Interested in learning more about security?

SANS Institute
InfoSec Reading Room

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

An Overview of Different Authentication Methods and Protocols

Authentication can be accomplished in many ways. The importance of selecting an environment appropriate Authentication Method is perhaps the most crucial decision in designing secure systems. Authentication protocols are capable of simply authenticating the connecting party or authenticating the connecting party as well as authenticating itself to the connecting party. This overview will generalize several Authentication Methods and Authentication Protocols in hopes of better understanding a few options that are availa...

Copyright SANS Institute
Author Retains Full Rights

AD

ForeScout

Richard Duncan
October 23, 2001

An Overview of Different Authentication Methods and Protocols

Introduction

Authentication can be accomplished in many ways. The importance of selecting an environment appropriate Authentication Method is perhaps the most crucial decision in designing secure systems.

Authentication protocols are capable of simply authenticating the connecting party or authenticating the connecting party as well as authenticating itself to the connecting party.

This overview will generalize several Authentication Methods and Authentication Protocols in hopes of better understanding a few options that are available when designing a security system.

Passwords

Passwords are the most widely used form of authentication. Users provide an identifier, a typed in word or phrase or perhaps a token card, along with a password. In many systems the passwords, on the host itself, are not stored as plain text but are encrypted. Password authentication does not normally require complicated or robust hardware since authentication of this type is in general simple and does not require much processing power. Password authentication has several vulnerabilities, some of the more obvious are:

Password may be easy to guess.
Writing the password down and placing it in a highly visible area.
Discovering passwords by eavesdropping or even social engineering.

The risk of eavesdropping can be managed by using digests for authentication. The connecting party sends a value, typically a hash of the client IP address, time stamp, and additional secret information. Because this hash is unique for each accessed URI, no other documents can be accessed nor can it not be used from other IP address without detection. The password is also not vulnerable to eavesdropping because of the hashing. The system is, however, vulnerable to active attacks such as the-man-in-the middle attack.

One-time passwords

To avoid the problems associated with password reuse, one-time passwords were developed. There are two types of one-time passwords, a challenge-response password and a password list.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The challenge-response password responds with a challenge value after receiving a user identifier. The response is then calculated from either the response value (with some electronic device) or select from a table based on the challenge.

A one-time password list makes use of lists of passwords which are sequentially used by the person wanting to access a system. The values are generated so that it is very hard to calculate the next value from the previously presented values. For example, the S/Key system calculates values x_i starting from initial value R: $x_1=f(R)$, $x_2=f(f(R))$, ..., $x_n=f(x_{n-1})$. The f() is chosen so that f^{-1} is very difficult. First the x_n is used, then the x_{n-1} is used. [Applied Cryptography Second Edition: protocols, algorithms, and source code in C, p. 53].

It is important to keep in mind that Password systems only authenticate the connecting party. It does not provide the connecting party with any method of authenticating the system they are accessing, so it is vulnerable to spoofing or a man-in-middle attack.

Public-key cryptography

Public key cryptography is based on very complex mathematical problems that require very specialized knowledge. Public key cryptography makes use of two keys, one private and the other public. The two keys are linked together by way of an extremely complex mathematical equation. The private key is used to decrypt and also to encrypt messages between the communicating machines. Both encryption and verification of signature is accomplished with the public key.

The advantage of public-key cryptography is that the public key is readily available to the public. In fact, public-keys are often published to public directories on the Internet so that they can be easily retrieved. This simplifies key-management efforts.

The integrity of the public key is of the utmost importance. The integrity of a public key is usually assured by completion of a certification process carried out by a certification authority (CA). Once the CA has certified that the credentials provided by the entity securing the public key are valid, the CA will digitally sign the key so that visitors accessing the material the key is protecting will know the entity has been certified.

Basically, the public-key authentication process includes the following:

> Client selects some random numbers and sends the results to the server as a message: Message 1.
>
> The server then sends different random numbers back to the client based on Message 1.
>
> The Clients then computes the new value and sends Message 2 to the server.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

The Server then uses the clients public key to verify that the values returned could have only been computed using the private key.

This authenticates the client to the server. If the client wants to authenticate the server, the same procedure is repeated with changed roles.

Zero-knowledge proofs

Zero-knowledge proofs make it possible for a Host to convince another Host to allow access without revealing any "secret information". The hosts involved in this form of authentication usually communicate several times to finalize authentication.

The client will first create a random but difficult problem to solve and then solves it using information it has. The client then commits the solution using a bit-commitment scheme and then sends the problem and commitment to the server.

The server then asks the client to either prove that the problems are related or open the committed solution and prove that it is the solution. The client complies with the request.

Typically, about ten successful exchanges will be required to take place before the authentication process is complete and access is granted.

The zero-knowledge proof can be made to be non-interactively. In this instance only one message from client to server is needed. This method utilizes a one-way hash function where the committing answers are based on the output of that hash function. The number of proofs needed is generally larger (64 or more), to avoid brute-force attacks.

The zero-knowledge proof of identity has it share of problems. Perhaps the most vulnerable one is that while Host A thinks he is proving his identity to Host B, it is possible for Host B to simultaneously authenticate to a third party, Host C, using Host A's credentials.

Digital Signatures

In many instances it is not necessary to authenticate communicating parties; for instance when downloading application updates or patches from the Internet. From a security point-of-view, the server does not need to screen who is downloading the software. The user downloading the software does not necessarily care what particular server it is downloading from. However, the user may want to be assured that the downloadable data is genuine and not a Trojan Horse or other malicious or invalid information. In this instance a digital signature would best serve to authenticate the downloadable data.

A digital signature is a digest calculated from a signed document (typically a one-way hash function) which is then signed (encrypted with private key). The client verifies the digest signature by decrypting it with the server's public key and compares it to the digest

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

value calculated from the message received. The signature can also be used by the server to verify data the client is sending.

Widely used Authentication Protocols

In this section we will briefly examine some of the more commonly used protocols used to address security issues within open networks. Authentication is the first and most important line of defense in a system of trusted and open networks.

Secure Sockets Layer

Secure Sockets Layer (SSL), developed by Netscape Communications, provides a secure method of communication for TCP connections, especially for HTTP connections.

SSL work in this manner: after a TCP connection is established, the client sends a client hello message to which the server responds with a server hello message. The hello messages establish connection attributes which include the protocol version, a session identifier, the cipher suite used, and the compression method in addition to random values for both the server and the client.

After the hello messages are exchanged, the server will send its certificate. When the server has been authenticated, depending on the cipher suite used, the server may then request a certificate from the client. After receiving the client hello, the server instructs the client to start using encryption and finishes the initial handshake. The application transfer can now take place.

When the client and the server decide to resume a previous session or duplicate an existing session, only the hello messages are exchanged. If the server does not find a matching session identifier, it will assume the connection is a new one. The advantage of resuming previous session is that it saves processing time, which may have a considerable effect on server performance.

IP SEC

The IP Authentication header provides strong authentication and integrity for IP datagrams. Depending on the signing algorithm used, it may also provide non-repudiation, excluding those fields that are changed during transmit, like hop count or time to live. The authentication header has fields for the next header, payload length, security parameters index (SPI: identifies security association (SA) between two hosts), sequence number, and authentication data. [Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825,]

The authentication is transport-protocol independent, so there may be data from more than one different protocol, for instance TCP and UPD. The authentication data is calculated with a message digest algorithm.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

To avoid replay attacks, the 32-bit sequence number is not allowed to wrap around; one must establish a new SA and generate new keys. This happens once in 232 packets so, if 1460 byte TCP segments are transferred one can transfer 5.7 TB of data using one SA.

Secure Shell

Secure Shell (SSH) is a protocol for providing secure remote login and other secure network services over an insecure network.

With SSH (version 2) each host has a host key, during the connection establishment the client can verify he is talking to the right server. The server keys can be stored locally on the clients or they may be distributed by using a key distribution protocol. [Ylonen, T., Kivinen, T.,, Saarinen, M., Rinne, T., Lehtinen, S., "SSH Protocol Architecture"]

After a reliable byte stream is established between the client and the server, host authentication takes place using the transport layer functions. Both ends send version identification.

The key exchange begins with both the client and server sending a key exchange initialization packet. The initialization packet contains a list of algorithms for key exchange, keys, encryption, MAC, and the level of compression supported. The server and client may negotiate a different set of algorithms for each direction of data flow. For each category, the best algorithm is chosen that both the client and server support.

Kerberos

Kerberos authentication was developed at the Massachusetts Institute of Technology (MIT). There are two main components: a ticket, which is used for user authentication and securing data, and an authenticator that is used to verify that the user is the same user to whom the ticket was initially granted. When a user logs into a system, the system connects to the Kerberos server where it retrieves a session key to be used between the user and the ticket granting service (TGS). This is encrypted with a key based on the user's password. If the user provides the right password the end system is able to decrypt the session key. After this is done, the user password is erased from memory to avoid being compromise. The ticket (Ticket granting ticket: TGT) expires after a set amount of time.

When a user wants to connect to a service to which he does not already have a ticket, the user connects to the TGS and gets a ticket that can only be used to access the particular service the ticket was granted for. The user can now connect through an encrypted channel to the server. After the ticket expires, the user must request a new one from the TGS.

The major issue with Kerberos is its scalability. The Kerberos server must store secret keys for each of the users and each of the TGSs. Kerberos can get very complex in

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

enterprise implementations where trust relationship need to be in place between multiple organizations.

Conclusion

User authentication can be handled using one or more different authentication methods. Some authentication methods such as plain password authentication are easily implemented but are in general weak and primitive. The fact that plain password authentication it is still by far the most widely used form of authentication, gives credence to the seriousness of the lack of security on both the Internet and within private networks.

Other methods of authentication, that may be more complex and require more time to implement and maintain, provide strong and reliable authentication (provided one keeps its secrets secret, i.e. private keys and phrases).

That being said, one of the key factors to be considered in determining which method of authentication to implement is usability. The usability factor cannot be ignored when designing authentication systems. If the authentication methods are not deemed usable by those forced to utilize them, then they will avoid using the system or persistently try to bypass them. Usability is a key issue to the adoption and maintenance of a security system.

References:

Nielsen, J., "Usability Engineering", AP Professional, 1993.

Schneider, B., "Applied Cryptography Second Edition: protocols, algorithms, and source code in C", John Wiley & Sons, Inc., 1996.

Eastlake, D., Kaufman, C., "Domain Name System Security Extensions", RFC 2065, IETF, January 1997. ftp://ftp.isi.edu/in-notes/rfc2065.txt

Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825, NRL, August 1995. ftp://ftp.isi.edu/in-notes/rfc1825.txt

Atkinson, R., "IP Authentication Header", RFC 1826, NRL, August 1995. ftp://ftp.isi.edu/in-notes/rfc1826.txt

Ylonen, T., Kivinen, T.,, Saarinen, M., Rinne, T., Lehtinen, S., "SSH Protocol Architecture" Study Guide

Alvaro Retana, Don Slice, Russ White, “Advanced IP Network Design” Cisco Press, 1999

Martin Krist, “Standard for Auditing Computer Applications” Auerbach Inc, 1999

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

© SANS Institute 2002, Author retains full rights.

© SANS Institute 2002, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: September 4th, 2015

Upcoming SANS Training

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

SANS Crystal City 2015	Crystal City, VAUS	Sep 08, 2015 - Sep 13, 2015	Live Event
SANS Network Security 2015	Las Vegas, NVUS	Sep 12, 2015 - Sep 21, 2015	Live Event
SANS Seoul 2015	Seoul, KR	Sep 14, 2015 - Sep 19, 2015	Live Event
SANS Tallinn 2015	Tallinn, EE	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Baltimore 2015	Baltimore, MDUS	Sep 21, 2015 - Sep 26, 2015	Live Event
Data Breach Investigation Summit & Training	Dallas, TXUS	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS Perth 2015	Perth, AU	Sep 21, 2015 - Sep 26, 2015	Live Event
SANS ICS Amsterdam 2015	Amsterdam, NL	Sep 22, 2015 - Sep 28, 2015	Live Event
SANS Bangalore 2015	Bangalore, IN	Sep 28, 2015 - Oct 17, 2015	Live Event
SANS Seattle 2015	Seattle, WAUS	Oct 05, 2015 - Oct 10, 2015	Live Event
SANS DFIR Prague 2015	Prague, CZ	Oct 05, 2015 - Oct 17, 2015	Live Event
SOS: SANS October Singapore 2015	Singapore, SG	Oct 12, 2015 - Oct 24, 2015	Live Event
SANS Tysons Corner 2015	Tysons Corner, VAUS	Oct 12, 2015 - Oct 17, 2015	Live Event
SANS Gulf Region 2015	Dubai, AE	Oct 17, 2015 - Oct 29, 2015	Live Event
SANS Tokyo Autumn 2015	Tokyo, JP	Oct 19, 2015 - Oct 31, 2015	Live Event
SANS Cyber Defense San Diego 2015	San Diego, CAUS	Oct 19, 2015 - Oct 24, 2015	Live Event
SANS Sydney 2015	Sydney, AU	Nov 09, 2015 - Nov 21, 2015	Live Event
SANS South Florida 2015	Fort Lauderdale, FLUS	Nov 09, 2015 - Nov 14, 2015	Live Event
SANS London 2015	London, GB	Nov 14, 2015 - Nov 23, 2015	Live Event
Pen Test Hackfest Summit & Training	Alexandria, VAUS	Nov 16, 2015 - Nov 23, 2015	Live Event
SANS Hyderabad 2015	Hyderabad, IN	Nov 24, 2015 - Dec 04, 2015	Live Event
SANS Cape Town 2015	Cape Town, ZA	Nov 30, 2015 - Dec 05, 2015	Live Event
SANS San Francisco 2015	San Francisco, CAUS	Nov 30, 2015 - Dec 05, 2015	Live Event
Security Leadership Summit & Training	Dallas, TXUS	Dec 03, 2015 - Dec 10, 2015	Live Event
SANS Milan 2015	OnlineIT	Sep 07, 2015 - Sep 12, 2015	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced