**RIS 420 - Research Assignment #1**

Members Names: Thomas Reid Zuk, Cyril Vicherek, Mike Nguyen

Date of submission: March 6th, 2016

**Table of Contents**

Netbios Page 3

Microsoft-ds Page 6

DNS Page 9

SSH Page 12

RSYNC Page 15

IRC Page 18

FTP Page 21

LDAP Page 24

RADIUS Page 29

GNutella Page 32

MySQL Page 35

Ms-SQL Page 38

**NetBIOS 137-139:**

**Description:**

NetBIOS stands for network basic input/output system. NetBIOS is an API that operates on the Session layer of the OSI model to provide a communication interface between an application and a network. It provides name resolution, datagram and session services across 3 ports; 137, 138 and 139 respectively. NetBIOS packets can be transported over many different Network layer protocols such as TCP/IP, UDP and IPX. Data using protocol 139, the session service, utilizes TCP, while ports 138 and 137 services utilize UDP. From RFC 1088 " NetBIOS is a standard which specifies a means of creating virtual circuits and of transmitting and receiving point-to-point, multicast, and broadcast datagrams."[1]. NetBIOS allows for old systems and applications to operate on a TCP/IP network. According to RFC 1002[2](1987) NetBIOS allows for various node types to be accommodated on local and internet topologies with or without IP Broadcasting. NetBIOS achieves this by encapsulating IP PDUs within a NetBIOS PDU and assigning IP numbers to hosts. A business or organization would use NetBIOS services in a device that is not supportive of modern IP network capabilities for supporting legacy services and devices. Most uses of NetBIOS are older file transfer services and file servers.

**Vulnerabilities:**

If not properly mitigated NetBIOS over TCP/IP can provide large amounts of information on a network such as hosts, services running and network layout. The Session service provided by the NetBIOS over TCP/IP allows two devices to establish a connection, having this exposed could allow someone to intercept and manipulate these services.

Using a tool called NBSTAT NetBIOS services on port 139 can reveal large amounts of information about a host. Computer name, remote name, IP Address, list of local NetBIOS names, contents of the session table and destination session IP addresses.

**Risks and Risk Mitigation:**

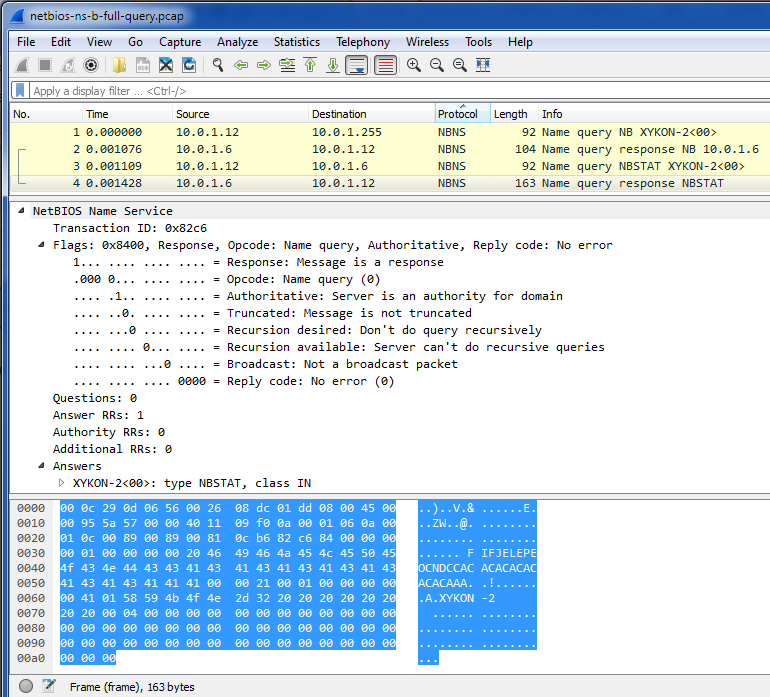
An attacker gaining such detailed information about a network and system they could use it to find other vulnerabilities within a network or a host. With the information gained using NBSTAT an attacker could possibly connect to a device if null sessions are allowed. From there they would be able to see all the connected shares. An attacker could retrieve sensitive data that is shared on the network which could have various levels of impact against the business such as financial reports, business plans, product planning and any other information employees are sharing through the service.

To help mitigate the risks the following steps should be taken. Disable null sessions, have strong administration passwords, have strong passwords for file shares if they are in place, disable the Guest account, do not allow root access remotely, do not share system folders such as Windows folders.

**How information is collected/logging:**

Since NetBIOS is a transport protocol and not a software or standalone service is does not keep logs. A device running a service that utilizes NetBIOS may keep logs like a file server may keep track of how connected, downloaded, uploaded files, but this is not logging NetBIOS itself.

**Wireshark Analysis:**



Screen shot of netbios pcap and the highlighted packet shows a Name Query response. The flags available are Response, Opcode, response type (in this case Authoritative), if the message is truncated or not, recursion desired, recursion available, where it is a broadcast packet or not, and the reply code which usually contains error codes. The answer to the original query is the name of the device, which is 'XYKIN-2<00>' as well as the type and class.

**Microsoft-DS: 445**

**Description:**

Microsoft-DS port 445 stands for Microsoft Directory Services. It is a Microsoft Active Directory service that is required for Active Directory access and authentication. (msdn site) It has been implemented for use since the launch of Windows 2000. Port 445 and Microsoft-DS replaced the usage of port 139 in NetBIOS service for file sharing and transfer. Port 445 is known as "SMB over IP". Business would use Microsoft-DS for file sharing services in Windows environments either locally, across the internet or through a VPN that connects multiple locations together. RFC's that were found about microsoft-ds were limited and did not include any relevant information to how the protocol operates or the services it served. The only information in the RFC's found were a list of services names and transport protocol port number that it uses.

**Vulnerabilities:**

Leaving port 445 open has been known to leave system vulnerable to worms which can have any malicious intent designed by the creator. A list of some past vulnerabilities related to leaving port 445 open are Sasser Worm, Korgo AB, HLLW Moega, Trojan Netdepix, Windows null session exploits and various root kits. One famous example is the Iraqi worm or iraqi\_oil.exe which uses a combination or port 445 open, weak passwords and privilege escalation to gain access to a target machine.

NetBIOS worms using port 445 have used the PSExec tool and other tools to replicate by continually scanning networks and the internet for other machines with port 445 open and replicating themselves. A trojan called Lioten is associated with port 445 vulnerabilities.

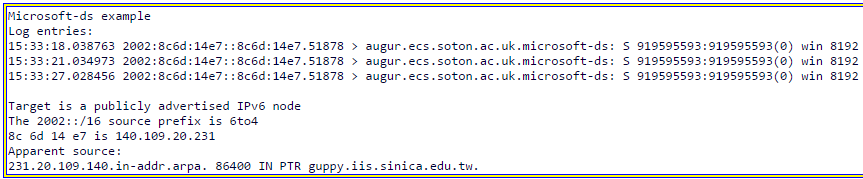
**Risks and Risk Mitigation:**

Remote control of a Windows machine with port 445 is possible. If port 445 is open on a host that host is highly susceptible to an attack and can even have a worm uploaded to the system. Users stand to have their machine remote controlled which means any documents may be stolen, any passwords or accounts that the machine has access to may be compromised and their machine may be used in further attacks to further spread the malicious code.

Many ISP have taken risk mitigation of port 445 vulnerabilities into their own hands by blocking any traffic using port 445. Although many ISP's help to prevent attacks associated with port 445 it is not a good idea to leave it solely in their hands. On a router blocking port 445 traffic will help to ensure no traffic uses port 445. In general disabling or blocking any machine exposed to the internet with port 445 is the best way to protect against an attacker.

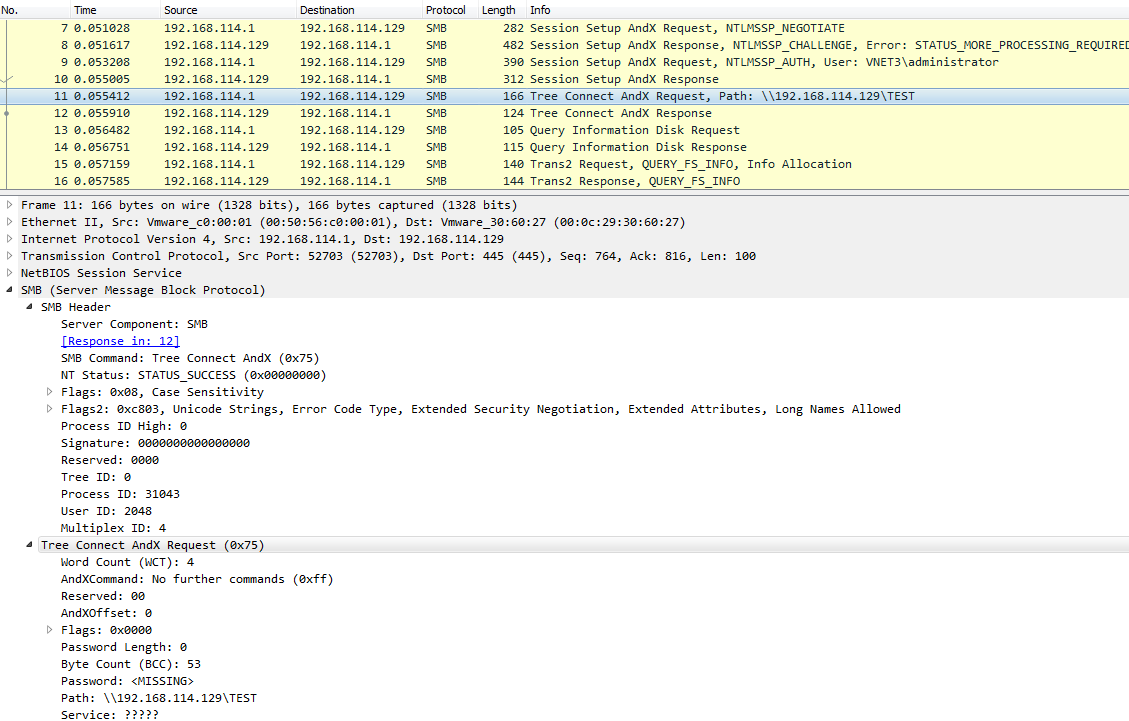
**Information gathered and Logging:**

Microsoft-DS stores log entries of devices that connect to a device running Microsoft Active Directory services. Below is an image of a Microsoft-ds log file.



[1]

**Wireshark Analysis:**



The screen shot above shows a connection attempt to a file sharing server. The first packet shows a request to connect, with the host replying with a challenge by asking for login credentials. The user replies sending it VNET3\administrator userID. The selected packet shows a request to the \TEST directory on the file server. The following packets are successful responses to the request to access the directory.

**DNS: 53**

**Description:**

DNS stands for Domain Name System and is a vital backbone for the functionality of the internet. DNS resolves domain names such as google.com and facebook.com into IP Addresses. All internet devices has a IP address, so DNS is a critical service for all users of the internet, especially non-technical users. Ports above 49152 can be used to receive local DNS and Remote DNS queries and responses to a local or remote DNS server. From RFC 1035 "From the user's point of view, domain names are useful as arguments to a local agent, called a resolver, which retrieves information associated with the domain name."[1] DNS can be encrypted which adds a level of security for the user knowing that their DNS is not being tampered with by a man-in-the-middle attack.

All websites, businesses, charities and personal websites alike all user DNS servers in some way or another. Large businesses like Google.com have their own DNS server that both the public and internal operations can use. Most websites use an external DNS provider when they purchase a domain name for their website to manage domain name to IP resolution services. Consumers looking to make personal websites may use a site such as GoDaddy.com to buy a domain, who in turn manages DNS services. Within an organizations network an internal DNS may be used to resolve names of local servers to make services easier to use such file servers.

**Vulnerabilities:**

Due to the nature of how many users browse the internet, through domain names, exploiting DNS services is a large target for attackers. DNS spoofing and DNS cache poisoning are two methods of exploiting DNS services that expose users to many attacks. Users can be exposed to man-in-the-middle attacks where-by they make a DNS request and at some point in the request cycle an attacker tampers with the request or response to redirect someone to a malicious website.

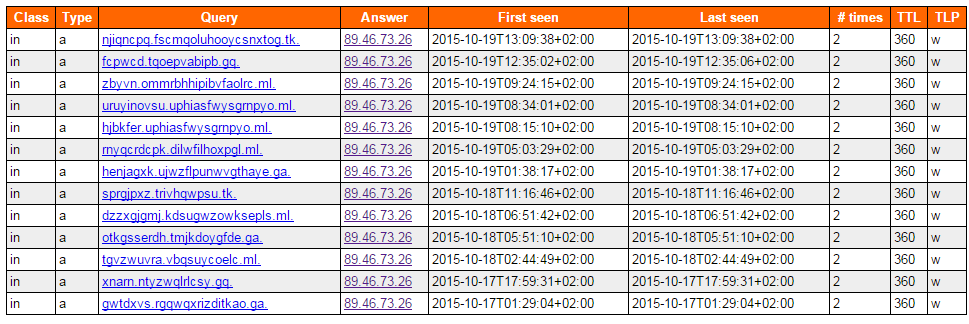
**Risks and Risk Mitigation:**

A device connected to a spoofed DNS could have any traffic querying a domain name redirected to a malicious site, traffic intercept (man in the middle) and traffic manipulated. A possible outcome of a DNS attack could be a user requests access to their banking website, and an attacker alter the request or the response to point to a phishing website they have designed to look like the banking website. The unsuspecting user would enter their log in information and it is now in the hands of the attacker. From there a smart attacker would forward the credentials to the bank website and redirect the user so they don't have any suspicion that they were just compromised.

A good way to prevent a device from being exposed to a DNS spoofing attack is to manually configure the DNS settings of a device to use only approved and properly configured DNS servers such as Google DNS. For organizations with services that use internal domain names a local DNS is required. Configuring machines to use the internal DNS as the primary DNS and then using a trusted DNS for internet queries is a great way to reduce and eliminate DNS spoofing attacks. Services called DNSCrypt are also available which encrypts DNS traffic so someone performing a man-in-the-middle attack could not tamper with DNS traffic, or make it exceedingly challenging to tamper with. DNSSEC is also a service available for DNS security. DNSSEC provides authentication as well as a chain of trust to establish a confidence in the response of DNS queries.

**Gathered information and Logging:**

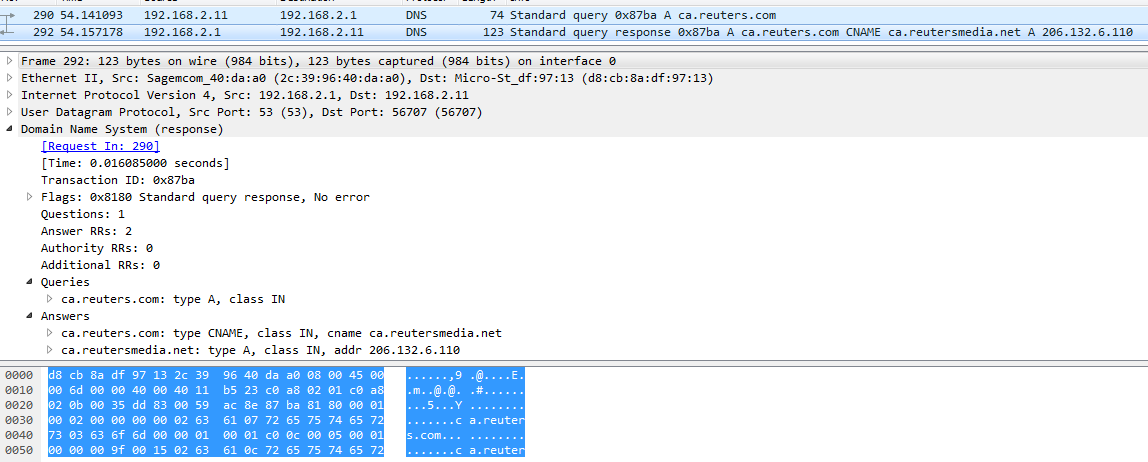
Logging of DNS on both the DNS server and the clients that utilize a DNS can be turned on and various level of details of logging can be specified. Below is a image of a sample DNS log.



[2]

**Wireshark Analysis:**

The picture below is a screen shot of a DNS query and a query response in wireshark. The request comes from the host 192.168.2.11 and goes to 192.168.2.1. The response is for the website ca.reuters.com. The response shows the original query for the domain name and the IP of that the requested website is hosted at, 206.132.6.110. A detailed look at the into the packet shows the time it took for the DNS response, flags set, the original query information and answers/responses. This particular response has no errors, has 1 question and contains 2 answers for the response.

****

**SSH:**

**Description**

The SSH (Secure Shell) Protocol is a “is a cryptographic (encrypted) network protocol operating at layer 7 of the OSI Model to allow remote login and other network services to operate securely over an unsecured network”[1]. In simpler terms, the SSH protocol grants a user access to a shell (either on a remote system or local system) over a secure, encrypted network connection.

The SSH protocol uses a client-server architecture. For example, let's say a user (*UserA*) wants to access a remote workstation (*StationA*), according to the architecture, *UserA* would be the SSH client and *StationA*would be the SSH Server, although it could go the other way around as well. SSH is often used using the *ssh* command most commonly found on Unix environments.

The default protocol port is *port 22* but can be configured (along with other options) in the configuration file (/etc/ssh/sshd\_config for the server and /etc/ssh/ssh\_config for the client). Further configurations include: the type of authentication to use (Public Key or Password Authentication); the SSH protocol to use (SSHv1 or SSHv2); Permissions (which user is allowed to login); Permitted IP Addresses and plenty more options.

Because SSH is a protocol used so widely, it can expose critical information to malicious users. For example, knowing that the protocol communicates over a network, a malicious user could run a portscan (e.g using the *nmap* utility) on a company's network and identify critical infrastructure which the attack could exploit. Further more, incorrect configurations could expose private information (e.g. passwords sent over plaintext). A malicious user on the *inside* of a company could use SSH to create a tunnel and bypass certain restrictions like portals, firewalls, proxies, etc.

A company could utilize SSH into their infrastructure to access servers or workstations for remote administration. Most of the infrastructure would act as a SSH Server which a system administrator would connect to via a SSH Client like a laptop, workstation, or even a mobile device.

**Vulnerablities, Risk and Mitigation**

SSH has numerous RFCs (being that it is a protocol which has been around for a while and has undergone plenty of changes). The initial RFCs are: RFC 4250 – RFC 4256[2], RFC 4335, 4344, 4345[3]

There are a few major vulnerabilities for SSHv1 primarily:

* Packet injection into an SSH session.[4]
* Arbitrary code execution via an integer overflow attack.[5]
* Man-in-the-Middle attacks which allows the attack to re-route traffic to another malicious server using the same session id.[6]

Due to many critical vulnerabilities in SSHv1, a new version meant to mitigate and patch these vulnerabilities was developed, SSHv2. However, SSHv2 is also susceptible to some attacks (varies based on implementation):

* Remote code execution and Denial of Service attacks.[7]

If not configured correctly, SSH could expose multiple risks. Primarily granting accessed to unauthorized and malicious users, theft of data and intellectual property and possible denial of service attacks.

**Logs**

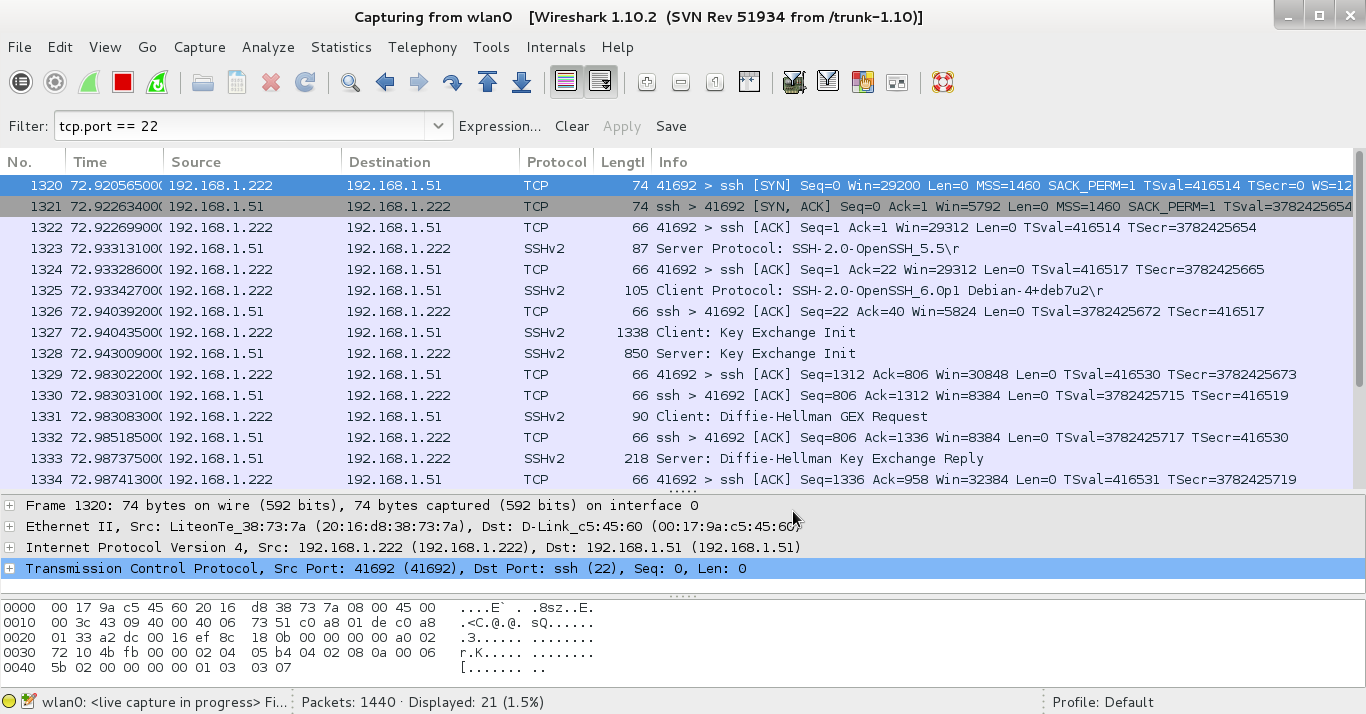
Data about SSH, such as log files and configuration files, can be found (on most \*nix systems) at:

* Configuration Files
  + /etc/ssh/ssh\_config
  + /etc/ssh/sshd\_config
  + ~/.ssh/known\_hosts
* Log Files
  + /var/log/auth
  + /var/log/secure

**WARNING:** Configuration file locations may vary from system to system, but the general idea is constant among the platforms.

Some of these risks can be mitigated by configuring SSH properly. This includes using strong Authentication methods such as Public Key Cryptography or Two Factor Authentication, allowing only authorized users to access the servers, white-listing known IP addresses and blacklisting unknown IP addresses, proper key management and firewall rule implementations (e.g. Allow only incoming and outgoing connections for certain ports, etc)

**Wireshark Analysis:**

Above is a sample packet of a SSH connection. As we can see, the connection is made using the SSHv2 Protocol, the latest version of the SSH Protocol. Further more, we are communicating over TCP. Before two host can communicate, there needs to be a key exchange between the client and the server.

**Rsync:**

**Description:**

Rsync is a tool which allows a user to securely transmit files over a network (either using TCP protocol or the UDP protocol) while using the least amount of resources as possible. The protocol ensures a secure connection is made by using the SSH protocol to create a secure tunnel connection.

Rsync ensures low resources are used by using a series of checksums, one cryptographic (md5sum) to ensure the data has not been tampered with and one non-cryptographic (adler-32) uses as a Cyclic Redundancy Check to ensure there was no transmission or copy error made during the transfer. When the files on the local and remote location are not the same, Rsync will copy only the difference between the files, rather than the entire file. This process works by checksuming the file in blocks and transferring only the blocks that differ and applying the new data to the file. Ports *873/tcp* and *873/udp* are the reserved ports for the rsync *daemon.*

A business could use *rsync* to transfer files from LocationA to LocationB over a secure network. An example of this might be scheduled backups. An administrator might use *rsync* to create a backup of one machine and transfer it to another machine, a designated backup server, for example.

**Vulnerabilities, Risk and Mitigation**

The *rsync* protocol only has one major RFC: RFC 5781[1]

There are three major Security Vulnerabilities released for the protocol:

* CVE-2007-6200[2]
  + A vulnerability which allows an attacker to bypass file filters and read or write hidden files.
* CVE-2007-6199[3]
  + A vulnerability which allows an attack to access unauthorized files by creating symlinks outside of the module hierarchy
* CVE-2007-4091[4]
  + A vulnerability which allows for arbitrary code execution.

Other than the vulnerabilities directly related to rsync, an attack could exploit the underlying SSH protocol which rsync uses to establish a secure connection.

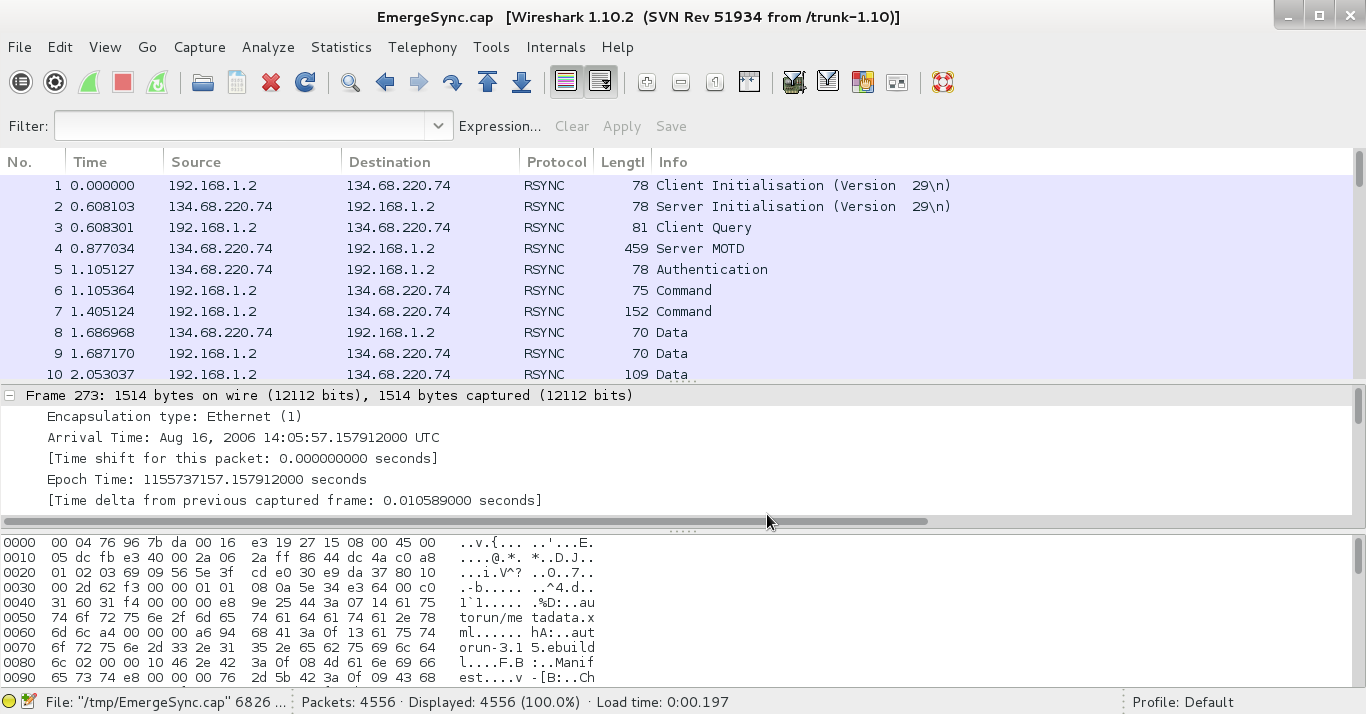
The two major risks in using the *rsync*  protocol are: 1) Loss of data if, by chance, some error occurs in the transfer of the file. This is not such a big risk because the data is being copies and not moved. This means that if some error occurs in the transmission, you can always recopy the files until the data matches, as it should. 2) If the data transferred was not encrypted and just sent in plaintext, a malicious user could sniff the traffic on the wire and steel the data being transmitted. If important files are being transferred, this may lead to a loss of intellectual property or harm to an organization.

Data files for *rsync* can be found on most \*nix systems at:

* Configuration Files
  + /etc/rsyncd.conf
* Log Files
  + rsync logs are generally sent of the syslog daemon, but only when rsync is running in daemon mode. To manually tell rsync to log the data, you can specify the –log-file on the command line when performing the rsync command.

The risks could be mitigated by ensuring you use the latest version of the rsync utility as well as the correct version of the SSH protocol.

**Wireshark Analysis:**

Above is a sample capture of Gentoo's *emerge-sync* tool which uses the *rsync* protocol. The transmission is made up of a few main parts: 1) A packet is sent from the client to the server saying “hey! I want to communicate with you”. 2) The authentication between the client and server is performed. 3) The rsync commands are sent. 4) The data that is to be copied is transmitted.

**IRC: 194**

**Description:**

IRC stands for Internet Relay Chat, it is used for exactly what the name implies, for text communication over the internet. There are many web application and client side IRC clients that are popular internet chat rooms for various use and well known for use by hacker groups such as LulzSec and Anonymous. Many IRC networks use alternative ports instead of 194, such as 6667, 6697, and 994. Some web site based IRC networks list they have used the follow ports 6665, 6666, 6667, 8000, 8001, 8002, 6697, 7000, 7070. Some ports are associated with SSL/encrypted traffic, others are associated with plain text IRC networks.

IRC is a text chat service, so any text sent using IRC services could be expose. This could include website links, personal information, stories, and general conversation information. The most common use for IRC by a business is running an IRC service for users. IRC is likely not used for internal business operations as it is unprofessional and very casual in natural. Most businesses would elect to use more secure and professional services like email.

Many browser based IRC services use SSL to encrypt their users traffic. Freenode, a popular IRC client uses 6697, 7000, 7070 ports for SSL traffic.

**Vulnerabilities:**

Some IRC services that are browser based inherit vulnerabilities such as cross site scripting, remote code execution, expose users IP's which have lead to DDOS and DOS attacks. Privilege escalation of the chat room or chat channel has been noted in some clients and services using IRC but this does not compromise a users machine. With privilege escalation a ordinary user could gain admin access of the channel and lock down the channel, kick users and prevent users from talking. IRC botnets do exist and were prime in the early years of the internet during the 1990's and 2000's and at relatively scarce today.

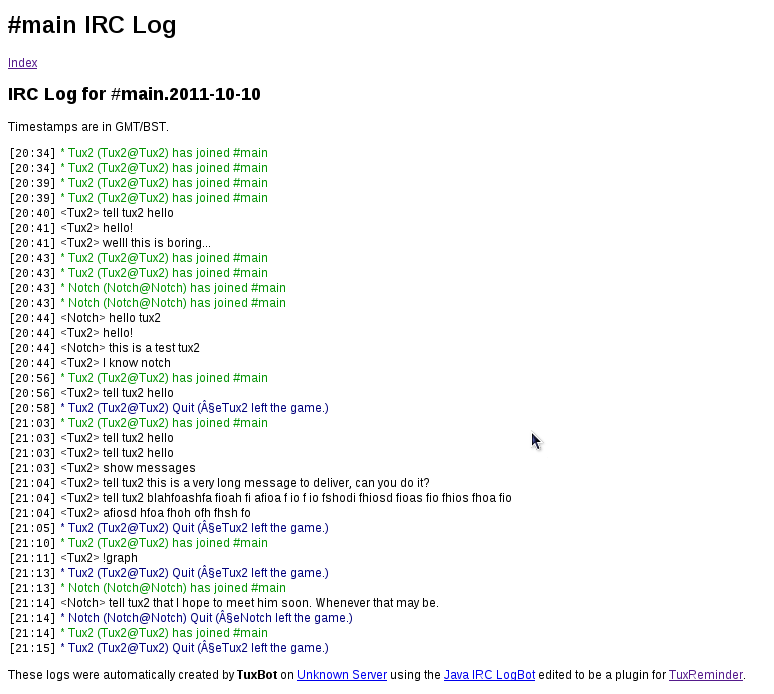
**Risks and Risk Mitigation:**

Any information shared across an IRC chat room, if it is not encrypted and not password protected can be spied on. Before sharing and private communications over IRC ensure the traffic is encrypted, the channel is private, or no users have joined that you do not want to share your communication with. There are well known standalone and browser IRC clients and information on their security can be found easily. An example of a standalone IRC client is mIRC, an example of a popular browser based client is Freenode IRC.

**Gathered information and logging:**

Both the server and client of IRC can log the chat and is decided by the user whether they store their chat logs and up to the administrators of the server if they want to log the chat or not. Logging could store basic information on the client side like names of the users, the channel name, the server address/name, users communications and time stamps of related events. The IRC server could log all this information and in addition log a users IP address.

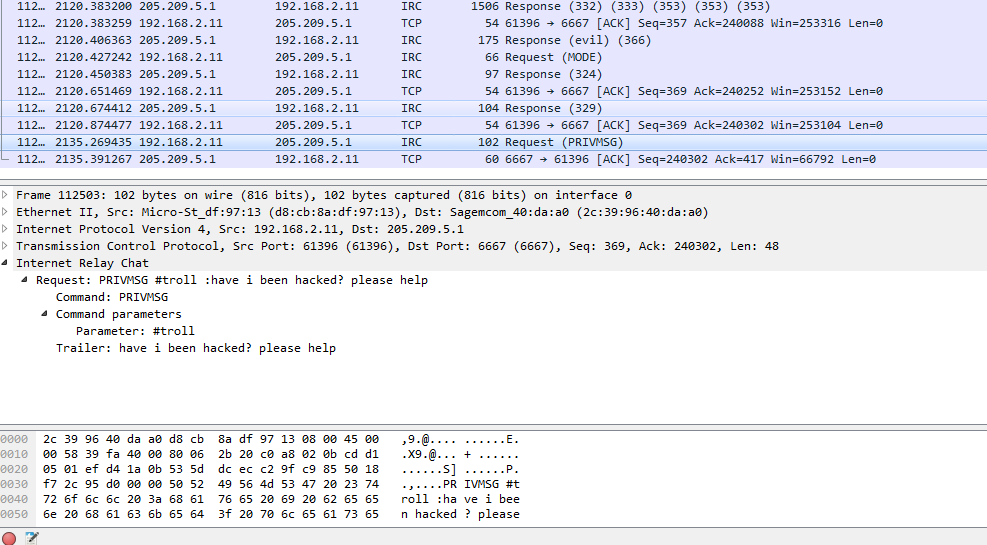
Sample IRC log



[1]

**Wireshark Analysis:**

The figure below is a screen shot of a IRC message. Inspecting the packet 2135.269435 (the selected packet) shows the message sent from the client 192.168.2.11 to the IRC server 205.209.5.1. The contents of the packet are very simple, a basic request to send a message signaled by PRIVMSG in the channel troll, indicated by '#troll' with the message "have i been hacked? please help".



**FTP: 20-21**

**Description:**

FTP stands for File Transfer Protocol, which allows for a client to be able to send or receives files from a server. FTP operates in Application Layer of the TCP/IP suite and OSI Models. The File Transfer Protocol was originally published in April 16 1971 as RFC114[1], and originally ran on the model that predated the current TCP/IP suite. The updated version became RFC765[2] in June 1980 which was when it migrated to TCP/IP, and further updated to its current documentation in October 1985 as RFC959[3]. Although FTP is most known to operate on port 20 as this is used for the actual data transfer, it also utilizes port 21 to as a control in which is used for communication between the client and server. FTP will use the TCP protocol in the transport layer in order to ensure that files successfully reach their destination. This data transfer has three possible modes which are stream mode, block mode and compressed mode. These are self-explanatory but FTP additionally has two modes of activity, passive and active which determine how the protocol establishes a connection between client and server. Passive mode was added in September 1998 in RFC2428[4] alongside of support for IPv6. Other updates to FTP in the form of security extensions came in RFC2228[5] which was published in October 1997. This was implemented to enhance confidentiality and integrity for the protocol and prevent sensitive information such as passwords from being viewed. It also protects commands, replies and data transfer between the client and server. Logs for this protocol are primarily kept in /var/log in Linux.

**Vulnerabilities**

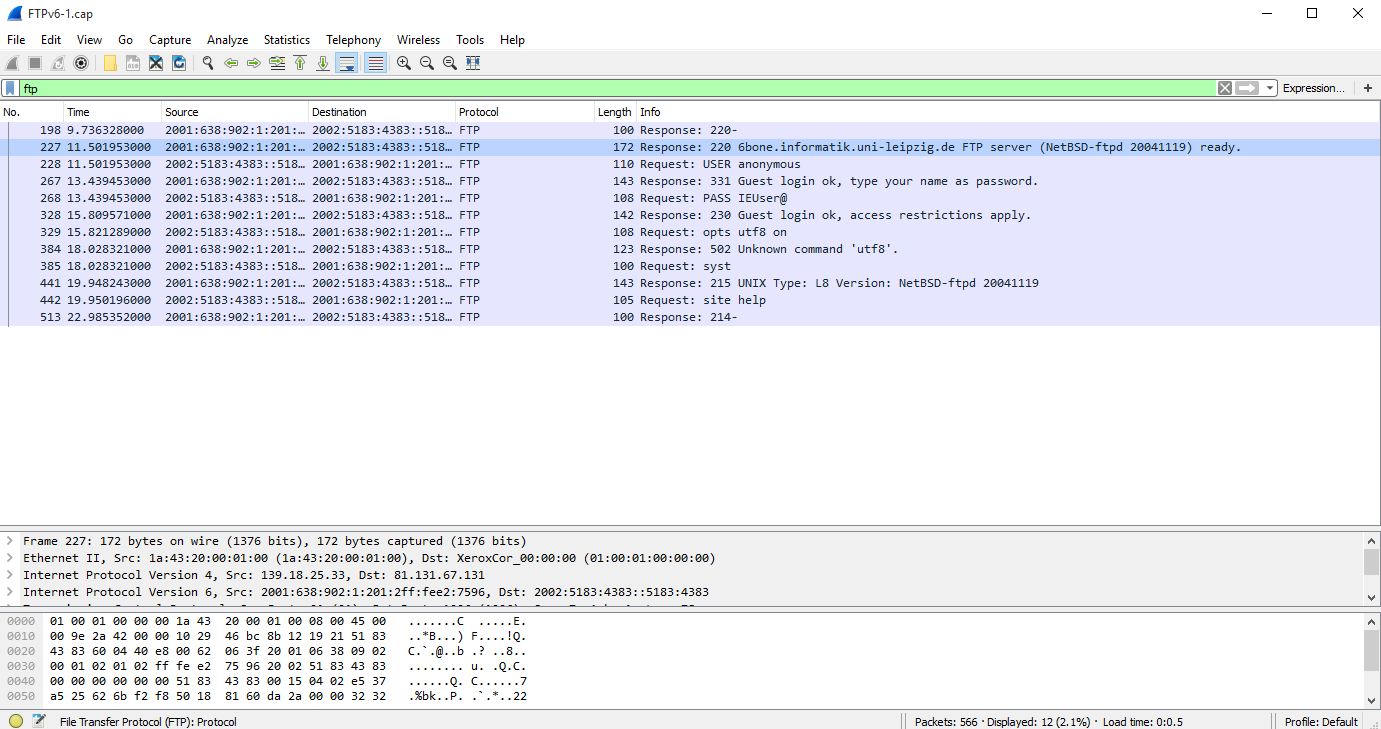
FTP is not designed as a secure protocol and in RFC2577[6], there is a list of issues and vulnerabilities that are associated with FTP. In the document is list that FTP is susceptible to a bounce attack. This attack involves sending the FTP “port” command to an FTP server containing the network address and port of the machine and service which is being attacked. This allows the attacker to attack another machine using the FTP server to push a file to that host through a vulnerable service. Also FTP is very susceptible to brute force if not properly configured. However the most easily exploited issue in FTP is that its traffic is not encrypted. This means that anyone doing packet capture will be easily able to capture the plaintext that is being sent to and from the FTP server. FTP is also vulnerable from port stealing. Lastly ‘by making a legitimate transfer, an attacker can observe the port number allocated by the server and guess the next one that will be allocated. This can allow the attacker to use all available ports and deny the service or allow the attacker to steal a file meant for someone else.’[6]

**Risk and Mitigation**

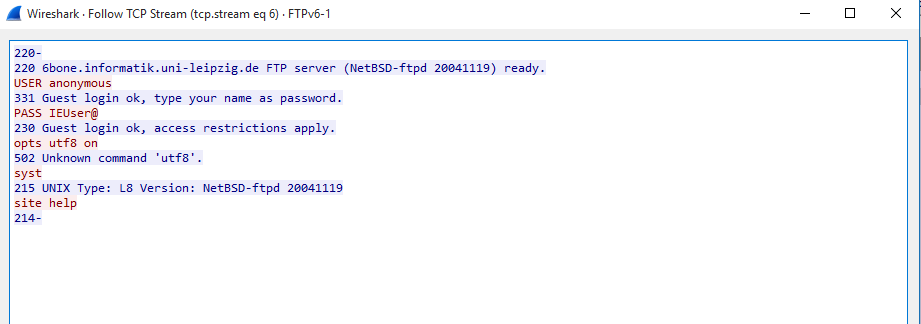
Since FTP was designed before the time of encryption, we find it to have many problems such as the possibility of exposing information due to the plaintext. Also the bounce attack allows for attackers to learn more about other machines that interact with the FTP server and learn what ports and services it has open/up. It will take extra precautionary steps in order to strengthen FTP and bring it up to standard. It is ideal to set-up a limit to the amount of attempts one can try to login to the server. This will lessen the possibility of being vulnerable to a brute force attack. RFC2228[5] also lists security extensions that will better secure communication between server and host and make it not plaintext. It will also hide commands that are being used during the communication. In order to deal with the FTP bounce attack, it is stated in RFC2577[6] to not permit “port” commands to ports whose TCP numbers are less than 1024 ( well-known ports utilized by many computers). This will still leave ports higher than 1024 which tend to be less important vulnerable but this is ideally good enough precaution. A final possibility is to disable the “port” command entirely if you want to totally mitigate the command being abused. FTP servers should also implement rules to limit which network addresses has the ability to access and download its files. It should be strictly only available to specified networks and remote hosts. Standard FTP will issue a response when the USER command which is used to ask whether this is a valid username exists. If it doesn’t exist, a response of 530 will appear otherwise if it does a response of 331. It is ideal that the setup always return 331, so that it will not be possible to continually guess at valid usernames. To avoid the possibly of port stealing, the FTP server should use random local ports for communication. There are additional security extensions in RFC2428[4] for FTP usage with IPv6.

There are now other alternatives, or rather derivatives of the FTP protocol such as FTPS and SSH FTP which comes with more security features. FTPS utilizes Transport Layer Security (TLS) and Secure Sockets Layer (SSL) while SSH FTP uses Secure Shell Protocol (SSH) to connect and transfer files.

**Wireshark Analysis**



This is a filtered conversation of packets using the FTP protocol. It is also good to note that this communication is using IPv6, however you can see that these machines use the Ipv4/Ipv6 dual stack and that this information can be found looking at the frame.



When we follow the packet stream, we can see that this FTP server has not used any security extensions, and that this information is in plaintext. This is not ideal as someone on the line, using a MiTM (man in the middle) attack could easily gain access to this FTP server.

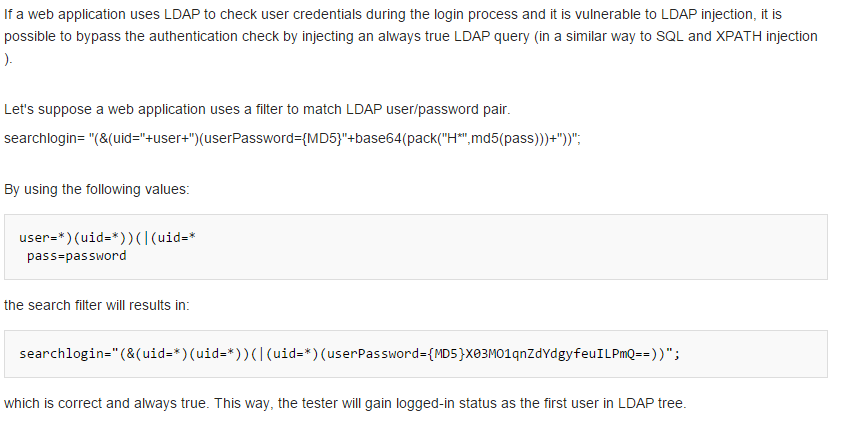
**LDAP: 389, 636, 3268**

**Description:**

LDAP stands for Lightweight Directory Access Protocol. LDAP is an open and vendor free service which means that it is not owned by any major company. As the name implies it is used for Directory information services and contains information about its users, computers, networks, and services and applications. It is ideally used to keep all information in a single location. LDAP was first introduced in RFC1777[1] in March 1995. At the time it was meant to work with the pre-existing X.500 Directory system but require less resources than the Directory Access Protocol (DAP) that was in place. Unlike the DAP system which was built on the OSI model, LDAP was purposed to use the working model of the TCP/IP Suite. LDAP and DAP have numerous different functions, but both follow the same structure. LDAP is able to perform the various tasks such as ADD, BIND, Delete, Search and Compare, and Modify entries in directories and databases. A company implementing LDAP may use it as a place to grab usernames and passwords. Since LDAP is a protocol for accessing directories and accessing databases, it can be used to accomplish a wide variety of tasks that mostly benefit from a centralized repository. With the update to LDAPv3 in 1997 found in RFC2251[2], it added TLS (Transport Layer Security) to the protocol. The most recent update found in RFC4511[3] provides the most up to date revisions to the protocol. LDAP by default will operate using TCP and UDP port 389. A common way to secure LDAP communication is by using an SSL tunnel, which by default utilizes port 636. LDAP is able to use LDAP-compliant directories, and thus pull its information. An example of this is the Global Catalog (GC). The GC which is a role given to a Domain Controller (DC) is a LDAP-compliant directory which holds a partial representation of every object can be accessed through port 3268. Searches done through the GC will also use this port. OpenLDAP houses its logs in /var/log by default with the name slapd.log.

**Vulnerabilities**

Using LDAP with the TLS connection enabled with the command “StartTLS” with LDAPv3 and up is very safe. However like many database applications such as SQL, LDAP can suffer from a server-side attack known as LDAP-injection. This is an attack which uses improper LDAP statements which could result in the attacker gaining information that should not be available. Similarly to SQL injection, the same principles and techniques apply to LDAP injection. This is a big deal as OWASP still lists SQL injection as a critical web application security risk (top 10). An example of this is found in the image below, and shows how it is possible to break the logically system. LDAP injection has many different angles of attack that may allow the attacker access unauthorized content, evade application restrictions, gather unauthorized information, or add/modify Objects inside the LDAP tree structure.

[4]

**Risk and Mitigation**

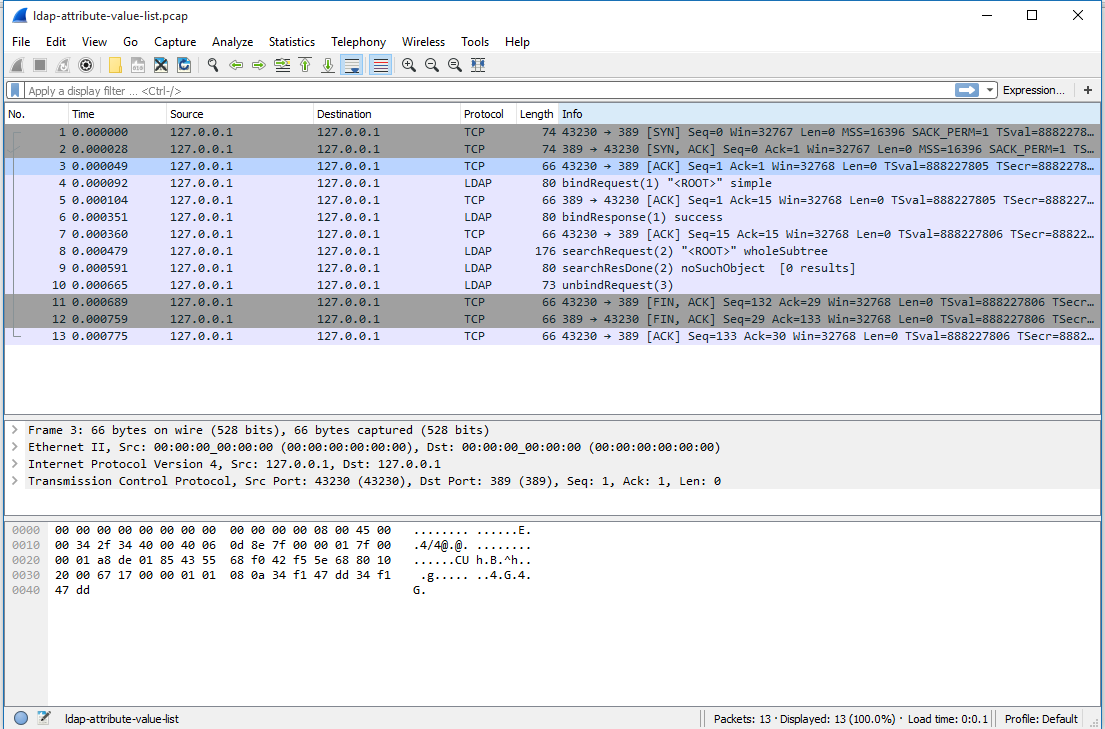
LDAP as a centralized repository means that it holds a lot of information regarding company staff, assets, etc. that could become compromised. In order to prevent communication from being captured, it is mandatory to ensure that TLS is running by using the “StartTLS” command. TLS/SSL was an optional part of LDAPv2, but in the current LDAPv3 has been made into a mandatory staple. Alternatively you can just run LDAPS which uses port 636 and automatically runs the protocol in SSL. In RFC2829[5], they outline the basic threats to the LDAP protocol which are listed below:

1. Unauthorized access to data via data-fetching operations
2. Unauthorized access to reusable client authentication information by monitoring others' access
3. Unauthorized access to data by monitoring others' access
4. Unauthorized modification of data
5. Unauthorized modification of configuration
6. Unauthorized or excessive use of resources (denial of service)
7. Spoofing of directory: Tricking a client into believing that information came from the directory when in fact it did not, either by modifying data in transit or misdirecting the client's connection.

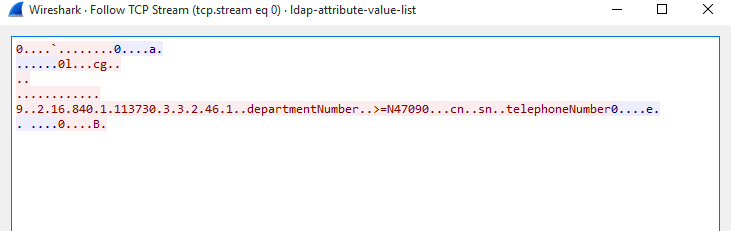
[6]

These issues can be solved by implementing client and server authentication using SASL (Simple Authentication and Security Layer) in order to prevent unauthorized access or directory spoofing. TLS/SSL as mentioned above will prevent threat 2, 3 and 7. Additional information can be found in RFC2829[5]. LDAP offers 3 different methods to authenticate users. They are No Authentication, Basic Authentication, and SASL (Simple Authentication and Security Layer). In No Authentication, it is as the name implies and is primarily used for information that is meant to be publically shared and of no value. Basic Authentication utilizes a login system however, this information is either sent in plaintext or base64 encoded. These two methods will not provide enough security for a company. SASL is a framework that enables the use of multiple different types of security mechanisms. These can include Kerberos, TLS, Digest-MD5 and many others. In order to combat the possibility of LDAP-injection, user/client input will need to be sanitized to prevent these dangerous search queries. The use of escape characters to replace existing ‘problem’ characters or symbols is ideal to solve this issue. These are just some ways to secure LDAP and keep sensitive information safe.

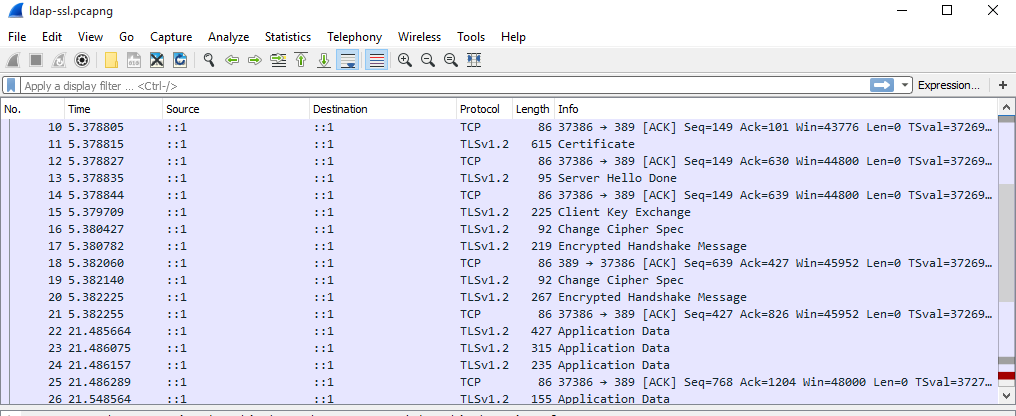
**Wireshark Analysis**



Under the info tab we can see that this set of packets captures is performing an LDAP search command. And after we look at the TCP stream, we can find out that it is returning a list of attributes.

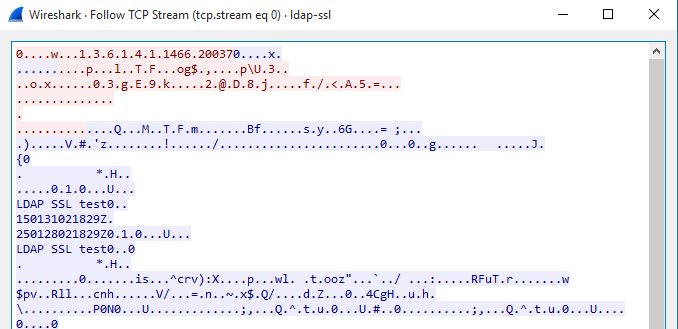


This is an example whereby this LDAP server has not implemented SSL, and results can be seen as plaintext. The example below will show what happens when SSL is implemented.



No longer is it possible to tell what type of command requests are being issued such as Bind, Search etc. These will all results in Application Data field in the info tab.

Following the TCP stream will lead nowhere now that the information is encrypted. Below is a small sample.



**RADIUS: UDP 1645-1646(old), 1812-1813; TCP 1645-1646, 3799, 2083 (Experimental)**

**Description:**

RADIUS stands for Remote Authentication Dial-In Service and is a networking protocol that provides AAA for users who connect and use it. AAA stands for Authentication, Authorization and Accounting and is provided by RADIUS in a centralized form. RADIUS is most typically run on both UNIX and Windows Machines. The details for Authentication and Authorization can be found in RFC2865[1] while Accounting is found in RFC2866[2]. Developed by Livingston Enterprises Inc. in 1991, RADIUS is a client/server protocol that at the time used UDP ports 1645 (authentication) and 1646 (accounting). At the time, it was a proposed solution to control dial-in access to NSFnet from Merit Network. It was not until 1997 that the protocol was added to the IETF standards in RFC2058[3]. As documented, RADIUS servers are responsible for receiving user connection requests using AAA and sending back configuration information to the client to deliver service. RADIUS is primarily used by internet service providers to access clients thus establishing a connection to the Internet. An access client might be an end-user trying to dial-in to a service provider, or be VPNs and wireless access points. These clients send their credentials in the form of a RADIUS message to a RADIUS server, where this information is authenticated and a reply is sent back. Nowadays RADIUS will use UDP 1812 for authenticating and UDP 1813 for accounting instead of 1645 and 1646 respectively. It is also possible for possible for RADIUS to communicate with TCP instead of UDP. In the experimental RFC6613[4] and 6614[5], RADIUS is sent similarly over TCP port 1812 and 1813. There is also the additional TCP port 3799 for dynamic authorization. Secured RADIUS using TLS (RFC 6614) will default use TCP port 2083. As aforementioned, RADIUS is primarily used by ISPs but it is possible that a company with a large set of IPs can make great use of this also. A UNIX machine with FreeRADIUS has its log configuration in the radius.conf. For Windows you will have to configure the Network Policy Server to perform accounting (logging) for the various features of RADIUS. DIAMETER is meant to be a successor to RADIUS.

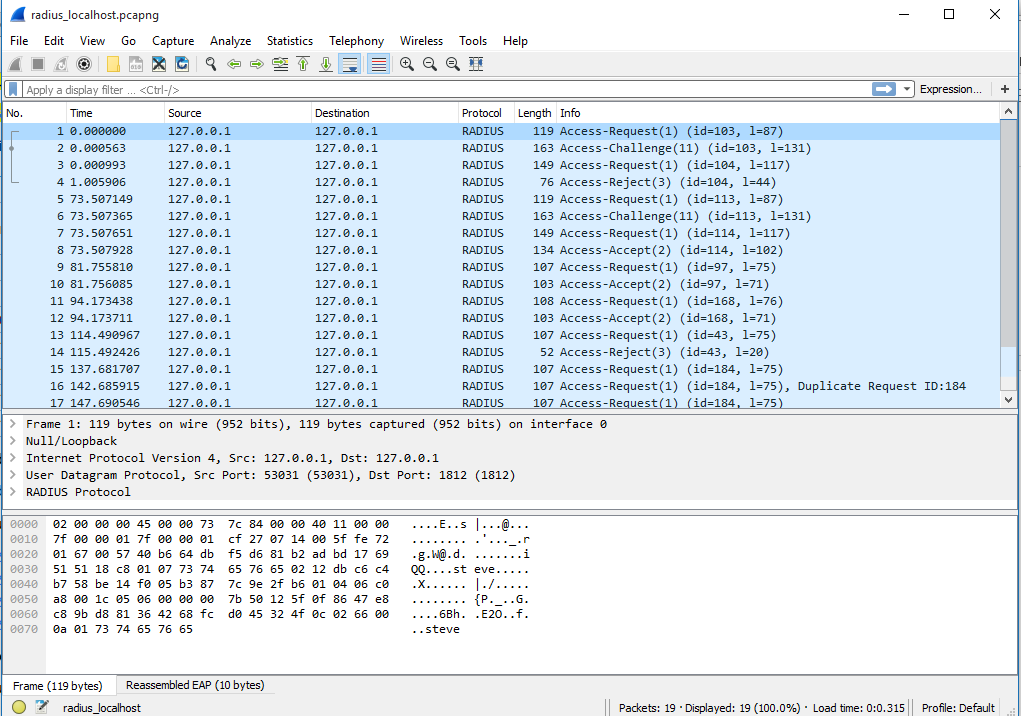
**Vulnerabilities**

Since RADIUS still primarily operations using UDP packets, this means that it is possible for an attacker to specially craft packets or forge packets with more ease compared to TCP. This also means that the protocol is more vulnerable to spoofing. Most attacks revolve around the shared secret, which is used by RADIUS in conjunction with MD5 hashing to conceal the password. In the case of a Response Authenticator Attack, if the observer sees a valid Access-Request, Access-Accept or Access–Reject packet sequence, an offline attack can be used to try to crack it. The attacker will have to compute the MD5 for the compiled fields and attempt to recreate the same hash. Other possible avenues of attack may include replay attacks that use replays of server responses, such as the Access-Accept and gain access without any login credentials. RADIUS only protects the user credentials, but other information that is passed through such as tunnel-group or VLAN memberships can be easily accessed. It is also noted that an attacker can perform a Denial of Service attack on a Network Policy Server (NPS) by using a packet with carefully crafted username strings that prevents authentication for the NPS thus preventing it from connecting. The Request Authenticator for RADIUS is not truly random and an attack may able to guess the next possible correct request.

**Risks and Mitigation:**

Since RADIUS deals with enabling connections for the different access clients, it is ideal that the RADIUS message be safe from possible attackers that may be listening. Potential risks that may arise from RADIUS is that like mentioned before, RADIUS by default only secures information regarding the username/password, and other pieces of information is available to prying eyes. By using IPsec, the whole RADIUS message can become encrypted and the RADIUS sensitive fields such as Access-Request, Access-Accept and Access-Reject can be hidden also. The method that RADIUS uses to secure the usernames and passwords with shared secret and MD5 is not ideal. As the aforementioned topic of attacks on the shared secret is quite common for RADIUS one may implement the Radsec protocol found in RFC 6614 to strengthen RADIUS. Radsec claims to fix the flaws in the RADIUS sensitive fields and also strengthens the username and passwords. Since MD5 is proven to be weak and UDP being much easier to spoof for, Radsec adds TLS (Transport Layer Security) Encryption for RADIUS. It also makes it so communications no longer exist on the UDP transport protocol but rather the TCP transport protocol.

**Wireshark Analysis**



By looking at the capture we can tell what type of actions are. As mentioned in the vulnerabilities segment for RADIUS, an attacker can look a valid Access-Accept and try to break it offline. Or try to do a replay attack by using the same segment.



The above is a result of following the stream for an Access-Accept. Most of the information is hidden away from plain sight.

**Gnutella:**

Gnutella is a “large peer-to-peer network. It was the first decentralized peer-to-peer network of its kind, leading to other, later networks adopting the model. It celebrated a decade of existence on March 14, 2010 and has a user base in the millions for peer-to-peer file sharing.”[1] Essentially, Gnutella allows users to share files between peers over a network which is decentralized, that is to say, does not go through a specified central location or container centralized servers. In other words, the network is made up of individual peers connected together. The network knows which clients are on the network by utilizing Distributed Hash Tables, a system which is

“a class of a decentralized distributed system that provides a lookup service similar to a hash table: (*key*, *value*) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.”[2]

According to */etc/services*, Gnutella uses ports:

* gnutella-svc
  + 6346/tcp and 6346/udp
* gnutella-rtr
  + 6347/tcp and 6347/udp

The service might expose important information about an organizations infrastructure which an attack could use to exploit and gain access into the system. If a malicious user were to portscan a companies network and discover a Gnutella server, the attacker could use an exploit to gain entrance into the company's system and cause a denial of service, or damage such as loss of intellectual property.

A business could use the protocol internally in their company to create an isolated network on which to share data. The advantage of this is that the data is distributed across multiple peers and does not require a central client-server architecture, but rather data can be pulled from any peer currently accessible. This might be useful if the company is low on resources and cannot afford that sort of infrastructure, but can accomplish the same thing with available workstations acting as peers.

Although there does not seem to be an official IEEE documented Gnutella RFC, there is a project (RFC-Gnutella)[3] for the purpose of documenting the evolution of the protocol.

As far as know vulnerabilities to the protocol. There is a CVE that was published:

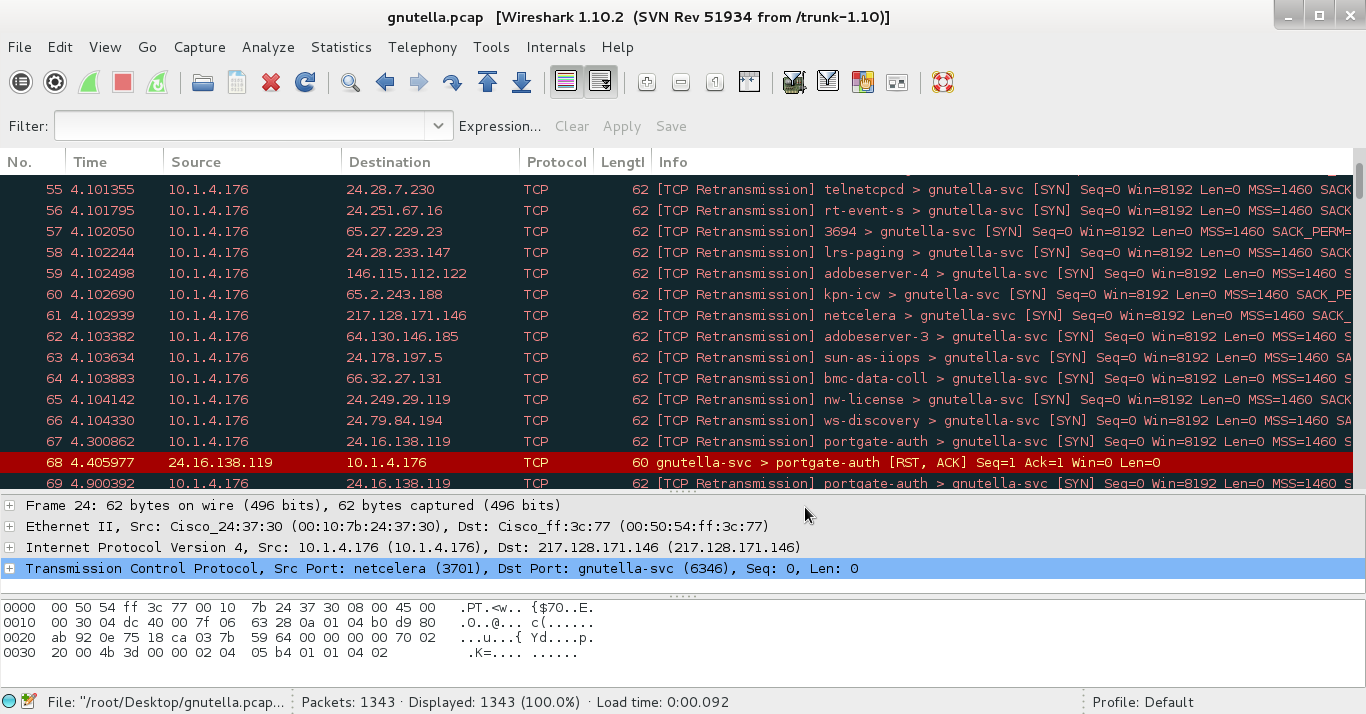
* CVE-2001-1004[4]
  + A cross-site scripting attack in a gnutella client.

As far as attacks on the actual architecture and implementations of the protocol. There was some research[5] done by the University of California which describes many attacks on a Gnutella network including: Distributed Denial of Service Attacks, *Pong* Attack, IP Harvesting, Malware Injection and Privacy Violations through GUID tracing.

The major risk in using the service is users may be prone to Distributed Denial of Service attacks which could cause loss of business which could potentially harm the company.

The majority of configuration is done through a Gnutella client such as *gtk-gnutella* which allows users to configure various aspects of the system.

To mitigate the risks of gnutella, make sure to update the package and clients as frequently as possible to ensure all vulnerabilities are patched as quickly as possible. Ensure you have proper firewall rules in place to deny access to outside IP addresses that are not part of the internal company network.

The *gnutella* protocol uses the TCP protocol to transmit data. The above capture shows a series of TCP transmission packets between gnutella clients.

**MySQL: 3306**

**Description:**

MySQL is a relational database system that was created by Oracle in 1995. MySQL has a registered port of 3306, meaning the Internet Assigned Numbers Authority (IANA) has assigned and the port for use with the MySQL port. No other ports are commonly associated with a MySQL data base because it is a registered port and service.

Being a database, MySQL could potentially expose any information stored in a database that uses MySQL. MySQL is one of the top relational database platforms and is used by organizations such as NASA, Verizon, and YouTube. MySQL may be used by any software, website, or web application for its relational database needs. Encrypting MySQL traffic, via a SSL certificate for a website, is highly recommended. Sending unencrypted database traffic exposes any connection to the database to interception of data. This could be anything from price of a product on a eCommerce platform, to credit card information, to usernames and passwords. MySQL on a website would be server side so the actual query to the database would not be sent from a users browser to the server, but the data retrieved and being submitted to a database should be encrypted.

**Vulnerabilities:**

Databases are a large target for many attacks, user data ranging from pointless internet forum comments, to names, passwords, addresses and emails can be very valuable for attackers. MySQL has had its fair share of vulnerabilities in the past, and there are currently no known vulnerabilities. The past year, 2015, saw only 1 vulnerability published and fixed, in comparison to 2014 which saw 38 vulnerabilities. SQL injection is a infamous vulnerability with the SQL platform, MySQL has only had 4 known total SQL injection vulnerabilities, all of which have been patched.

**Risks and Risk Mitigation:**

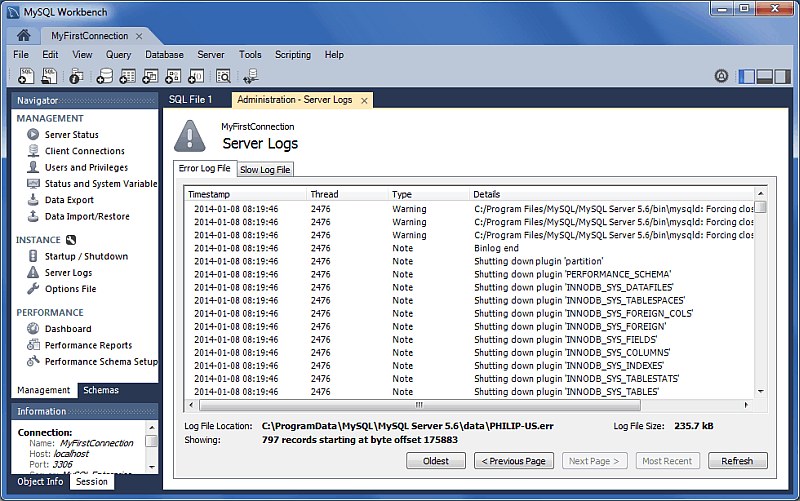
Practices to help prevent any exploitation of MySQL services include never allow 'root' access remotely, never use the 'root' user account for executing queries from a service such as a website. Have strong passwords for all accounts, especially the root account. Keep MySQL services up to date to prevent against any known vulnerabilities that may be discovered.

**Gathered Information and Logging:**

MySQL servers support logging of various types of information the service uses. Logs of queries performed, time to execute query, and statistics can be recorded. Creating statistics of the logs is often done to help improve efficiency of a database. Logging of users connected to the database and the queries they run is highly recommended for security. Knowing which user performed which queries can help trace a possible attack, loss of data and recovering of data.

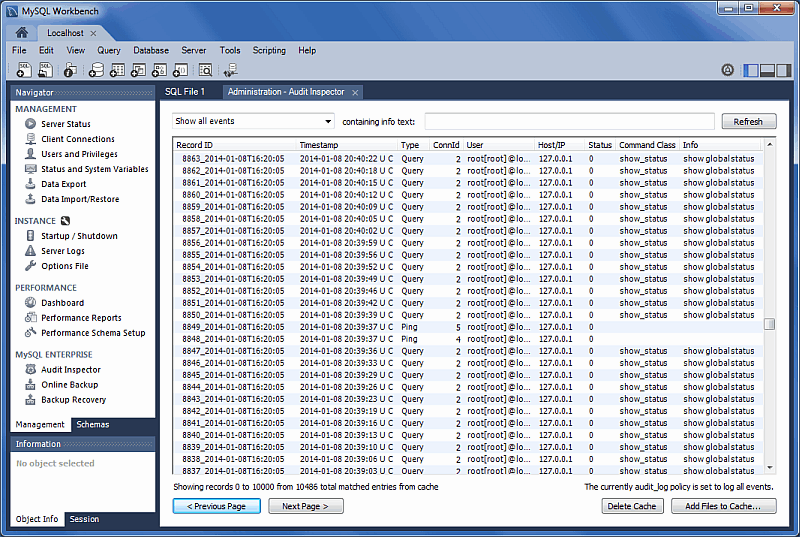
Sample of a MySQL server event log.

[1]



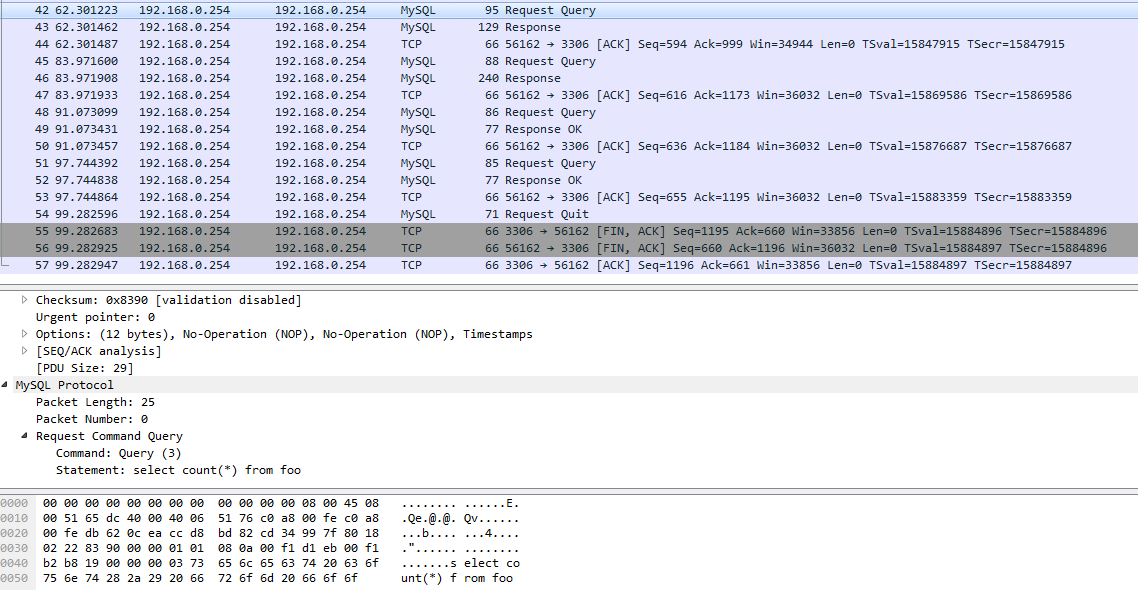
Sample of MySQL server query log

[2]



**Wireshark Analysis:**

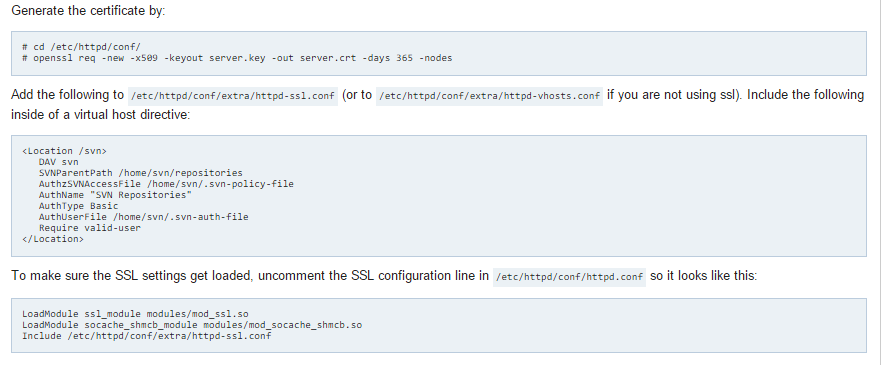
The screenshot below shows the various MySQL transactions between a client and a MySQL server. The highlighted packet shows a query request to the server. The contents of the packet are very simple, command type and the statement. In this case the command is "Query" and the statement is "select count(\*) from foo".



**SVN: 3690**

**Description:**

SVN is the abbreviated form for the Apache Subversion protocol which is primarily used an “enterprise-class centralized version control for the masses.” Subversion which was founded by CollabNet Inc. was a project that started 2000 and within a year was able to sufficiently operate on its own code. The Subversion project was to be an open source version-control system which would be the successor to CVS (Concurrent Versions System). It would operate similarly to the widely used CVS at the time, but be free and widely accessible. Subversion is used for keep track of revisions and changes done to a set of files. Using a client-server structure, the server would store the current version of a project/task in its history, and clients would in turn have to “pull” the file to work on it, and to commit would have to “push” it back to the server.It was not until Feb 23 2004 would they release their first iteration and in 2009, that it would be accepted into the Apache Incubator. Once a project has entered the Incubator, it becomes a top-level Apache project that gets more attention for development. Currently SVN operates at the latest version of 1.9 with partial support for machines running the older SVN 1.8. The latest update SVN was in December 2015. SVN by default will operate on TCP port 3690. A company might use SVN to help manage projects that are being worked on by many simultaneous clients. It also allows the company to keep track of every bit of change done to the project, providing a form of audit trail in case they need to revert some changes. It is possible to enable SSL for SVN. To do this, we must go into the Apache configuration located at /etc/httpd/conf/ and modify a couple of files. The steps to do his are in the image below. Otherwise most security features for SVN themselves are solely based on the current Apache configurations.



[1]

**Vulnerabilities**

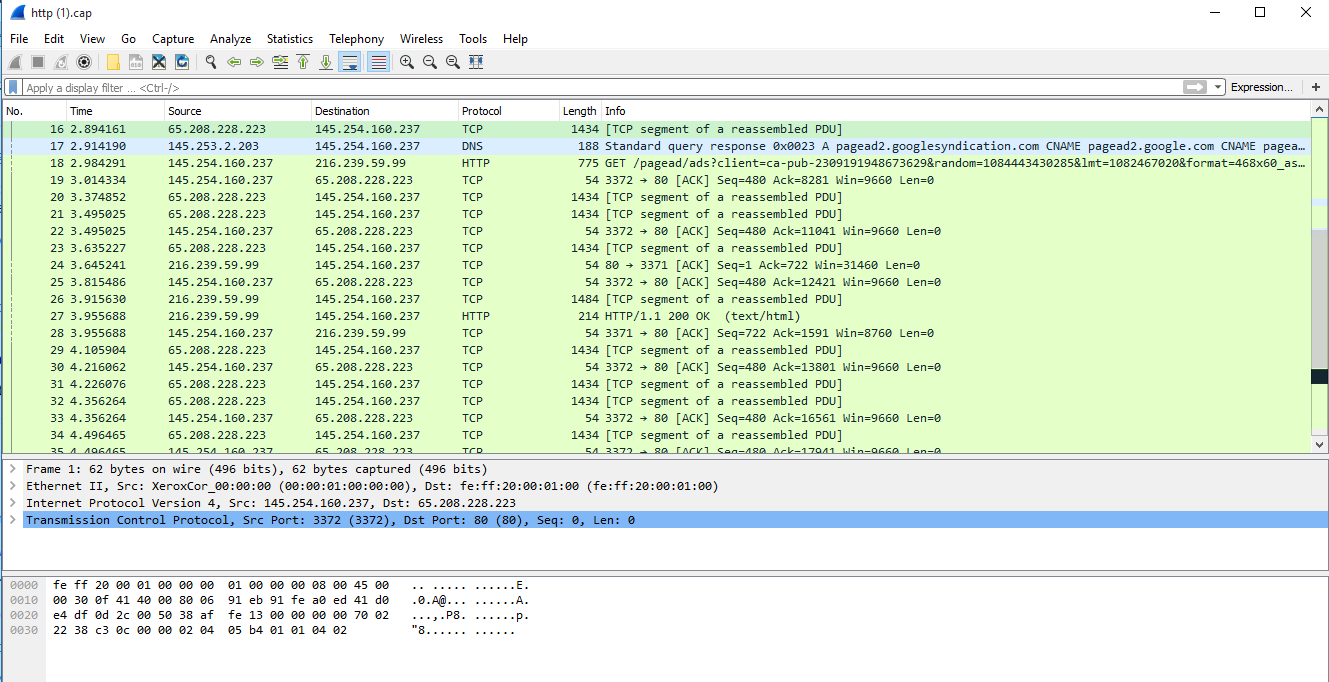
In most recent version 1.9-1.9.3 there is a possible way for to cause an integer overflow in the function read\_string. Older versions of SVN are problematic as they have bugs that thus been fixed in the latest iteration. Some of these issues may lead or cause various types of security flaws. One such example is in versions 1.8.0 through 1.8.11 for the mod\_dav\_svn server may cause a denial of service due to a memory consumption when an attacker uses a large number of REPORT requests. In versions 1.7 to 1.8.11 there exists a flaw that may let a remote attacker cause a server crash using a REPORT request for a resource that doesn’t exist. The majority of these issues are targeted towards denying the SVN server either through an infinite loop or a crash.

**Risks and Mitigation:**

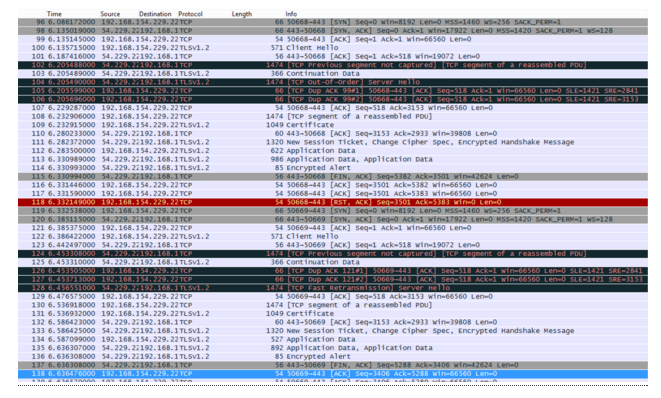
Sensitive information is of concern when dealing with SVN and it is important also that we can keep the SVN server up for as long as possible. Since the concern of most attackers is take the servers itself our best preventative measure is to ensure that any known vulnerabilities are patched out of the system. It is ideal to run the latest version or a version that is still currently supported. As one the main projects from Apache, SVN will routinely get updates that we should implement as soon as possible. For those vulnerabilities with a High or Critical CVE (Common Vulnerabilities and Exposure) rating, Apache will tend to roll out patches within a week to address these issues. Such is the case for the integer overflow on the most recent version. Within 3 days of publishing, an update was sent out to deal with it. Adding SSL to the regular Apache service also beefs up SVN. We may need to look into strengthening Apache itself as the two are correlated and interworking.

**Wireshark Analysis:**

Since Subversion is an Apache service, we can usually find it running alongside an HTTP service.To filter out for the ra\_svn protocol in Wireshark, we must use a tcp port filter on port 3690.



The info tab should pull up information regarding requests from clients as well as. Below is an example of implementing SSL with SVN, and capturing the conversation of a commit.



Due to a misconfiguration in the http.conf with no KeepAlive statement, even though it is the same IP making the commits, every 10kb of data has to redo the SSL handshake making it run very poorly.

**Sources**

**NetBIOS**

[1] https://tools.ietf.org/html/rfc1088

[2] https://tools.ietf.org/html/rfc1002

http://www.techrepublic.com/blog/it-security/the-problem-with-netbios/

http://www.pcmag.com/encyclopedia/term/47773/netbios

https://technet.microsoft.com/en-us/library/cc940063.aspx

https://wiki.wireshark.org/NetBIOS

https://isc.sans.edu/port.html?port=445

http://www.pcapr.net/browse?q=netbios

**Microsoft-DS**

[1] https://www.ietf.org/proceedings/68/slides/v6ops-6/sld7.htm

http://www.speedguide.net/port.php?port=445

https://msdn.microsoft.com/en-us/library/dd979226(v=crm.6).aspx

https://wiki.wireshark.org/SampleCaptures

**DNS**

[1] https://www.ietf.org/rfc/rfc1035.txt

[2] https://www.invincea.com/wp-content/uploads/2015/10/T-Online-malvertising-attack-DNS-logs-image-1.png

https://technet.microsoft.com/en-us/library/dd197515(v=ws.10).aspx

https://developers.google.com/speed/public-dns/

https://ca.godaddy.com/help/what-is-dns-665

https://www.infoblox.com/sites/infobloxcom/files/resources/infoblox-whitepaper-cybercriminal-guide-exploiting-dns.pdf

**SSH**

[1] https://en.wikipedia.org/wiki/Secure\_Shell

[2] https://tools.ietf.org/html/rfc4250

<https://tools.ietf.org/html/rfc4251>

<https://tools.ietf.org/html/rfc4252>

<https://tools.ietf.org/html/rfc4253>

<https://tools.ietf.org/html/rfc4254>

https://tools.ietf.org/html/rfc4255

https://tools.ietf.org/html/rfc4256

[3] https://tools.ietf.org/html/rfc4335

https://tools.ietf.org/html/rfc4344

https://tools.ietf.org/html/rfc4345

[4] http://www.kb.cert.org/vuls/id/13877

[5] http://www.kb.cert.org/vuls/id/945216

[6] http://www.kb.cert.org/vuls/id/684820

[7] https://www.rapid7.com/resources/advisories/R7-0009.jsp

**RSYNC**

[1] https://tools.ietf.org/html/rfc5781

[2] https://www.cvedetails.com/cve/CVE-2007-6200/

[3] https://www.cvedetails.com/cve/CVE-2007-6199/

[4] https://www.cvedetails.com/cve/CVE-2007-4091/

**IRC**

[1] http://dev.bukkit.org/media/images/34/653/irc-log.png

https://tools.ietf.org/html/rfc2813

https://tools.ietf.org/html/rfc7194

http://www.speedguide.net/port.php?port=194

https://freenode.net/irc\_servers.shtml

**FTP**

[1] https://tools.ietf.org/html/rfc114

[2] https://tools.ietf.org/html/rfc765

[3] https://tools.ietf.org/html/rfc959

[4] https://tools.ietf.org/html/rfc2428

[5] https://tools.ietf.org/html/rfc2228

[6] https://tools.ietf.org/html/rfc2577

**LDAP**

[1] https://www.ietf.org/rfc/rfc1777.txt

[2] https://www.ietf.org/rfc/rfc2251

[3] https://www.ietf.org/rfc/rfc4511

[4] https://www.owasp.org/index.php/Testing\_for\_LDAP\_Injection\_(OTG-INPVAL-006)

[5] https://www.ietf.org/rfc/rfc2829.txt

https://en.wikipedia.org/wiki/Lightweight\_Directory\_Access\_Protocol

https://www.owasp.org/index.php/LDAP\_injection

**RADIUS**

[1] https://tools.ietf.org/html/rfc2865

[2] https://tools.ietf.org/html/rfc2866

[3] https://tools.ietf.org/html/rfc2058

[4] https://tools.ietf.org/html/rfc6613

[5] https://tools.ietf.org/html/rfc6614

http://www.juniper.net/techpubs/software/aaa\_802/sbrc/sbrc70/sw-sbrc-admin/html/Concepts2.html

https://en.wikipedia.org/wiki/RADIUS

http://books.gigatux.nl/mirror/wireless/0321202171/ch13lev1sec4.html

**GNutella**

[1] https://en.wikipedia.org/wiki/Gnutella

[2] https://en.wikipedia.org/wiki/Distributed\_hash\_table

[3] http://rfc-gnutella.sourceforge.net/

[4] https://www.cvedetails.com/cve/CVE-2001-1004/

[5] http://alumni.cs.ucr.edu/~csyiazti/courses/cs260-2/project/html/

**MySQL**

[1] https://dev.mysql.com/doc/workbench/en/images/wb-navigator-server-logs-error-log.png

[2] https://dev.mysql.com/doc/workbench/en/images/wb-audit-inspector-main.png

http://dev.mysql.com/doc/refman/5.7/en/introduction.html

https://www.mysql.com/customers/

**SVN**

http://serverfault.com/questions/728509/svn-over-ssl-many-reconnects-on-commit

https://subversion.apache.org/

https://wiki.archlinux.org/index.php/Subversion#To\_SSL\_or\_not\_to\_SSL.3F

https://www.cvedetails.com/vulnerability-list/vendor\_id-45/product\_id-20053/Apache-Subversion.html