARP reply to Bob, mapping Alice's IP address to a MAC address that does not exist on the LAN segment.
5. Alice and Bob will be trying to send data to each other, but because their respective ARP caches contain mappings to non-existent MAC addresses, the data will not arrive at the intended destination. However, Eve, who is strategically located in the middle and listening in promiscuous mode, is able to capture all traffic between Alice and Bob.
6. Eve can use Hunt's ARP daemon to control the traffic between Alice and Bob. She can insert commands, completely take over the session or simply relay all the traffic between Alice and Bob.
7. All this time Bob thinks that he is still connected with Alice.
8. Alice will notice a lack of response from Bob if Eve hijacks the session. Note that during the hijack there will not be an ACK storm because Alice is not receiving data from Bob.

It should be noted that this technique works regardless of the relative locations of Alice and Bob, provided that Eve is located on a network segment that is passing traffic between Alice and Bob. However, in order that other connections on the network are not affected by Eve's antics, Eve may need to use Hunt's ARP relay daemon to relay the data for some of these connections. Under certain circumstances, for example if Eve is located on an intermediate segment and has spoofed the MAC addresses of routers, the volume of relay traffic could be overwhelming and cause the attack to be disruptive to other systems.

Using Hunt in a switched Ethernet LAN environment

Recall that one reason to use switched Ethernet is to segment a network in order to increase performance. If we move our hijack scenario to an environment that uses switched Ethernet LANs, Eve will have to work a little bit harder in order to hijack sessions. If she simply sets her Ethernet interface to promiscuous mode, she will not see any connections because there are no other devices on her network segment. However, Eve can use Hunt's MAC discovery daemon to determine which other systems are connected to the switch and then use the ARP spoofing capabilities to dupe the switch and the devices connected to it. Once Eve can see the connections, the steps performed by her are the same as those described in the previous hijack scenario. However, some explanation is required about how Hunt is able to dupe the switch and other devices.

Duping the switch. An Ethernet switch maintains a table that maps MAC addresses to a port on the switch. The switch constructs the table by learning source MAC addresses from traffic that originates from systems connected to its ports. Given that it is legitimate to have multiple MAC addresses mapped to a single port (for example, although network performance will suffer, it is legitimate to connect a hub to a port on a switch), Hunt's ARP spoofing causes the switch to add a mapping between a non-existent MAC address and the port to which Eve is connected.

Duping other devices. By forging ARP replies that contain non-existent MAC addresses and sending them to systems connected to the switch, Eve can cause some of the traffic from those systems to be directed to her port on the switch. So, when Alice sends data to Bob, for example, the data will be addressed to a non-existent MAC address. However, the switch thinks that this non-existent MAC address is on the same port as Eve and will send the data to Eve's port. If Eve can see connections that have active sessions, then she can hijack them with Hunt.

Tightening Ethernet security

Some Ethernet switches provide mechanisms that enable a security policy to be implemented to prevent unauthorized access to the network. As a starting point, it is good practice to disable unused ports on a switch in order to prevent an attacker gaining physical access to a network. If an attacker is somehow able to plug his or her machine into a switch port, this addition to your security policy will prevent the attacker from being able to connect to the network. More often that not, though, an attacker will gain access to a system that is already connected to the network, which means that the port will already be enabled.

Switches may also have security features that allow a network administrator to associate an individual MAC address or a group of valid MAC addresses to a port so that only those addresses are allowed to send data through that port. If a device attempts to send data through a port using a source MAC address that does not appear in the 'access list', the switch will flag it as an address violation, block the transmission attempt, and possibly raise an exception.

Even with the above security policy implemented on a switch, it is still possible to perform session hijacking by modifying Hunt to send out ARP replies that map the IP addresses of the target (Bob) and trusted (Alice) systems to the MAC address of its own system (Eve). This modified approach to ARP spoofing will cause traffic between the affected systems to be sent to Eve, but will not generate an address violation in the switch. Eve will act as a relay for the connections and can intrude upon them at her leisure.

Turning our attention away from Ethernet switches and towards the systems that are connected to them, it is possible to create static ARP tables on some systems. The steps required to do this will vary from system to system, but the basic idea is that the system needs to be configured to ignore ARP information from the network and only use a static table that has been configured by the system administrator. Although this is a manually intensive process, it is an effective way to prevent systems becoming victims of ARP spoofing.

Conclusion

From the above we can conclude that it is necessary to consider a multi-layered approach to security.

Preventing physical access to our network infrastructure components is a good starting point. Another positive step towards securing Ethernet LANs is to install Ethernet switches, especially on critical segments such as DMZs and server farms. Although this may prevent some less sophisticated attackers from sniffing passwords or hijacking sessions, security policy should be implemented on the switches to protect against unauthorized connection to the network and more sophisticated attacks that involve the type of ARP spoofing that is currently implemented by Hunt.

ARP spoofing also exploits a weakness in the design of the Address Resolution Protocol itself. With this in mind, a static ARP table could be implemented on devices that support it.

IP spoofing exploits weaknesses in the design of TCP/IP. In view of this, ingress and egress filters can be used on the routing devices in order to prevent some spoofed packets from being routed between networks. In addition, use TCP/IP software that generates sequence numbers that are hard to predict.

Session hijacking can compromise connections even if the sessions have been established using strong authentication and sniffers can reveal passwords if they travel over the network in plain text. When communicating in the presence of adversaries, encryption can be used to secure remote management sessions to critical components of security infrastructure, such as firewalls, routers and PKI systems. Secure Shell (SSH), for example, can provide us with strong authentication *and* encrypted connections.

Factors, such as time, money, resources and operating environment, may influence your final implementation, but awareness of security weaknesses and knowledge of the countermeasures that are available, will certainly increase your chance of keeping attackers at bay.

References

Bellovin, Steve. "Security Problems in the TCP/IP Protocol Suite". Reprinted from Computer Communications Review 2:19, pp. 32-48, April 1989. URL: http://www.research.att.com/~smb/papers/ipext.pdf (20 Jun. 2000).

Brenton, Chris. "What is Egress Filtering and How Can I Implement It?". Egress Filtering v 0.2. 29 February 2000. URL: http://www.sans.org/infosecFAQ/egress.htm (20 Jun. 2000).

daemon9 (pseud.) / route (pseudo.) / infinity (pseudo.). "IP-spoofing Demystified (Trust-Relationship Exploitation)". Phrack Magazine Volume Seven, Issue Forty-Eight. URL: http://phrack.infonexus.com/search.phtml?view&article=p48-14 (20 Jun. 2000).

Farmer, Dan / Venema, Wietse. "Improving the Security of Your Site by Breaking Into it". URL: http://www.alw.nih.gov/Security/Docs/admin-guide-to-cracking.101.html (20 Jun. 2000).

Fuller, Edward. "Denial of Service Attack". 6 April 2000. URL: http://www.sans.org/infosecFAQ/dos.htm (20 Jun. 2000).

Joncheray, Laurent. "Simple Active Attack Against TCP". URL: http://www.insecure.org/stf/iphijack.txt (20 Jun. 2000).

Plummer, David. "An Ethernet Address Resolution Protocol". RFC 826. November 1982. URL: http://www.faqs.org/rfcs/rfc826.html (20 Jun. 2000).

Postel, J. "Multi-LAN Address Resolution". RFC925. October 1984. URL: http://www.faqs.org/rfcs/rfc925.html (20 Jun. 2000).

Volobuev, Yuri. "Playing redir games with ARP and ICMP". 19 September 1997. URL: http://www.insecure.org/sploits/arp.games.html (20 Jun. 2000).

"Ethernet Technologies". Internetworking Technology Overview. URL: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ethernet.htm (20 Jun. 2000).

"LAN Switching". Internetworking Technology Overview. URL: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/lanswtch.htm (20 Jun. 2000).

"The Kevin Mitnick/Tsutomu Shimomura affair". Random Access WWW Journal. Friday, January 21, 2000. URL: http://www.gulker.com/ra/hack/index.html (20 Jun. 2000).

"When and how should I use static ARP?". The Solaris Security FAQ. URL: http://www.sunworld.com/common/f_security-faq.html#Q2.14 (20 Jun. 2000).

Pavel Krauz's Home Page. URL: http://www.cri.cz/kra/index.html. (20 Jun. 2000).