Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Packet Sniffing In a Switched Environment

This paper focuses on the threat of packet sniffing in a switched environment, and briefly explores the effect in a non-switched environment. Detail is given on techniques such as "ARP (Address Resolution Protocol) spoofing", which can allow an attacker to eavesdrop on network traffic in a switched environment. Third party tools exist which permit sniffing on a switched network. The result of running some of these tools on an isolated, switched network is presented; it clearly demonstrates that the threat they pose is ...

Copyright SANS Institute
Author Retains Full Rights

AD

Packet Sniffing In a Switched Environment

Tom King
GSEC Practical v1.4, Option 1
August 4th 2002, updated June/July 2006.

Preface

In this paper you'll learn about packet sniffing and how to better protect your networks from potential threats. If you'd like more training on network security, we recommend taking the SANS SEC502 Firewalls, Perimeter Protection and VPNs course, available both online and via live classroom training.

Abstract

This paper focuses on the threat of packet sniffing in a switched environment, and briefly explores the effect in a non-switched environment. Detail is given on a number of techniques, such as "ARP (Address Resolution Protocol) spoofing", which can allow an attacker to eavesdrop on network traffic in a switched environment.

Third party tools exist that permit sniffing on a switched network. The result of running some of these tools on an isolated, switched network is presented, and clearly demonstrates that the threat they pose is real and significant.

The final section covers ways to mitigate the threat of network sniffing in both non-switched and switched environments. The thesis of this paper is that encryption is the only true defence to the threat of sniffing.

A note about permission

A number of techniques and tools to enable network sniffing are detailed in this paper. Tests have been run on an isolated network, constructed especially for this piece of work.

If you want to use any of the tools or techniques listed in this paper on a network at your college or place of work, you should seek permission in writing from appropriate management. It would also be prudent to talk to the network team at your site. It is possible to severely disrupt a network through the inappropriate use of some of the tools described here.

Introduction

For most organizations, packet sniffing is largely an internal threat. A third party on the Internet, for instance, could not easily use packet sniffing software to eavesdrop on traffic on a corporate LAN. But since the greatest threat to corporate systems frequently is internal[1], we should not take comfort from this.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

There are many reasons that businesses are updating their network infrastructure, replacing aging hubs with new switches. A frequently stated driver for moving to a switched environment is that "it increases security". However, the thinking behind this is somewhat flawed. Packet sniffing in a switched environment *is* possible -- anyone equipped with a laptop (and armed with a selection of freely available software) may be able to monitor communication between machines on a switched network.

Packet sniffing tools have been available from the early days of networked computing environments. The tools are powerful software, which facilitate trouble-shooting for network administrators. However, in the hands of a malicious third party, they are a devastating hacking tool, which can be used to glean passwords and other sensitive information from a LAN.

Traditionally, packet sniffers have been regarded as fairly obscure tools that require a certain technical competence to operate -- dangerous utilities, perhaps, but not easy to guide or operate. All this has changed in the last few years, with specialized, easy to use password-detecting sniffers becoming widely obtainable. Many of these "new generation", specially tailored tools are freely available on the Internet. With built-in logic allowing many network protocols to be decoded, they have the capability to filter the sniffed traffic on the fly, and highlight sensitive information such as usernames and passwords.

Packet sniffing in a non-switched environment is a well understood technology. A large number of commercial and non-commercial tools enable eavesdropping of network traffic. The idea is that to eavesdrop on network traffic, a computer's network card is put into a special "promiscuous" mode. Once in this mode, all network traffic (irrespective of its destination) that reaches the network card can be accessed by an application (such as a packet sniffing program). A detailed explanation of how packet sniffing works may be found in Robert Graham's excellent FAQ on sniffing[2].

In a switched environment, it is more of a challenge to eavesdrop on network traffic. This is because usually switches will only send network traffic to the machine that it is destined for[3]. However, there are a number of techniques that enable this functionality to be usurped. Tools exist that combine the ability of sniffing on a switched network with the capability of filtering the traffic to highlight sensitive information.

Packet Sniffing in a non-switched environment

In a non-switched environment, the latest generation of packet sniffing tools is highly effective at reaping passwords and other sensitive information from the network.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

A large number of commonly used protocols either transmit data in plaintext (which can easily be sniffed), or they do not use strong enough encryption to prevent a sniffing and cracking attack. Examples of plaintext protocols include smtp, pop3, snmp, ftp, telnet and http. Perhaps the best known encrypted protocol that is vulnerable to sniffing and cracking attacks is Microsoft's LM (LAN Manager) protocol, used for authenticating Windows clients.

Microsoft has tried to address the glaring weaknesses in LM, with the introduction of NTLM (V1 and V2). NTLM is an improvement, but is still susceptible to a sniffing and cracking attack. Hidenobu Seki, the author of ScoopLM and BeatLM tools (qv) gave a fascinating presentation[4] covering the detail of LM, NTLM v1 and v2 and how it can be cracked at BlackHat's "Windows Security 2002 Briefings and Training".

Since the first draft of this paper, Kerberos has become widely used as the authentication protocol of choice in modern Windows environments (Windows XP clients, Windows 2003 servers). The move from LANMAN/ NTLM to Kerberos was widely thought to cure the problem of sniffing (then cracking) Microsoft passwords[5]. This is *not* the case, however. Tools such as KerbCrack[6] enable cracking of Kerberos logins.

Tools to sniff in a non-switched environment

A quick search on the Internet will reveal a large number of freely available sniffing tools. In this section, I focus on two tools, `dsniff` and `ScoopLM`, which excel at sniffing sensitive information.

`dsniff`

For plaintext protocols, to eavesdrop on username, password and other sensitive information, a very useful tool is `dsniff` from Dug Song[7]. The `dsniff` tool is available for various flavours of Unix, and there is a port (of an older version of the software) for Windows[8].

In addition to sniffing the plaintext protocols mentioned above (and others), dsniff is exceptionally good at filtering the sniffed traffic to display only "interesting" information such as usernames and passwords. In their esteemed ***Hacking Exposed*** book[9], McClure, Scambray and Kurtz describe `dsniff` as offering "passwords on a silver platter". It makes eavesdropping on sensitive information a trivial exercise.

A sample run of `dsniff` is depicted in Figure 1, showing the Windows port of dsniff harvesting passwords on a small network.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

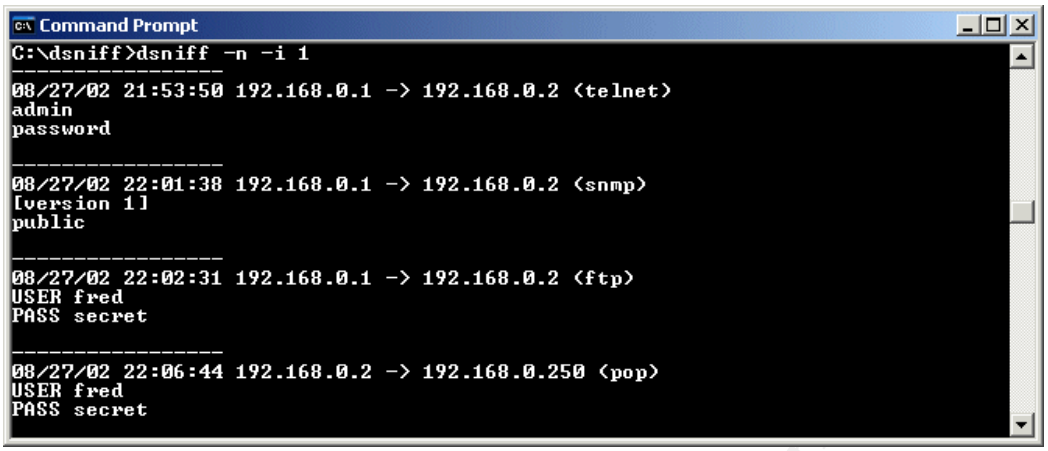

Figure 1 - dsniff sniffing plaintext protocols in a non-switched environment

ScoopLM

L0phtcrack is a well-known password sniffing and cracking tool, which is capable of eavesdropping Windows NT/ 2000 usernames and encrypted passwords from a network. It is a commercial tool, available from @Stake[10]. However, there are other freely available tools that can perform a similar job, and are very simple to use.

A great example is the ScoopLM tool[11], which is freeware and downloadable from the Internet. ScoopLM will sniff NT/ 2000 usernames and LM/ NTLM encrypted passwords. Its brother, BeatLM[12], enables cracking of encrypted passwords that ScoopLM has harvested by brute-force or dictionary attacks. Together, they are a significant threat to the security of Microsoft networking in a non-switched environment.

Figure 2 shows a sample run of ScoopLM, sniffing NT usernames and encrypted passwords. The sniffed usernames and passwords can then be saved to a temporary file, and loaded into BeatLM to be cracked.

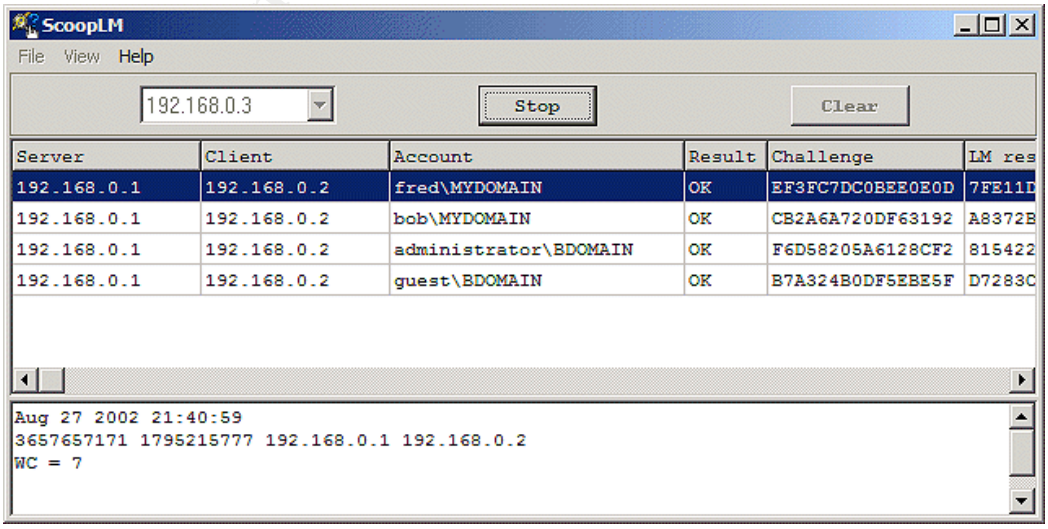

Figure 2 - ScoopLM in action, sniffing NT usernames and encrypted passwords

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

The above examples demonstrate how simple it is to discover sensitive information by eavesdropping on a non-switched network. This fact has helped drive businesses to replace hubs in their network by switches. There are many other good reasons for doing this -- increasing network performance, for example. Replacing hubs by switches in the belief that it will cure the problem of sniffing is misguided. The following section will demonstrate why.

Packet Sniffing in a switched environment

Switches

On the surface, it would seem that replacing hubs by switches will mitigate the packet sniffing threat to a large extent. The fact that switches will only send network traffic to the machine that it is destined for implies that if machine A is communicating with machine B, machine C will not be able to eavesdrop on their conversation. In Figure 3, let us assume that machine A instigates a telnet connection to machine B.

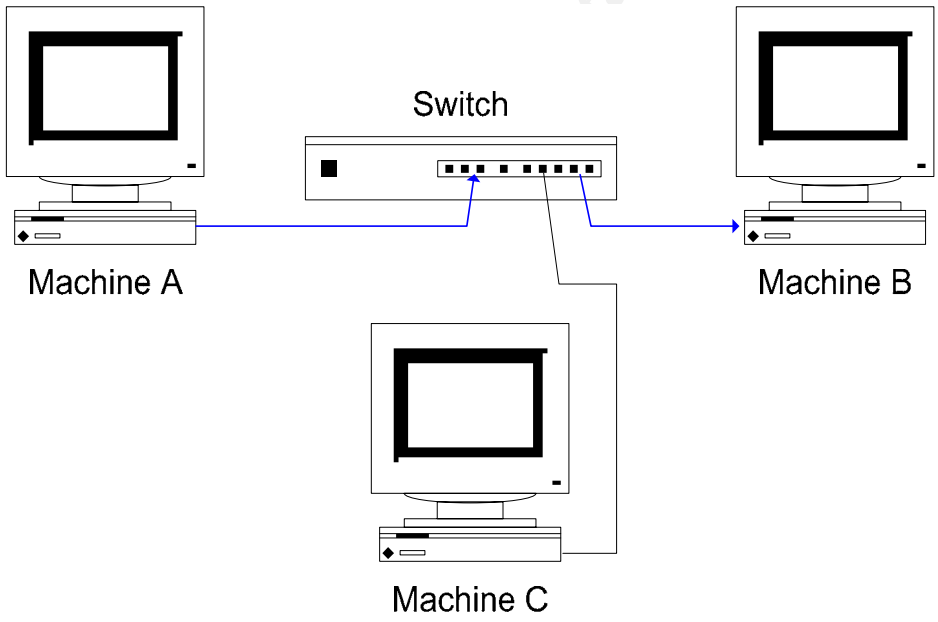

Figure 3 - Three machines connected via a switch. Traffic flowing from A to B is illustrated by the arrowed lines.

In the situation depicted above, Machine C cannot easily see the network traffic for the telnet session passing between machines A and B. The switch ensures that this traffic does not travel over any unnecessary ports - it only flows over the ports that machines A and B are connected to.

However, a number of techniques exist that will subvert the above, enabling C to snoop on the network traffic between A and B.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

How to sniff in a switched environment

It's all about the "man-in-the-middle"[13]. Sniffing traffic in a switched environment is achieved by setting up a "man-in-the-middle" attack. The attacker uses a variety of techniques to force network traffic to/ from the victim to go to the attacker's machine. When this occurs, the attacker can inspect (or even modify) the victim's network traffic.

There are a large number of techniques that permit sniffing in a switched environment. Common techniques include ARP spoofing, MAC flooding, MAC duplicating, ICMP redirection, DHCP spoofing and port stealing.

The tools covered in this paper all use the ARP spoofing technique, so this is covered in detail. An excellent description of ARP spoofing, MAC flooding and other techniques can be found in Sean Whalen's paper on the Packet Storm website[14]. Another useful resource describing popular approaches to switched network sniffing is the main page for the sniffing tool, ettercap[15].

ARP spoofing is a reasonably straightforward technique, a classic man-in-the-middle[16] attack. This is best explained by an example. Taking the above example of machines A, B, and C, assume C wanted to eavesdrop on network traffic between A and B. For a man-in-the-middle attack, C pretends to A that it is in fact B. Then, when A sends traffic destined for B, it is intercepted by C. C passes this information on to B, pretending that it came from A. Similarly, C also performs a comparable role for traffic from B, which is destined for A. The goal of the man-in-the-middle attack is depicted in Figure 4.

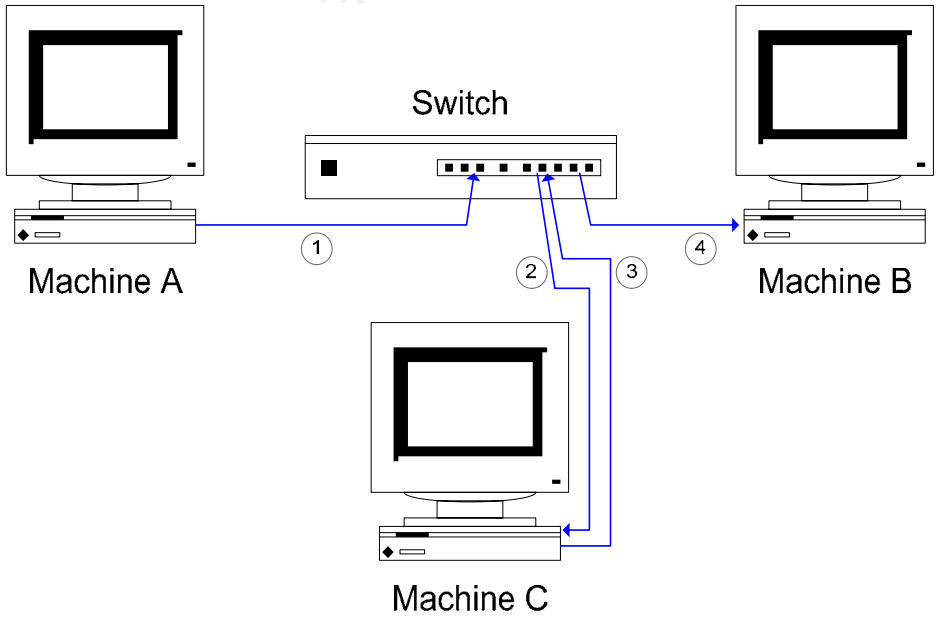

Figure 4 – The man in the middle attack. C intercepts network traffic from A which is destined for B.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

In more detail, using ARP spoofing to complete the man-in-the-middle-attack, two steps, detailed below, need to be performed.

First, however, we need to understand how A and B will normally communicate.

For this to happen, A requires B's MAC address. To get this, A will check in its ARP cache to see if it already has B's MAC address.

> If this is the case, it will use the MAC address pulled from the ARP cache.
>
> If this is *not* the case, A will broadcast an ARP request. B will respond with its MAC (and IP) address. B's IP address and corresponding MAC address will be stored in A's ARP cache, for future use.

A can now send packets of data to B. For B to communicate with A, a similar process will take place.

Let us now assume that A and B have established each other's MAC addresses, and are communicating through a switch. How can C eavesdrop on the conversation? This is where ARP spoofing comes into play.

1. The first step is for C to pretend to A that it is in fact B. If this can be achieved, network traffic destined for B will be routed to C. Likewise, C must pretend to B that it is in fact A. How can this be achieved? The answer is that C "poisons" the ARP cache on A and B. This is straightforward, because:

   "ARP is a stateless protocol that does not require authentication, so a simple ARP replay packet sent to each host will force an update in their ARP cache"[17]

   So, C sends a spoofed ARP packet to A, instructing A to send packets destined for B to C. The spoofed ARP packet C sends forces A to update its own ARP cache. In A's updated ARP cache, B's IP address maps to C's MAC address. This means future communication from A which is destined for B will go via C.

   The following tables show what happens to A's ARP cache.

Machine A's ARP Cache – *before* C sends spoofed ARP packet

IP Addresses	MAC Addresses
[B's IP Address]	[B's MAC Address]
[C's IP Address]	[C's MAC Address]
…	…

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Machine A's ARP Cache – *after* C sends spoofed ARP packet

IP Addresses	MAC Addresses
[B's IP Address]	[C's MAC Address]
[C's IP Address]	[C's MAC Address]
…	...

C also does something similar to B. It sends a spoofed ARP packet to B, instructing B to update its ARP cache so that A's IP address maps to C's MAC address.

Once this has been done, packets that A attempts to send to B are routed to C. Packets that B attempts to send to A are routed to C as well.

2. There is one more important step. Machine C also has to ensure that traffic it receives is sent on to its true destination. So, for example, when A sends traffic destined for B, it is intercepted by C, but sent on from C to B. This can easily be achieved by IP forwarding, a facility supported by many operating systems. Alternatively, an application can take responsibility for forwarding the traffic to its true destination.

Once the above steps have been performed, C will be intercepting network traffic between A and B.

"Re-poisoning" the ARP Cache

It is worth noting that once a spoofed ARP packet has been sent to a target machine, the attacker will need to re-send this information on a regular basis, to "re-poison" the ARP cache. This is because operating systems automatically refresh ARP caches on a frequent basis (every 30 seconds is a typical refresh rate).

Susceptibility of Operating Systems to ARP poisoning

As of 2006, most modern operating systems (including Windows XP SP2) are still susceptible to this attack. Although Solaris was viewed as being resistant by some[18], this is not the case -- ettercap has techniques that allow the ARP cache on a Solaris machine to be subverted[19].

"Port security" and ARP spoofing

Many switches now offer a configurable "port security" option, to help network administrators lock down which machines can connect to switches. Put simply, "port security" allows us to lock down a port on a switch to a given MAC address. This helps prevent un-trusted machines connecting to the switch.

However, there is significant administration overhead to widely deploy and support "port security" on anything more than a very small network.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Further, "port security" does not prevent ARP spoofing[20]. With ARP spoofing, we are just poisoning the ARP cache on target machines (in the above example, machines A and B); this is not something that "port security" on a switch prevents.

Session hijacking – made possible by ARP spoofing

An interesting side-effect is made possible through eavesdropping by ARP spoofing/ IP forwarding. Because we are performing a man-in-the-middle attack, we can alter (add, modify or delete) packets we intercept, or even create brand new packets.

This enables us to hijack certain types of sessions, telnet, for example. In addition to sniffing the telnet traffic, we can forge commands made by the client, or replies made by the server. This enables all sorts of nefarious activities – how about forging a "`mail hacker@hack.com </etc/passwd`" command, from the client, for instance?

Session hijacking is not just a theoretical possibility. Tools such as `ettercap`[21] and `hunt`[22] make it simple to achieve.

Since the original draft of this paper, a number of tools (including `ettercap` and `Cain`) have built on the session hijacking idea to offer an attack against SSL data streams. This can be used to intercept highly sensitive data in transit to https sites.

Wireless networks

Since the first draft of this paper, wireless networks have gone mainstream, and are now found in many businesses and home setups.

Many wireless networks -- especially public hotspots -- have no security at all. On such networks, packet sniffing via man-in-the-middle techniques can be very powerful. Any sensitive information (such as usernames and passwords) that is not using secure protocols can be discovered trivially. Further, attacks against secure protocols such as SSL undermine commonly held dogmas that browsing to https sites (even on untrusted networks) is safe.

Tools to sniff in a switched environment

The number of tools that enable sniffing in a switched environment is on the increase. In this section, I focus on two tools in particular, `ettercap` and `Cain`. Both tools excel at sniffing sensitive information on a switched network.

Setup of isolated network

An isolated network was set up to investigate sniffing in a switched environment. Three machines (A, B and C) were set up, following the example detailed above. As above, A and B are the victim machines and C is the attacking machine, which runs the sniffing software. The following table summarizes the setup of the machines on the isolated network.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Machine Name	IP Address	MAC Address
A	192.168.0.1	00-02-e3-0a-ee-e4
B	192.168.0.2	00-50-22-88-f1-48
C	192.168.0.3	00-00-39-ca-13-81

All machines were setup to run Windows 2000 Professional SP2. The switch used in the isolated network was a simple 5 port 10/100Mb switch, manufactured by Unex Innovation Corp.

ettercap

First, we cover `ettercap`, a tool that describes itself as "a powerful and flexible tool for man-in-the-middle attacks". It runs on many of the leading platforms including Windows, Linux, xBSD and Mac OS X.

`ettercap` was downloaded from http://ettercap.sourceforge.net/download.php then installed on machine C. Before running `ettercap`, the ARP cache on machines A and B were checked, via the `arp /a` command. As expected, the ARP cache on A was storing the true IP and MAC addresses of B and C:

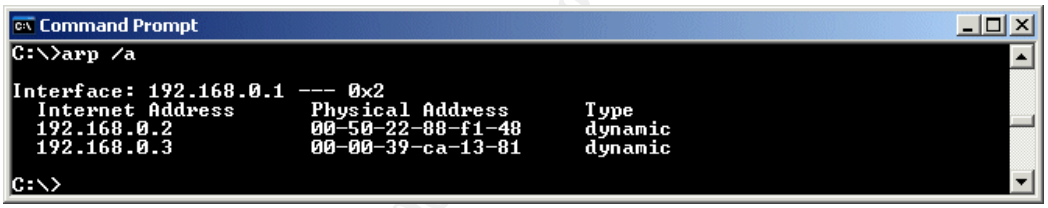

Figure 5 - the ARP cache on machine A prior to running ettercap

Similarly, the ARP cache on B was storing the true IP and MAC addresses of A and C.

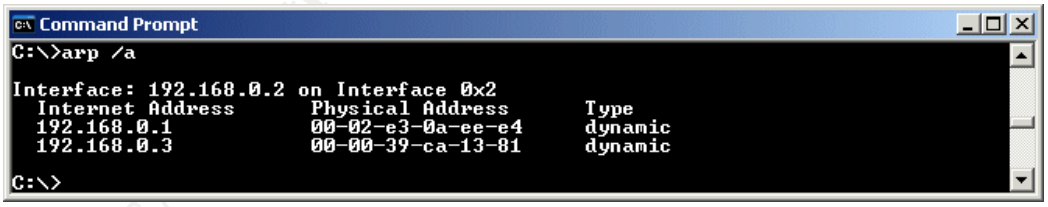

Figure 6 - the ARP cache on machine B prior to running ettercap

Next, `ettercap` was run on machine C, and set to sniff traffic between A and B. At this stage, `ettercap` performs ARP spoofing to set up the man-in-the-middle attack. Re-examining the ARP caches on A and B is illuminating: note how machine C's MAC address replaces the true MAC addresses for machines A and B:

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

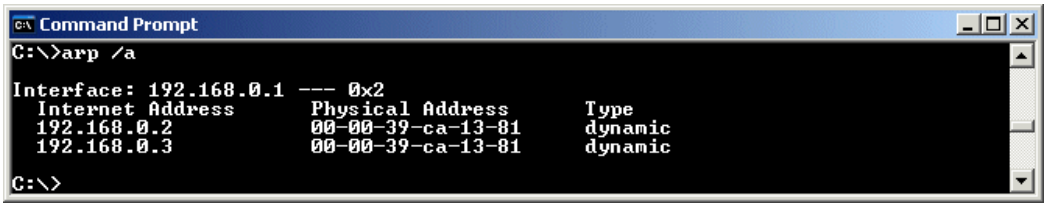

Figure 7 - the ARP cache on machine A now ettercap is running

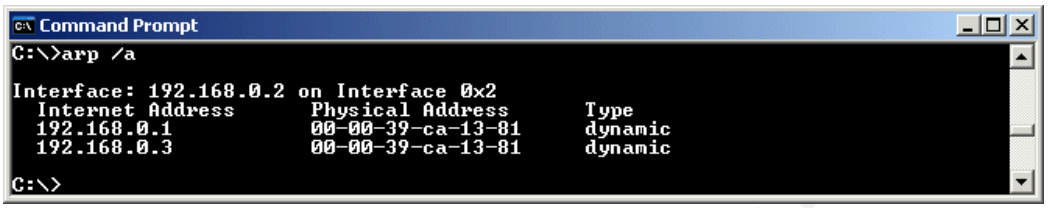

Figure 8 - the ARP cache on machine B now ettercap is running

Now traffic between A and B was being intercepted by C. Similar to `dsniff`, `ettercap` has in-built knowledge of a large number of network protocols. It can highlight interesting areas of sniffed traffic, such as usernames and passwords. The following diagram depicts `ettercap` eavesdropping the start of a telnet session between A and B:

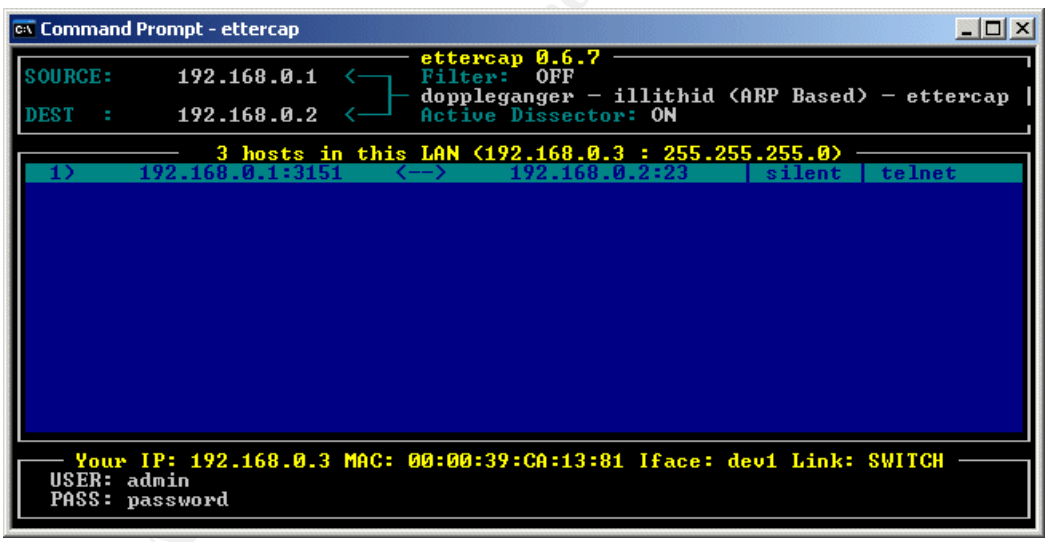

Figure 9 - ettercap sniffing a telnet session between A and B

During a sniffing session, `ettercap` may detect a large number of usernames and passwords. The data may be saved to a simple ASCII file for examination at a later date.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Cain

Another tool that is capable of sniffing in a switched environment is Cain[23]. Available for Windows only, this tool can do far more than just sniff traffic on a switched network.

In a similar vein to dsniff and ettercap, Cain has built-in knowledge of various network protocols, and can highlight interesting areas of sniffed traffic.

Cain also has built in cracking technology to enable brute-force and dictionary attacks against encrypted passwords that it sniffs from the network. In a similar manner to BeatLM, Cain can attempt attacks against Microsoft's authentication protocols (including LM, NTLMv1, NTLMv2). However, it goes further than BeatLM by offering the facility of cracking Cisco MD5 hashes, encrypted APOP passwords and others.

Highlights of other facilities built in to Cain include various networking utilities (including traceroute and tools to analyze routing protocols), and the capability of enumerating NT users and shares from remote machines.

The breadth of functionality covered by Cain is impressive. It is amazing that a single tool can cover most of the key roles offered by better known sniffing/ enumeration/ password cracking tools such as L0phtcrack, Revelation[24], userdump[25], pwltool[26], john the ripper[27] and ettercap.

Cain was downloaded from http://www.oxid.it, and installed onto machine C. The ARP caches on machines A and B were checked, and found to contain the expected data (as in Figures 5 and 6). Next, Cain was configured to use ARP spoofing – referred to as APR (ARP poisoned routing) within the application – to intercept network traffic between machines A and B. This is depicted in Figure 10:

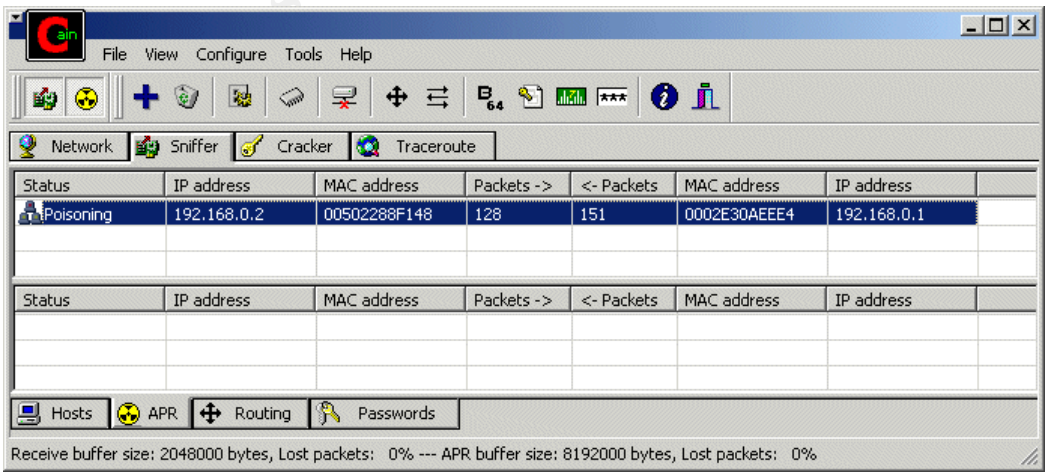

Figure 10 - Cain uses ARP spoofing to intercept data between machines A and B

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Once this had been done, `Cain` used its built-in knowledge of network protocols to enable key data to be displayed. As with the test with `ettercap`, a telnet session between machines A and B was initiated. For many protocols, `Cain` simply captures the username and password. For telnet sessions, the *entire* session (including the username and password) is captured and logged to a text file, as shown in Figure 11:

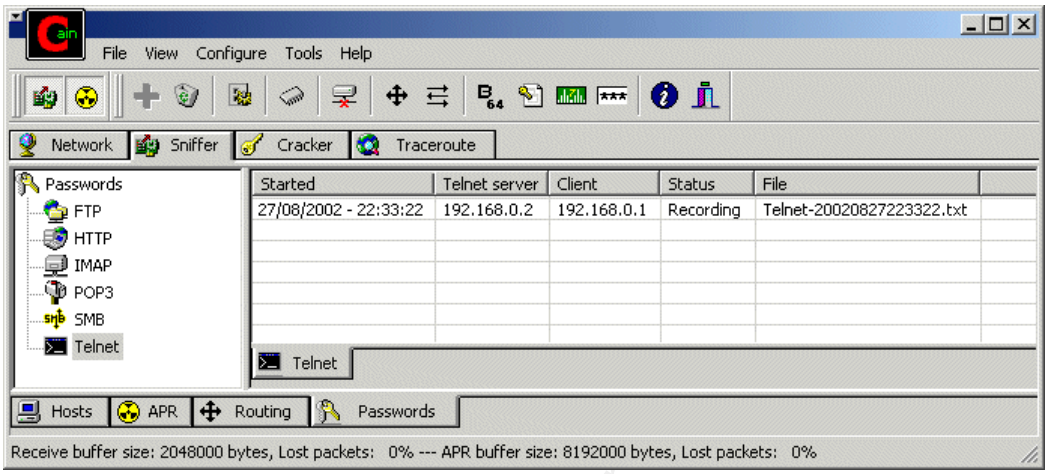

Figure 11 - Cain recording a telnet session between two machines

The above tests demonstrate that tools such as `ettercap` and `Cain` present a very real threat to many network environments. What can be done to protect against this threat?

Recommendations for mitigating the threat from packet sniffing

Detecting packet sniffers

One way to mitigate against the threat of packet sniffing tools is to try to detect if they are used on the network.

a) Detecting in a non-switched environment

Detecting tools designed to run in a non-switched environment is difficult. This is because the tools are usually "passive". They work by putting the network interface card into promiscuous mode, allowing any network traffic that reaches the card to be examined. Akin to a radio receiver, sniffers do not necessarily cause extra, suspicious traffic to be transmitted on the network, so how can they be discovered?

A number of techniques can be used to try to detect machines whose network cards are running in promiscuous mode, and likely to be sniffing traffic. Many of the techniques used rely on detecting specific weaknesses in TCP/IP stacks. L0pht's antisniff[28] employs knowledge of the idiosyncrasies of TCP/IP stacks in NT and Unix to detect machines running in promiscuous mode.

b) Detecting in a switched environment

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

As indicated previously, sniffing in a switched environment implies a man-in-the-middle attack. Eavesdropping in this case will be "active" in that network traffic will be delivered to the attacking machine, then forwarded onto the true recipient. Detecting this is somewhat easier than detecting the "passive" tools.

It is possible to detect techniques such as ARP spoofing – software such as LBNL's `arpwatch`[29] can detect suspicious ARP network traffic, and inform a network administrator.

Newer tools, such as Security Friday's `promiscan`[30] use multiple techniques to detect sniffers.

Ultimately, however, software cannot be relied upon to reliably detect all instances of network sniffing with one hundred percent accuracy.

Locking down the network environment

Imagine it were possible to prevent network sniffing software being installed on any machine on the network. Is this possible?

Solutions such as Microsoft's Software Restriction Policies and AppSense[31] can help to ensure that only approved software is run -- packet sniffing tools and other hacking tools could be prevented from executing. However AppSense is not relevant in all environments since it only supports Microsoft Windows. Furthermore, AppSense cannot prevent unauthorized machines (for instance a rogue laptop running eavesdropping software) from connecting to the network.

Another approach to securing the network environment against sniffing is to make extensive use of "virtual LANs" (vlans) to segment a network into logical segments. Sniffing across vlans is not easily achieved.

Encryption

The most viable solution to protect against packet sniffing is encryption.

In the FAQ[32] for `dsniff`, Dug Song advises as follows: "Don't allow proprietary, insecure application protocols or legacy cleartext protocols on your network". This is valuable advice. Substituting insecure protocols (such as telnet) with their secure, encrypted counterparts (such as ssh) presents a significant barrier to eavesdropping. Replacing all insecure protocols, however, is unlikely to be feasible in many environments.

Instead of halting the use of cleartext protocols, one possibility is to encrypt all network traffic by using IPSec[33]. By encrypting using IPSec, it is possible to continue to use plaintext protocols -- all data is encapsulated by IPSec, and is encrypted for its transfer across the network. Thus legacy applications that may rely on using older, plaintext protocols will be unaffected.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

IPSec is completely transparent to applications and to users. It is an open standard, supported by many vendors, including Microsoft and Cisco. Furthermore, many Unix implementations support IPSec. The easy configurability of IPSec within Windows 2000 and XP further increases its accessibility.

Implementation of a layer three encryption technology such as IPSec solves the sniffing problem completely. The scalability, widespread availability and seamless operation of IPSec highlight it as a pragmatic solution to the problem of network eavesdropping.

This paper covered packet sniffing and how to better protect your networks from potential threats. If you'd like more training on network security, we recommend taking the SANS SEC502 Firewalls, Perimeter Protection and VPNs course, available both online and via live classroom training.

© SANS Institute 2007, Author retains full rights.

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

References

[1] Verton, Dan. "Analysts: Insiders may pose security threat". 10 October 2001. URL: http://www.computerworld.com/securitytopics/security/story/0,10801,64774,00.html

[2] Graham, Robert. "Sniffing (network wiretap, sniffer) FAQ". Version 0.3.3. 14 September 2000. URL: http://www.robertgraham.com/pubs/sniffing-faq.html. Note: the "official" URL no longer works as of June 2006. However, a copy of this useful FAQ can be found at http://www.windowsecurity.com/whitepapers/Sniffing_network_wiretap_sniffer_FAQ_.html

[3] Tyson, Jeff. "How LAN Switches Work". URL: http://www.howstuffworks.com/lan-switch.htm

[4] Seki, Hidenobu. "Cracking NTLMv2 Authentication". URL: http://www.blackhat.com/presentations/win-usa-02/urity-winsec02.ppt

[5] "The feasibility of attacking Windows 2000 Kerberos passwords", March 2002. URL: http://www.securiteam.com/windowsntfocus/5BP0H0A6KM.html

[6] Vidstrom, Arne. "Kerbcrack", URL: http://www.ntsecurity.nu/toolbox/kerbcrack/

[7] Song, Dug. "dsniff". URL: http://monkey.org/~dugsong/dsniff

[8] Davis, Michael. "dsniff". URL: http://www.datanerds.net/~mike/dsniff.html.

[9] McClure, Stuart. Scambray, Joel. Kurtz, George. "Hacking Exposed (Third Edition)". McGraw-Hill, 2001. pages 464-465.

[10] "L0phtcrack 4". http://www.atstake.com/research/lc/index.html. Note since @stake was taken over by Symantec, this URL no longer functions.

[11] Seki, Hidenobu. "ScoopLM". January 2002. URL: http://www.securityfriday.com/ToolDownload/ScoopLM/scooplm_doc.html

[12] Seki, Hidenobu. "ScoopLM". February 2002. URL: http://www.securityfriday.com/ToolDownload/BeatLM/beatlm_doc.html

[13] "Man-in-the-middle attack". URL: http://en.wikipedia.org/wiki/Man_in_the_middle

[14] Whalen, Sean. "An Introduction to ARP Spoofing". Revision 1. April 2001. URL: http://www.packetstormsecurity.org/papers/protocols/intro_to_arp_spoofing.pdf

[15] Ettercap's homepage. URL: http://ettercap.sourceforge.net

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

[16] Cohen, Fred. "The All.Net Security Database". May 1999. URL: http://www.all.net/CID/Attack/Attack74.html

[17] Montoro, Massimiliano. "Introduction to ARP Poison Routing (APR)". Revision 1.0. URL: http://www.oxid.it

[18] Beeky, Mike. "ARP Vulnerabilities: Indefensible Local Network Attacks?" 2001. URL: http://www.blackhat.com/presentations/bh-usa-01/MikeBeekey/bh-usa-01-Mike-Beekey.ppt

[19] "Iverson". "ARP Poisoning HowTo". URL: http://ettercap.sourceforge.net/forum/viewtopic.php?t=2392

[20] "Ettercap effects on switches". URL: http://ettercap.sourceforge.net/forum/viewtopic.php?t=2

[21] Ettercap's homepage. URL: http://ettercap.sourceforge.net

[22] Krauz, Pavel. "Hunt Project". Original URL no longer works: http://lin.fsid.cvut.cz/~kra/index.html#HUNT. However, Hunt can be found at http://www.packetstormsecurity.org/sniffers/hunt/

[23] Montoro, Massimiliano. "Homepage for Cain". URL: http://www.oxid.it

[24] "Snadboy Software". URL: http://www.snadboy.com

[25] "Hammer of God Utilities". URL: http://www.hammerofgod.com/download.htm

[26] "Lastbit Software". URL: http://lastbit.com/vitas/pwltool.asp

[27] "John the Ripper password cracker". URL: http://www.openwall.com/john

[28] "Antisniff 1.021" URL: http://www.packetstormsecurity.org/sniffers/antisniff

[29] "arpwatch" URL: ftp://ftp.ee.lbl.gov/arpwatch.tar.gz

[30] "Promiscan". URL: http://www.securityfriday.com/products/promiscan.html

[31] "AppSense Application Manager". URL: http://www.appsense.net/content/products/products.asp#amde

[32] Song, Dug. "dsniff Frequently Asked Questions". URL: http://monkey.org/~dugsong/dsniff/faq.html

[33] "IPSec". URL: http://en.wikipedia.org/wiki/Ipsec

© SANS Institute 2007, As part of the Information Security Reading Room Author retains full rights.

Last Updated: May 23rd, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced