Abstract

Using Passive Network Discovery to Fingerprint Vulnerabilities within Ethernet Broadcast Frames.

This paper examines how open source embedded network tools were used to perform persistent internal audits of Ethernet Local Area Network broadcast traffic. The initial requirements to define the project phases were developed based on the analysis of each open source learning stage. Open Source UNIX version, Ubuntu, was selected as the platform to prototype because of its ease of use and usable business productivity, internet, drawing and graphics applications. To understand why hosts within the ONR LAB were experiencing a decrease in system performance and transmission speed. A Passive Network Discovery of Ethernet Broadcast Frames was captured and analyzed to determine if Local Area Network traffic between the local and foreign hosts is malicious or valid. The identification of remote active nodes and their system information was collected to build a resource map of all remote hosts requesting services from hosts within the ONR Lab and listing of local hosts listening ports and services running on those ports. The passive analysis approach was selected by the ONR UNIX Network Administration Team, because the collection of active LAN traffic would be not impact ECSU's LAN/WAN assets. Moreover, this paper goal is to show that persistent packet monitoring of Ethernet traffic can identify weaknesses that reduce LAN performance and possibly harm valuable assets used to support major and/or general support systems.

Introduction

Traffic measurement and host performance monitoring of the Elizabeth City State University (ECSU) Office of Naval Research (ONR) and Center of Excellence in Remote Sensing Education and Research (CERSER) computer research Labs has become a daunting task since the existing network administration tools on the SGI 02 IRIX platform are not are not scaleable to meet the number and nature of demands placed on the research labs by ONR mentors and students. To meet the demands the ONR UNIX Network Administration Research team had to think about transitioning from LAN administration on the MAC, IREX, and Windows platform to a scaleable unified robust UNIX platform. The realization of inadequate existing support became the first step towards defining a research agenda for selecting an open source operating system and the foundation for addressing all project development tasks. The importance of building a UNIX based network administrative console using open source embedded network tools became the research agenda and direction for this project. The project phases were developed by adopting the System Development Life Cycle (SDLC), the overall process of developing information systems through a multi-step process from investigation of initial requirements through analysis, design, implementation and maintenance (QuickStudy: System Development Life Cycle; http://www.computerworld.com/developmenttopics/development/story/0,10801,71151,00.html).
The synchronize and stabilize model was selected because its methods combine the advantages of the spiral model with technology for overseeing and managing source code. This method allows many teams to work efficiently in parallel. This approach was defined by David Yoffie of Harvard University and Michael Cusumano of MIT. The ONR UNIX Network Admistration team utilized this method by splitting the team into two groups. One group focused on the configuration of the open source operating system and its administrative functions. The second group focused on the interpretation of packet content (headers or data encapsulated in packets) captured using Local Host sensors. At the end of each mentoring session, both groups would bring together all existing configured components to review progress and strategies.

Transition

Since Microsoft Windows operating system was the only operating system the team members have had experience working with, the team had to learn the fundamentals of Ubuntu. To accomplish this goal, the team analyzed Ubuntu features and operations in detail, including screen layouts, business rules, and process diagrams. The team built a logic flow diagram that listed which screen views and/or applications in Ubuntu are equivalent/ “replacement” programs to Microsoft Windows common applications they were accustomed to daily and developed standard operating procedures (SOP) that explained how to do program installations, and system configurations.

Defining the functions and operation of system applications was accomplished with the aid of Ubuntu’s installation manager called Synaptic which automatically downloads, installs, and configures new applications. Configuring and using Synaptic Package Manager to install applications allowed the team to install applications without

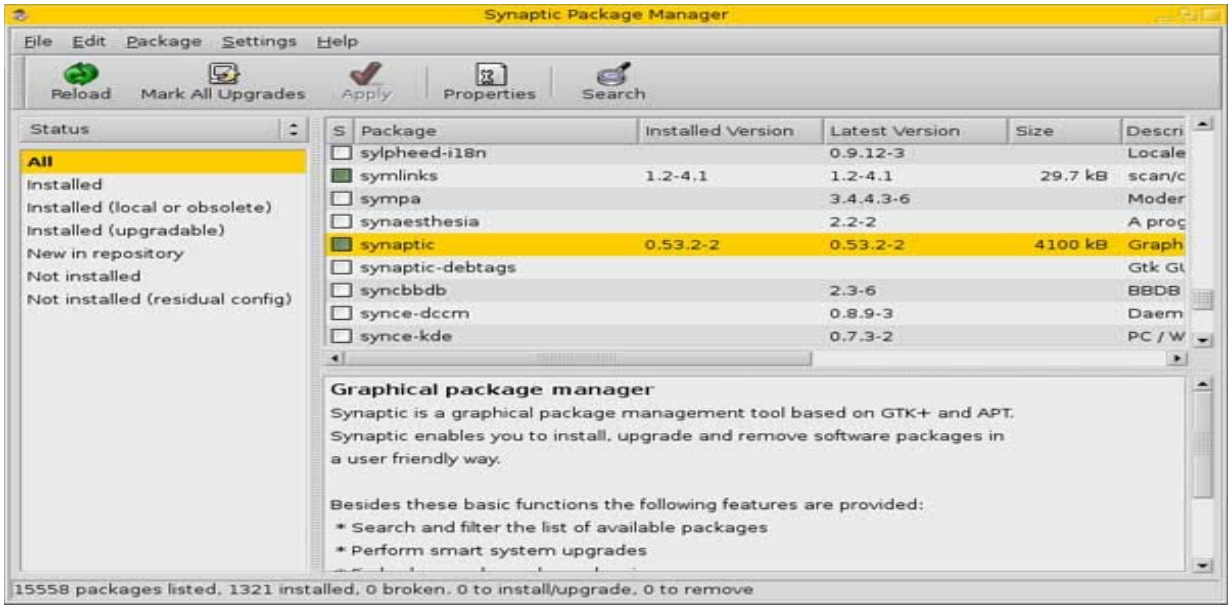
compiling them from source code. For example, listed below in figure one are two installation methods. Method 1 is the Microsoft Windows process and Method 2 is Ubuntu's process for installing applications.

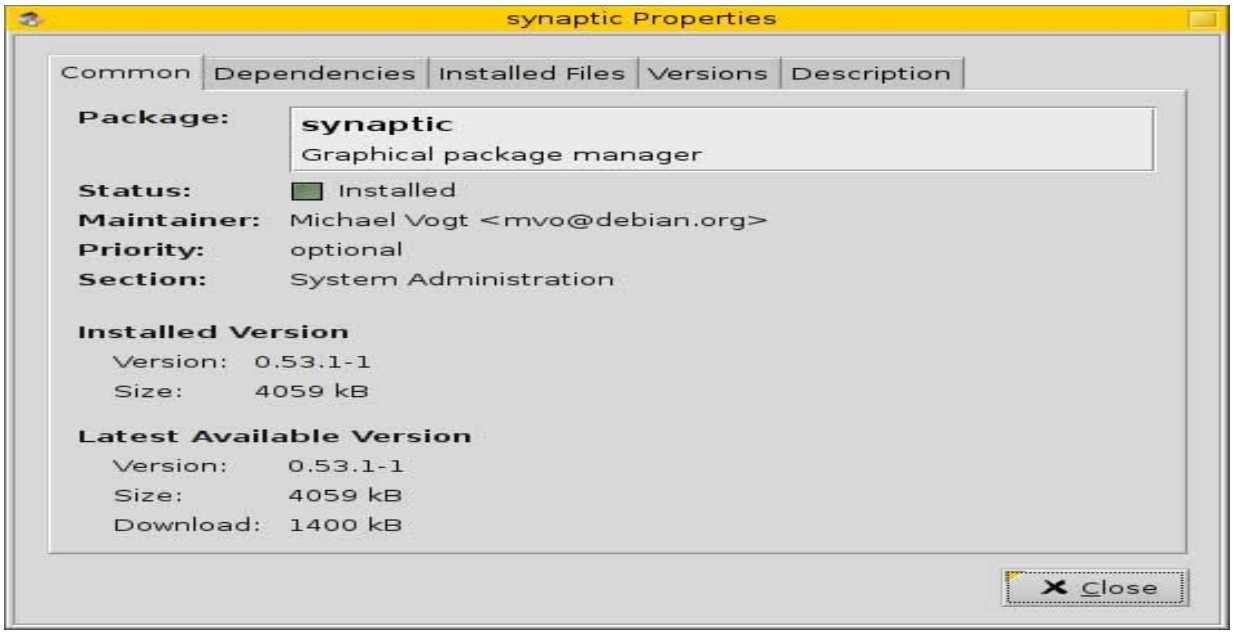
Method 1	**Method 2**
1. Search the web for a program	1. Search within a database for a program
2. Research and install any requirements	2. Select the program you want
3. Download the program	3. Click Install
4. Install the program	
5. Configure the new program to your environment (databases, etc.)	
6. *Pay* for a license to use the program	

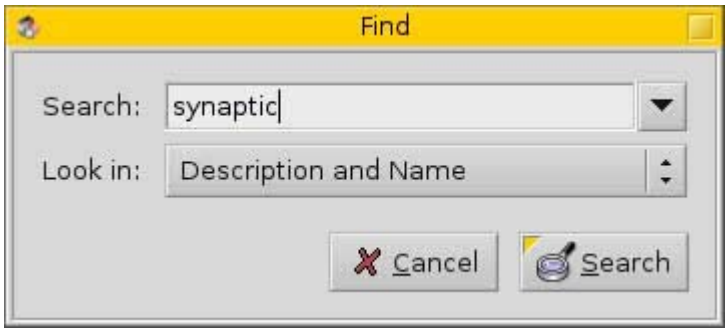
Method 2 reveals that the Ubuntu installation method is significantly easier than a Windows install. New installs are automatically configured to work with any applications you already have installed. Additionally, uninstalls are done as simply as un-checking the box for the application and applying the settings. To identify the programs required the ONR UNIX Network Administration Team configured Synaptic to display every program available in the Ubuntu "repositories". The repositories are simply locations which store information on which programs are available and where to download them. To enable all the repositories do this:

1. Open Synaptic (System > Administration > Synaptic Package Manager)
2. Enter your password (remember, installations require root access)
3. Select Settings > Repositories
4. Click Add
5. For each option under Repository combo box, select all the check boxes
   Note: This will enable certain closed source and sometimes non-free applications. Be sure you understand any EULA's for packages these may apply to (such as MP3 codecs).
6. Click Ok
7. In the Software Sources listing, make sure every box is checked, except for the CD Sources (should be the one at the top of the list)
8. Click Ok
9. Close and reopen Synaptic
10. Refresh the packages via Edit > Reload Package Information

Figure 2: *Synaptic Package Manager*


Synaptic Package Manager
File Edit Package Settings Help
Reload
Mark All Upgrades
Apply
Properties
Search
Status
All
Installed
Installed (local or obsolete)
Installed (upgradable)
New in repository
Not installed
Not installed (residual config)
S Package Installed Version Latest Version Size Descri
sylpheed-i18n 0.9.12-3 Locale
symlinks 1.2-4.1 1.2-4.1 29.7 kB scan/c
sympa 3.4.4.3-6 Moder
synaesthesia 2.2-2 A prog
synaptic 0.53.2-2 0.53.2-2 4100 kB Graph
synaptic-debtags Gtk GU
syncbbdb 2.3-6 BBDB
synce-dccm 0.8.9-3 Daem
synce-kde 0.7.3-2 PC / W
Graphical package manager
Synaptic is a graphical package management tool based on GTK+ and APT.
Synaptic enables you to install, upgrade and remove software packages in
a user friendly way.
Besides these basic functions the following features are provided:
* Search and filter the list of available packages
* Perform smart system upgrades
15558 packages listed, 1321 installed, 0 broken. 0 to install/upgrade, 0 to remove


synaptic Properties
Common
Dependencies
Installed Files
Versions
Description
Package:
synaptic
Graphical package manager
Status: Installed
Maintainer: Michael Vogt <mvo@debian.org>
Priority: optional
Section: System Administration
Installed Version
Version: 0.53.1-1
Size: 4059 kB
Latest Available Version
Version: 0.53.1-1
Size: 4059 kB
Download: 1400 kB
Close


Find
Search: synaptic
Look in: Description and Name
Cancel
Search

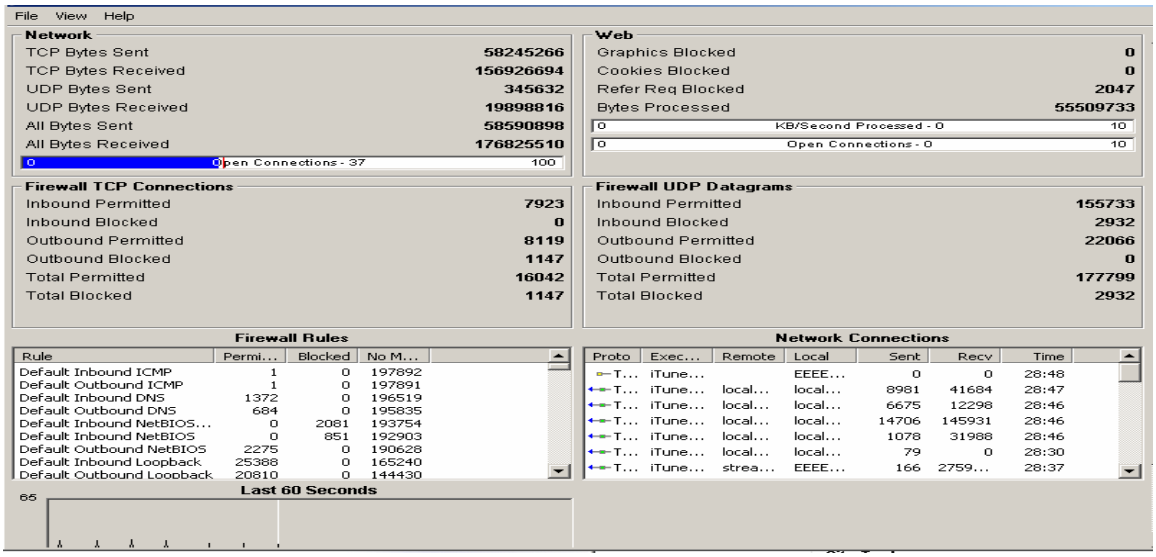
Installing and Configuring Firestarter

To recreate the network congestion problems the ONR research Labs are experiencing, the ONR UNIX Network Team configured their server with a software firewall called Firestarter. Firestarter is simply a graphic front-end to the network security features built in to the Linux kernel. It is lightweight, fast, unobtrusive and easy to use. Firestarter starts when Ubuntu does, so the host is never unprotected. The policy editor of Firestarter works pretty much the same way any router's configuration does. Every incoming request is blocked by default unless you explicitly specify to listen for it. Adding new rules to filter unwanted traffic that gets through the ECSU's main gateway and/or from another host on campus that is generating malicious noise can be implemented by doing the following:

1. Go to the Policy tab
2. Edit the Inbound traffic policy
3. Click Add Rule
4. Select or enter the application title
5. Enter the port range
6. Enter the local IP address of the machine on your network to receive the traffic
7. Click Add
8. Click Apply Policy

Another benefit of Firestarter is that the tool allows the ONR UNIX Network Research Team to open known hacker and/or exploited ports to view attempted connection attempts. By viewing intrusion attempts the team is able to see exactly how much garbage is caught, recent attempted attackers and/or the most frequent attacker. Most importantly, the team now have the IP address of the attacker and/or the orgin from the sender launched the attack.

Figure 3: Firestarter Statistics



Working with Protocols

IP Address (TCP -Transmission Control Protocol) PLEASE REVIEW

A set of rules that enables a broad spectrum of different kinds of computers to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent, so it is considered "reliable." Most long-haul traffic on the Internet uses TCP.

TCP handshake

A three-step process computers go through when negotiating a connection with one another. Simplistically described, in a normal TCP handshake:

1. Computer A sends a SYN packet (for "synchronize");

2. Computer B acknowledges the connection attempt and sends back its own SYN packet (thus, a SYN/ACK packet), and

3. Computer A acknowledges Computer B's response.

Once both computers are synchronized and acknowledged, they can begin passing data back and forth. To learn how attackers might exploit this, see SYN flood attack.

Subnetting

Subnetting an IP Network can be done for a variety of reasons, including organization, use of different physical media (such as Ethernet, FDDI, WAN, etc.), preservation of address space, and security. The most common reason is to control network traffic. In an Ethernet network, all nodes on a segment see all the packets transmitted by all the other nodes on that segment. Performance can be adversely affected under heavy traffic loads, due to collisions and the resulting retransmissions. A router is used to connect IP networks to minimize the amount of traffic each segment must receive. Subnet Masking Applying a subnet mask to an IP address allows you to identify the network and node parts of the address. The network bits are represented by the 1s in the mask, and the node bits are represented by the 0s. Performing a bitwise logical AND operation between the IP address and the subnet mask results in the Network Address or Number.

For example, using our test IP address and the default Class B subnet mask, we get:

10001100.10110011.11110000.11001000 140.179.240.200 Class B IP Address

11111111.11111111.00000000.00000000 255.255.000.000 Default Class B Subnet Mask ----------

10001100.10110011.00000000.00000000 140.179.000.000 Network Address

Default subnet masks:

Class A - 255.0.0.0 - 11111111.00000000.00000000.00000000

Class B - 255.255.0.0 - 11111111.11111111.00000000.00000000

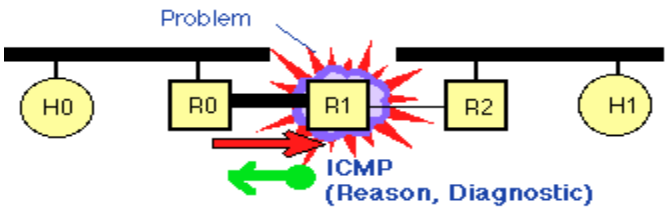
Class C - 255.255.255.0 - 11111111.11111111.11111111.00000000

ICMP

The Internet Control Message Protocol IP has a companion protocol that we haven't talked about yet. This is the Internet Control Message Protocol (ICMP), used by the kernel networking code to communicate error messages to other hosts. For instance, assume that you are on erdos again and want to telnet to port 12345 on quark, but there's no process listening on that port. When the first TCP packet for this port arrives on quark, the networking layer will recognize this arrival and immediately return an ICMP message to erdos stating ?Port Unreachable.?
>
> The ICMP protocol provides several different messages, many of which deal with error conditions. However, there is one very interesting message called the Redirect message. It is generated by the routing module when it detects that another host is using it as a gateway, even though a much shorter route exists. For example, after booting, the routing table of sophus may be incomplete. It might contain the routes to the Mathematics network, to the FDDI backbone, and the default route pointing at the Groucho Computing Center's gateway (gcc1). Thus, packets for quark would be sent to gcc1 rather than to niels, the gateway to the Physics department. When receiving such a datagram, gcc1 will notice that this is a poor choice of route and will forward the packet to niels, meanwhile returning an ICMP Redirect message to sophus telling it of the superior route.

This seems to be a very clever way to avoid manually setting up any but the most basic routes. However, be warned that relying on dynamic routing schemes, be it RIP or ICMP Redirect messages, is not always a good idea. ICMP Redirect and RIP offer you little or no choice in verifying that some routing information is indeed authentic. This situation allows malicious good-for-nothings to disrupt your entire network traffic, or even worse. Consequently, the Linux networking code treats Network Redirect messages as if they were Host Redirects. This minimizes the damage of an attack by restricting it to just one host, rather than the whole network. On the flip side, it means that a little more traffic is generated in the event of a legitimate condition, as each host causes the generation of an ICMP Redirect message. It is generally considered bad practice to rely on ICMP redirects for anything these days.

Figure 4: ICMP



Network Tools

Add information about the Network tools from the presentation.

FINGER

Figure 5: Finger 1

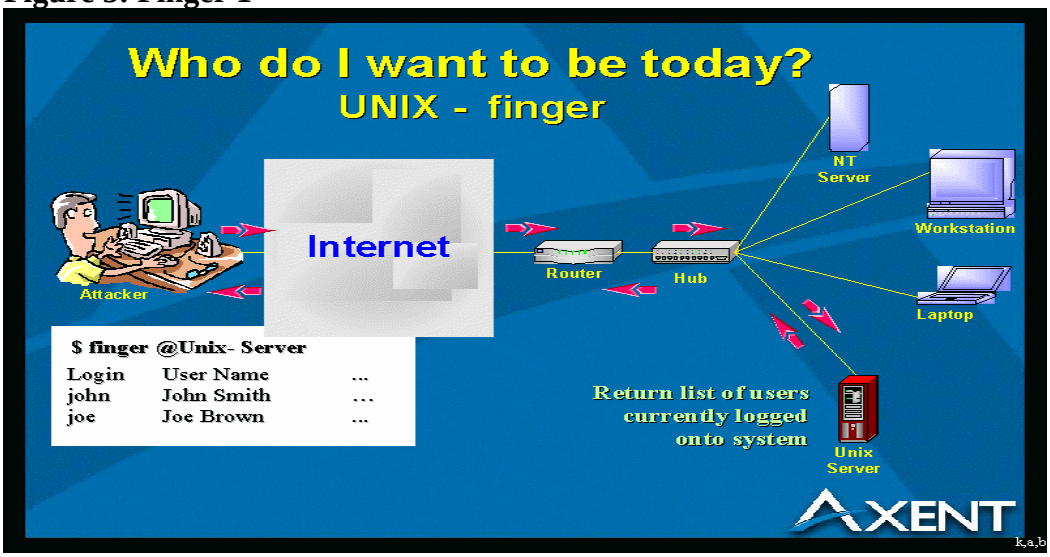


Figure: Finger 2

NETSTAT

Displaying Connections netstat supports a set of options to display active or passive sockets. The options ?t, ?u, ?w, and ?x show active TCP, UDP, RAW, or Unix socket connections. If you provide the ?a flag in addition, sockets that are waiting for a connection (i.e., listening) are displayed as well. This display will give you a list of all servers that are currently running on your system.

Invoking netstat -ta on vlager produces this output:

$ netstat -ta Active Internet Connections Proto Recv-Q Send-Q Local Address Foreign Address (State) tcp 0 0 *:domain *:* LISTEN tcp 0 0 *:time *:* LISTEN tcp 0 0 *:smtp *:* LISTEN tcp 0 0 vlager:smtp vstout:1040 ESTABLISHED tcp 0 0 *:telnet *:* LISTEN tcp 0 0 localhost:1046 vbardolino:telnet ESTABLISHED tcp 0 0 *:chargen *:* LISTEN tcp 0 0 *:daytime *:* LISTEN tcp 0 0 *:discard *:* LISTEN tcp 0 0 *:echo *:* LISTEN tcp 0 0 *:shell *:* LISTEN tcp 0 0 *:login *:* LISTEN

This output shows most servers simply waiting for an incoming connection. However, the fourth line shows an incoming SMTP connection from vstout, and the sixth line tells you there is an outgoing telnet connection to vbardolino.[1]

Using the ?a flag by itself will display all sockets from all families.

Conclusion

Add information about the Network tools from the presentation.