RIS420 Assignment 1

Well known Ports

* FTP
* LDAP

Registered Ports

* RADIUS
* SVN

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 RFC 114, and originally ran on the model that predated the current TCP/IP suite. The updated version became RFC 765 in June 1980 which was when it migrated to TCP/IP, and further updated to its current documentation in October 1985 as RFC 959. 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 RFC 2428 alongside of support for IPv6. Other updates to FTP in the form of security extensions came in RFC 2228 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 RFC 2577, 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.’[1]

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. RFC 2228 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 RFC 2577 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 RFC 2428 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

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

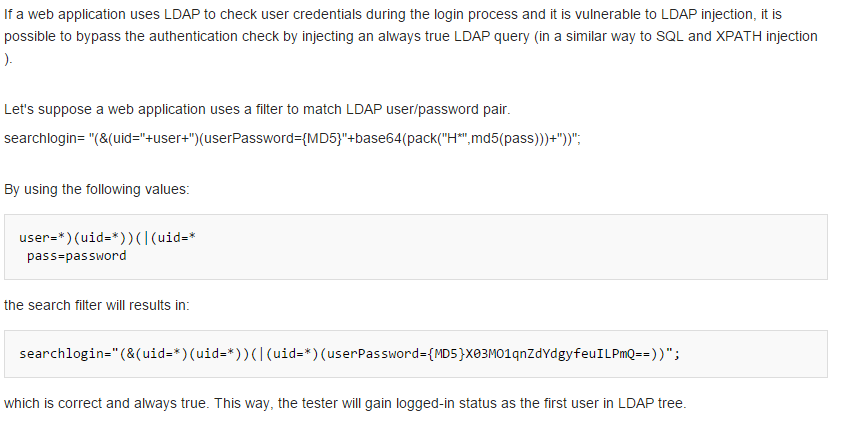
https://en.wikipedia.org/wiki/File\_Transfer\_Protocol

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 RFC 1777 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 RFC 2251, it added TLS (Transport Layer Security) to the protocol. The most recent update found in RFC 4511 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.

[2]

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, 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.

[3]

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. 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

https://www.ietf.org/rfc/rfc1777.txt

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

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

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

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

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 while Accounting is found in RFC2866. 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. 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 RFC 6613 and 6614, 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

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

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

https://tools.ietf.org/html/rfc2058

https://tools.ietf.org/html/rfc6613

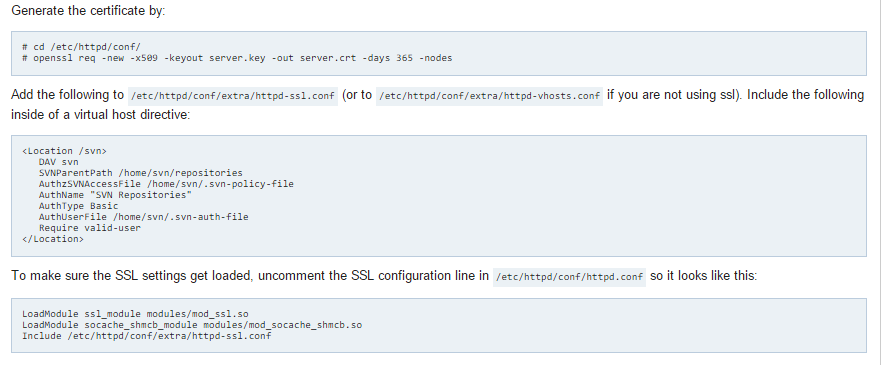
https://tools.ietf.org/html/rfc6614

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

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.



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:

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