Communication Protocols
and Internet Architectures

Harvard University

Lecture #11

Instructor: Len Evenchik
cs40@evenchik.com or evenchik@fas.harvard.edu

ALIGHLSOD1701

Lecture Agenda

- Course Logistics
- Q&A and Topics from Last Week
- Network and System Security (part 2)
- Encryption
- Hashing
- Authentication
- Digital Signatures
- Firewalls and VPN
- Website Security
- One Minute Wrap-Up

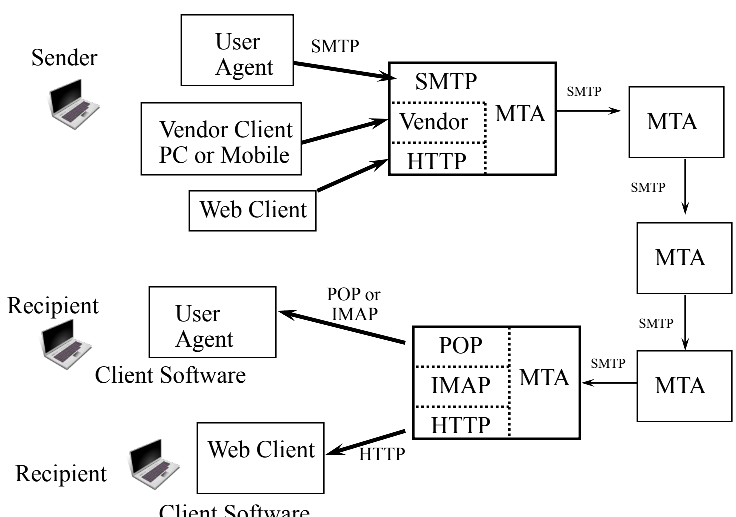
Course Logistics

Course Logistics

- Midterm – Please contact the instructor if you have questions about the midterm.
- Final Exam – Please check the weekly course information sheet for detailed information on the final.
- Upcoming Guest Lectures
- Homework #4 has been posted.
- Always check the weekly course information sheet for any updated schedule information for section meetings.
- **Please submit a one minute wrap-up each week. Thank You!**

Q&A
Topics from Last Week

Mail System Architecture
Protocols for Client to MTA



Simplified SMTP Procedure

>>> HELO Alpha.EDU
250 Beta.COM Hello Alpha.EDU, pleased to meet you
>>> MAIL FROM:<Smith@Alpha.EDU>
250 OK
>>> RCPT TO:<Jones@Beta.COM>
250 OK
>>> RCPT TO:<Green@Beta.COM>
550 No such user here
>>> DATA
354 Start mail input; end with <CRLF>**.**<CRLF>
>>> headers go here
>>>
>>> blah, blah, message body goes here
>>> blah, blah, more message
>>> <CRLF>**.**<CRLF>
250 OK
>>> QUIT
221 Beta.COM delivering mail for you

Example: Comer Textbook

Sending Email (b)

(Simple example using Telnet connection)

ls03:~ %
ls03:~ % telnet mail.dce.harvard.edu 25
Trying 140.247.197.235...
Connected to mail.dce.harvard.edu (140.247.197.235).
Escape character is '^]'.
220 mail.dce.harvard.edu ESMTP
Exim Mon, 24 Oct 2017 18:25:54 -0500

HELO somemachine.edu
250
MAIL FROM:<le@harvard.edu>
250 <le@harvard.edu> is syntactically correct

RCPT TO:<cscie-40@mail.dce.harvard.edu>
250 <cscie40@mail.dce.harvard.edu> verified

DATA

Sending Email (c)

220 mail.dce.harvard.edu ESMTP Exim Mon, 24 Oct 2016 18:25:54 -0500
MAIL FROM:<le@harvard.edu>
250 <le@harvard.edu> is syntactically correct
RCPT TO:<csci-40@mail.dce.harvard.edu>
250 <csci-40@mail.dce.harvard.edu> verified
DATA
354 Enter message, ending with "." on a line by itself
From: Len at Lectern
To: The TAs in the course
Date: Wed, Dec 1, 1901
Re: Planning for the midterm

Dear TAs,

Should we include anything on the exam on this
new thing called a telephone?

... Len
.
250 OK id=1AOQ7C-0000CR-00

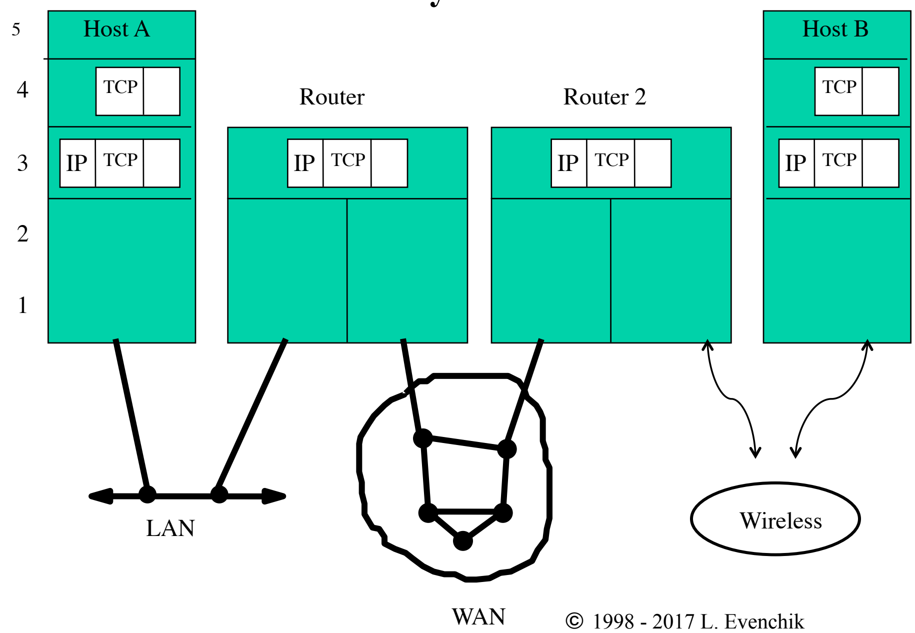
Sending Email (e)
Mail as Delivered (headers on)

Return-path: <le@harvard.edu>
Envelope-to: csci-40@mail.dce.harvard.edu
Delivery-date: Mon, 24 Oct 2017 18:31:09 -0500
Received: from ls03.fas.harvard.edu [140.247.34.xxx] (evenchik)
    by mail.dce.harvard.edu with smtp (Exim)
    for csci-40@mail.dce.harvard.edu
    id 1AOQ7C-0000CR-00; Mon, 24 Oct 2017 18:29:29 -0500
From: Len at Lectern
To: The TAs in the course
Date: Wed, Dec 1, 1901
Re: Planning for the midterm
Message-Id: <E1AOQ7C-0000CR-00@barkley.dce.harvard.edu>
Date: Mon, 24 Oct 2016 18:29:29 -0500

Dear TAs,
Should we include anything on the midterm on this
new thing called a telephone?
.. Len

Network and System Security

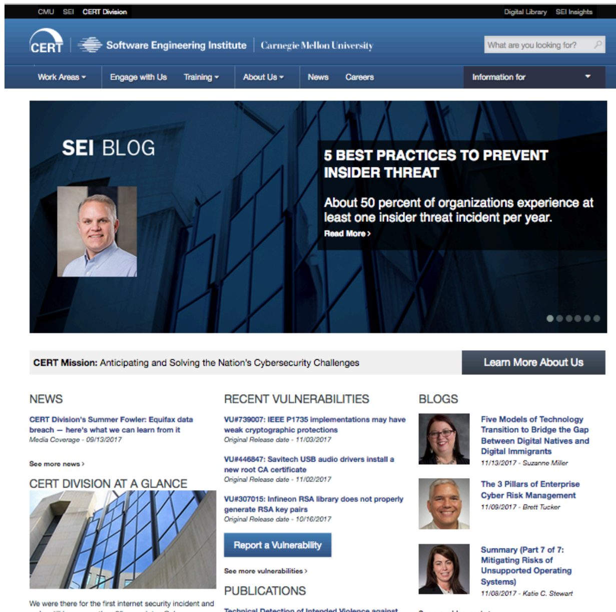
Where Should Security be Used in this Network?



Security Resources
No Single Resource is Enough

- CERT Coordination Center
  - 412-268-7090 (always have the current tel #)
  - www.cert.org
  - cert@cert.org
- US-CERT Coordination Center
  - 1-888-282-0870 (always have the current tel #)
  - http://www.us-cert.gov/
  - soc@us-cert.gov
- Your corporate IT group and legal department.
- IETF working groups, other well known security organizations
- Your ISP
- Your firewall, router and other equipment vendors

WWW.CERT.ORG



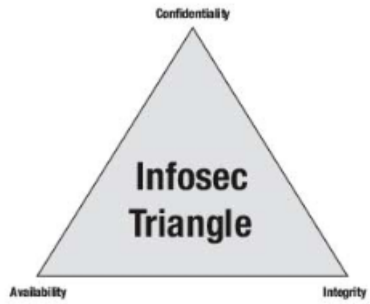
hik

The Basics

Infosec Triangle or CIA Triad

This is a common business oriented approach to understanding security; we will complement this with a more technical framework.

- Confidentiality
- Availability
- Integrity



The Most Common Passwords

COMMON PASSWORDS

The Worst Passwords of 2012, including their current ranking and any changes from the 2011 list:

1. password (Unchanged)
2, 123456 (Unchanged)
3. 12345678 (Unchanged)
4. abc123 (Up 1)
5. qwerty (Down 1)
6. monkey (Unchanged)
7. letmein (Up 1)

Most Common & Worst Passwords of 2014

Rank	Password	Change from 2013
1	123456	Unchanged
2	password	Unchanged
3	12345	Up 17
4	12345678	Down 1
5	qwerty	Down 1
6	123456789	Unchanged
7	1234	Up 9
8	baseball	New
9	dragon	New
10	football	New
11	1234567	Down 4
12	monkey	Up 5
13	letmein	Up 1
14	abc123	Down 9
15	111111	Down 8
16	mustang	New
17	access	New
18	shadow	Unchanged

1. **123456**
I can't be bothered to take even the most basic step to protect my personal information. Seriously, just go ahead and take it.

2. **password**
I failed to understand the question.

3. **12345678**
I tried "123456," but the computer said I had to use at least eight characters.

4. **qwerty**
Aren't I clever? My password is written right there on the keyboard.

5. **abc123**
I'm a fan of the Jackson Five.

These examples are a few years old but things have not improved much.
Source of tables – trade press

In Summary, Security Requires:

- Hardware
- Software
- Written Procedures and Processes
- People educated on what security means and how to properly do it.

Security is a system issue which requires all of the above, but without a doubt, people who understand and care about the issues are the most important element.

A More Technical Approach to Security

Structured way to Think about Security: Five Important Elements

- Privacy and confidentiality
- Authentication
- Authorization
- Integrity
- Nonrepudiation

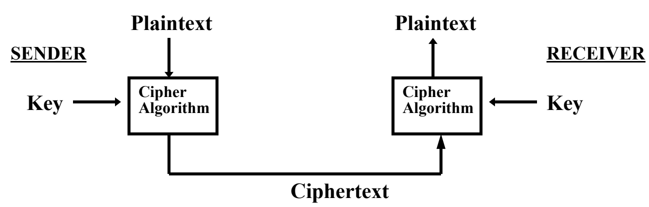
Cryptography

Approaches to Cryptography

- Symmetric cryptography
  - Shared secret key system
  - Same key used to encrypt and decrypt messages
  - Key length determines the "strength" of encryption
  - Key management is difficult
  - Examples are 3DES, IDEA, RC4 and AES
- Asymmetric cryptography (called Public Key)
  - Key pair - one public, one private
  - Data encrypted by one key must be decrypted by the other key
  - Examples are Diffie-Hellman (1976) and RSA (1978)

Advanced Encryption Standard (AES)

- Advanced Encryption Standard (AES) is a shared secret key encryption scheme.
- Provides Confidentiality for your data.
- Developed by NIST using a public evaluation process (15 candidates.) AES published in Nov. 2002.
- AES is a symmetric block cipher encryption algorithm
- AES supports key lengths of 128, 192 and 256 bits. Why different sizes?



Public Key Encryption

- Key distribution and management has always been the weak link in shared key systems
- Public key systems solve this problem by having two keys, a “private key” and a “public key”
- Users publish their “public key” and other people can send them encrypted messages by using this specific “public key”
- The “magic” that makes this possible is the use of complex algorithms that make it “very hard” to guess a private key even if you know the public key and the underlying algorithm.
- This approach is used extensively today for web traffic, email, other applications.

Public Key Algorithms

- Key distribution and management has always been the weak link in shared key systems
- Public key systems solve this problem by being able to publish a "public key"
- Algorithm must provide the following functionality:
  - D(E(P)) = P
  - It is very difficult to deduce D from E
  - E cannot be broken by a chosen plaintext attack
- Appropriate algorithms are based on hard problems such as taking the log of a number or factoring large numbers.

ssh-keygen

```
cscie40@courses (~): ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the
                key (/home/web/c/s/cscie40/.ssh/id_rsa): test4
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in test4.
Your public key has been saved in test4.pub.
The key fingerprint is:
25:dc:21:44:0b:bd:25:76:67:83:1e:86:0c:fa:ce:20 cscie131b@barkley
cscie40@dcepea (~):
```

Private Key

cmd **(~): cat test4**
-----BEGIN RSA PRIVATE KEY-----


```
MIICXQIBAAKBgQC/aSKmm6VdcqL6IQzK81998Ac8Coes/V2l4KGZItcSYboSE1e7
s3RVssdY9Xqol1cVEXhhQ/SnzcQhKti4CrC6dxyOwpVDdDSo7ZW8LWRg2Gw1jFoU
KDUeLsEqbzmEBdteuvixbUITaMGqjtKnjdFo8fi3Y7MW5sS2ZvdpweSkWwIBIwKB
gQC58RpYtHTBLYhgsmQy3cp6VuJ01wd02N6hI1snC+beqBPXC3nMR+1akGnhY353
43kqaL4VV/T6x+MY58s2cMjt9/5Llecfpt/uW53U56TZwniPgAQEiQe5nPowdzNZ
vs0QoVOpHUWPyvjTPAZVcEm68BaYI6FlESYdpOvwWkDlawJBAOujhwyWnX3u47po
VGJnfHNhVsO8wzsA4U26heTVE8nxgRwI0vsn4REJRNDByXqJ6pyLJHEVq6Y9lCnJ
QnwJYmkCQQDP80FugasGuZsgKQygS6ngndKXqD0y95RlFCda+t8krVuPdBnE6S1h
Iappr/6YKMVtVlCv0aFUhYtAlaDK1LAjAkEA13DwgIm0kGVifotF1k/8wUMNf8KG
nFiTekQipVUZaC1C182Nsm2akzusoZs73b/scd5NNDER9xO6qdyUjql+iwJBAIin
Kv95yCj9oHQ4O39HqiXkDgvjle5K7Hzv/JrfX2/fopF4LjDxAJBJUrp698MTemxr
6+FAnTeK9RvQCpPq2iUCQQDWkdkUUcmTuF15zKZ/M+5mUwfdpvErt7FICBeQl4X8
iak2TSIoZ1uBUq4YUdf38oCJX+QJVESdi8PovTVdUii3
```


-----END RSA PRIVATE KEY-----
cmd **(~):**

Public Key

cmd **(~): cat test4.pub**


```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIE
Av2kippulXXKi+iEMyvNfffAHPAqHrP1dpeCh
mSLXEmG6EhNXu7N0VbLHWPV6qJdXFRF4
YUP0p83EISrYuAqwunccjsKVQ3Q0qO2VvC1
kYNhsNYxaFCg1Hi7BKm85hAXbXrr4sW1CE2
jBqo7Sp43RaPH4t2OzFubEtmb3acHkpFs=
```


cmd **(~):**

Asymmentric Cryptography
(aka Public Key Cryptography)

Public Key Encryption

- Key distribution and management has always been the weak link in shared key systems
- Public key systems solve this problem by having two keys, a “private key” and a “public key”
- Users publish their “public key” and other people can send them encrypted messages by using this specific “public key”
- The “magic” that makes this possible is the use of complex algorithms that make it “very hard” to guess a private key even if you know the public key and the underlying algorithm.
- This approach is used extensively today for web traffic, email, other applications.

What Do We Want to Accomplish?

- How can we use Public/Private Key encryption to send a secure message that only the recipient can read?
- How can we use Public/Private Key to authenticate the sender of the message, and also encrypt the message so that only the recipient can read it?
- Given the relative slowness of Public Key versus symmetric (shared) key encryption, how do we use a combination of both types of encryption to quickly and efficiently send a secure authenticated message.

Public Key Encryption (1)

Assume that you want to send a private message to someone that only the recipient can read.

But note that public key encryption algorithms are much slower than symmetric key algorithms.

Therefore a combination of the two cryptographic approaches are used by most systems:

- Sender does….
- Recipient does….

Let's fill in the details…..

Public Key Encryption (2)

Assume that you want to send a private message to someone that only the recipient can read..

But note that public key encryption algorithms are much slower than symmetric key algorithms.

Therefore a combination of the two cryptographic approaches are used by most systems:

- Sender creates a session key (secret AES key)
- Sender encrypts message with that session key
- Sender then encrypts session key with recipient's public key
- Sender sends encrypted key and encrypted message to recipient.
- Recipient decrypts session key with private key
- Recipient then decrypts message using session key

Public Key Encryption

What functionality is NOT provided by the procedures we just described (on the previous slide)?

Public Key Encryption (3)

Assume now that you want to send a private message **and** authenticate the sender's identity.

Add a few steps to the procedure:

- Sender creates a session key
- Sender encrypts message with session key
- Sender encrypts session key with recipient's public key
- Sender encrypts the key again with the sender's private key. (This means the key is encrypted twice.)
- Sender sends encrypted key and encrypted message

- *What does the Recipient do?*

Hashing and Message Digests

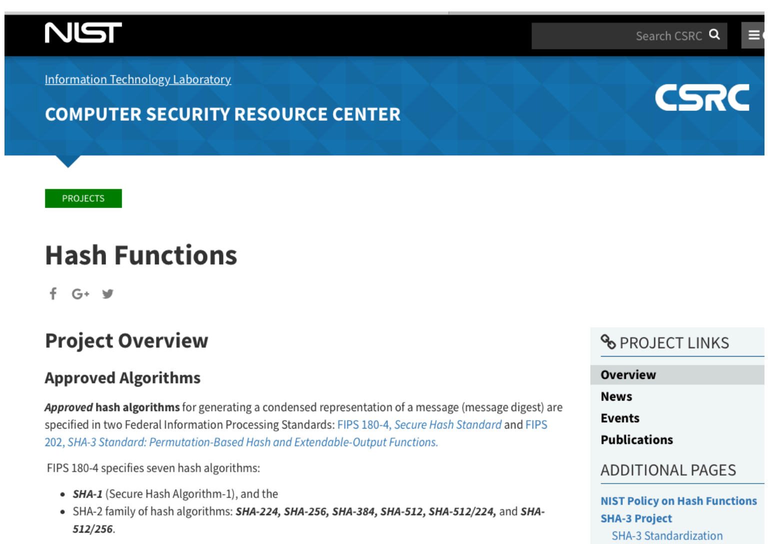
Hashing Functions and Message Digests

- Hash functions take an arbitrarily long piece of plaintext and compute from it a fixed length string.
- Hash functions are based on the fact that there are mathematical transformations that are easy to do but very, very hard to undo.
  - In mathematical terms y=f(x)
  - Given f and x, it is very easy to compute y
  - Given f and y, it is very hard to compute x
- Common message digests are 128 bits or longer
- **Hash functions can show that a message has not changed, but they do not provide confidentiality.**

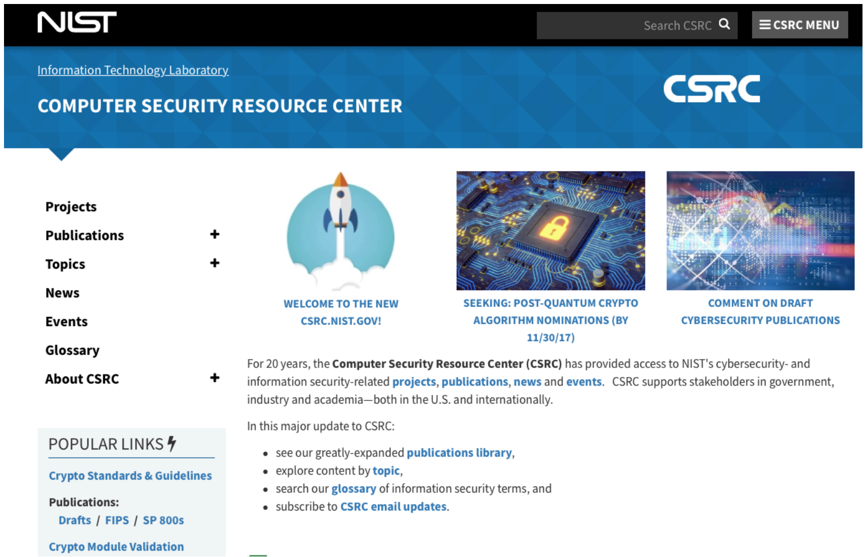
Hashing Functions and Message Digests (2)

- MD5 was the 5th hash function designed by Ron Rivest (1992). Security issues are well known with MD5 and it is no longer considered secure. However it is still used in some systems. See the CERT notes about this.
- Although it is still used, SHA-1 has been deprecated by NIST as of Jan. 2014. See RFC 6194 and the following:
  - http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf
  - http://googleonlinesecurity.blogspot.com/2014/09/gradually-sunsetting-sha-1.html
- SHA-2 (SHA-256) and SHA-3 are the current hash functions that have been standardized by NIST. See the NIST website:
  https://csrc.nist.gov/projects/hash-functions
- Remember, Hash functions do not provide confidentiality.

https://csrc.nist.gov/projects/hash-functions



https://csrc.nist.gov



Print of "testfile1"

cmd (~): **cat testfile1**
this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
cmd (~):

SHA-1 of a file called “testfile1”

cmd (~): **cat testfile1**
this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
cmd (~):

cmd (~): **sha1 testfile1**
sha1 (testfile1) = 88a5b867c3d110207786e66523cd1e4a484da697
cmd (~):

NOTE THAT WE ARE USING SHA-1 ONLY AS A SIMPLE EXAMPLE SINCE IT IS AVAILABLE AT THE COMMAND LINE.
IT IS NO LONGER SECURE!

Comparison of “testfile1” and “testfile2”
Note the small difference on the line with “Hello World”

cmd (~): **cat testfile1**
this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
cmd (~):

cmd (~): **cat testfile2**
this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World !
This is line five (5) of this file.
cmd (~):

SHA-1 Comparison for files "testfile1" and "testfile2"

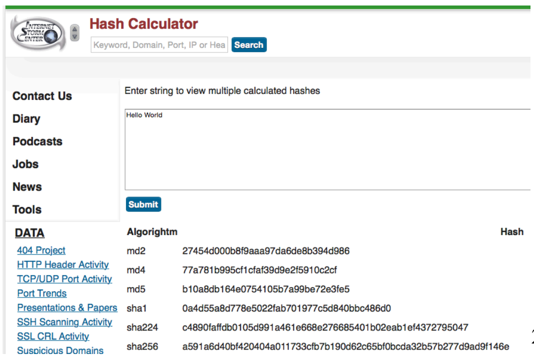
cmd (~): **SHA1 testfile1**
SHA-1 testfile1) = 88a5b867c3d110207786e66523cd1e4a484da697
cmd (~):

cmd (~): **SHA-1 testfile2**
SHA-1 (testfile2) = 874945e767b56391e8234780ce1d5150c11d9060
cmd (~):

NOTE THAT WE ARE USING SHA-1
ONLY AS AN SIMPLE EXAMPLE.
IT IS NO LONGER SECURE!

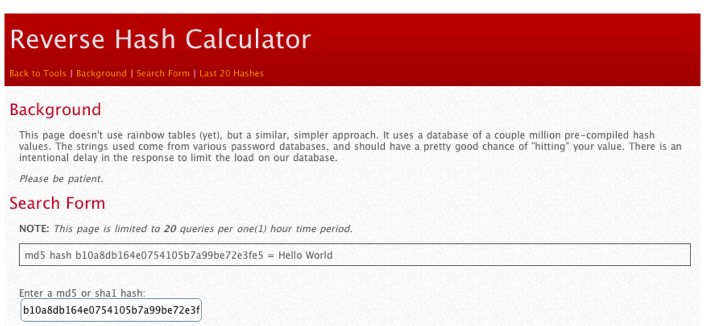
Online Hashing Calculator

- There are many online hash calculators that demonstrate how hash functions work.
- We will take a look at https://isc.sans.edu/tools/md5.html but please note that we cannot vouch for the cryptographic correctness of this implementation
- There are also reverse hash calculators on the net.



Online Reverse Hash Calculator

- A reverse hash calculator does just what is sounds like it does.
- We'll take a look at https://isc.sans.edu/tools/reversehash.html
- It is critically important that you understand that every security tool and system has limitations and you need to understand the details in order to use them properly.

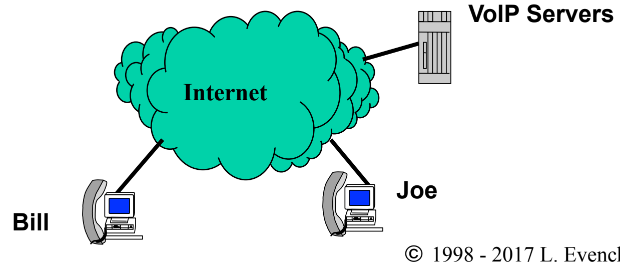


User Authentication via Hash Functions

Using VoIP System as an Example

User Authentication in VoIP/SIP

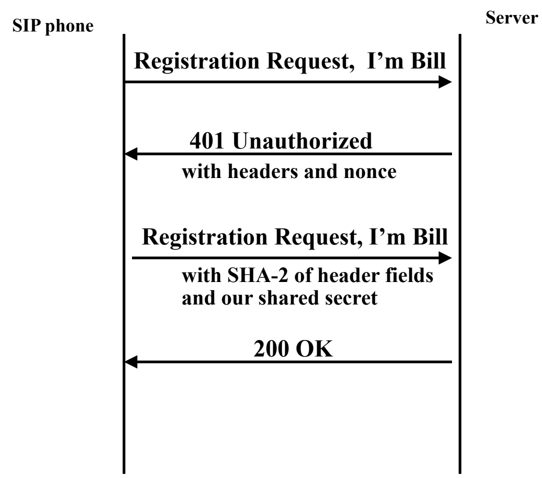
- The question is: How does a SIP VoIP server or system know that you are who you say you are? If a user is not authenticated, it would be easy for anyone to say that they are bill@harvard.edu and get that user's telephone calls.
- To answer this, first consider how this is done for your home telephone service (POTS), your cell phone, and the process of logging into your company mail server.



User Authentication

- The question is: How does a remote server know that you are who you say you are? If a user is not authenticated, it would be easy for anyone to say that they are bill@harvard.edu and get that user's telephone calls, or access to any other type of service.
- The name for this is user authentication and and it requires that the two parties in the communication (VoIP phone and the VoIP server) know a shared secret, but the secret should never be sent as clear text over the net. The technique is called HTTP Digest Authentication.
- The SHA-2 (or other hash) of the combination of the user name, shared secret, realm, and nonce (plus some other fields) is computed, sent, and then compared to the expected value to authenticate the user. The nonce provides protection against later replay.
- Let's study at an example using a VoIP SIP phone.

SIP VoIP User Authentication and Registration



This is an abridged trace of the packet flows

Authentication to VoIP Proxy Server (Step 1)

Session Initiation Protocol
Request-Line: REGISTER sip:siplearn.com:5060 SIP/2.0
Method: REGISTER
Message Header
Via: SIP/2.0/UDP
140.247.250.181;branch=z9hG4bKf7f8d7477263E836
Transport: UDP I'M BILL
Sent-by Address: 140.247.250.181
Branch: z9hG4bKf7f8d7477263E836
From: "Bill at ext 6003" <sip:bill@siplearn.com>;tag=8F21…
To: <sip:bill@siplearn.com>
CSeq: 1 REGISTER
Call-ID: bc8e2e39-68f1d8c0-b947fe7b@140.242.250.181
Contact: <sip:bill@140.247.250.181>……
Contact Binding: <sip:bill@140.247.250.181>;
methods="INVITE…..
etc…
etc…

abridged trace

401 Unauthorized (Step 2)

Session Initiation Protocol
Status-Line: SIP/2.0 401 Unauthorized
Message Header
Via: SIP/2.0/UDP 140.247.250.181; branch=xxx, received=140.247.250.181
Transport: UDP
Sent-by Address: 140.247.250.181
From: “Bill 6003" <sip:bill@siplearn.com>;tag=8F215C5A-D94BE88D
To: <sip:bill@siplearn.com>;tag=as47f93dba
Call-ID: bc8e2e39-68f1d8c0-b947fe7b@140.247.250.181
CSeq: 1 REGISTER
User-Agent: Asterisk PBX
Allow: register…
WWW-Authenticate:
Authentication Scheme: Digest
Algorithm: SHA-2
Realm: "siplearn.com"
Nonce Value: "0810d7034435aed35c"
Content-Length: 0

NONCE

abridged trace

Authorization (Registration) with Response to Challenge (Step 3)

Session Initiation Protocol
Request-Line: REGISTER sip:siplearn.com:5060 SIP/2.0
Method: REGISTER
Message Header
Via: SIP/2.0/UDP 140.247.250.181;branch=z9hG4bK5149a9846EA080EF
From: “Bill 6003" <sip:bill@siplearn.com>;tag=8F215C5A-D94BE88D
To: <sip:bill@siplearn.com>
CSeq: 2 REGISTER
Sequence Number: 2
Call-ID, Contact, etc, etc, etc
Authorization:
Authentication Scheme: Digest
Username: “bill"
Realm: "siplearn.com"
Nonce Value: " 0810d7034435aed35c "
Authentication URI: "sip:siplearn.com:5060"
Digest Authentication Response:
"9d68372b3929befa2a2eeaa0dcbf03df"
Algorithm: SHA-2

I'M BILL

NONCE

SHA-2 OF ID, NONCE AND SECRET

abridged trace

Digital Signatures

Digital Signatures

- A digital signature should “prove” that a message came from a specific user (lets call them UserA) and that the message has not been changed.
- A digital signature does not encrypt the message.
- What is an example of why you might not want to encrypt the message or document, but still validate that it was from a specific user and that it had not changed
- One way to produce a digital signature
  – UserA computes a one-way hash function on the contents of the message…
  – ***.. Lets work out the details, we will need to use public key encryption and hashing***

Digital Signatures

- A digital signature should "prove" that a message came from a specific user (lets call them UserA) and the message has not changed
- One way to produce a digital signature
  - UserA computes a one-way hash function on the contents of the message
  - UserA encrypts the hash code using their private key
  - The encrypted hash code is appended to the message and the combination is sent to UserB
  - UserB computes the same hash function on the contents of the message
  - UserB then decrypts the received hash code with UserA's public key
  - If the hash codes match, the message came from UserA and the message was not changed in transit

Signing of "testfile1"

```
cmd (~): gpg --clearsign < testfile1

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.7

iQCVAwUBP8u0S+nwDzqNmKQTAQIzEAQAiYTHo
PS4GZMUjFyzItigG2nWXuI3867oYyvPp/D9q+jTR6O
PapnwowXpgqJIZn0mIuxMoTO0pSkygcC3ILqo0o4
W5z6BN8ykfdXoyDMCuh4+n133OgjjYS/lyIrq9org+
gEw9nn4Chyyq5LvbHwgo1B6fr1mI+HGi4P4PvwdCDM=
=ZQMq
-----END PGP SIGNATURE-----
cmd (~):
```

cat of signature file of "testfile1sign"

cmd (~): **cat testfile1sign**
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

this is a test file to be used in the networks and protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.7

iQCVAwUBP8u0S+nwDzqNmKQTAQIzEAQAiYTHo
PS4GZMUjFyzltigG2nWXuI3867oYyvPp/D9q+jTR6O
PapnwowXpgqJIZn0mIuxMoTO0pSkygcC3ILqo0o4
W5z6BN8ykfdXoyDMCuh4+n133OgjjYS/lyIrq9org+
gEw9nn4Chyyq5LvbHwgo1B6fr1mI+HGi4P4PvwdCDM=
=ZQMq
-----END PGP SIGNATURE-----
cmd (~):

Signature Verification

cmd (~): **gpg --verify testfile1sign**
gpg: Warning: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: Signature made Mon Dec 01 16:36:11 2013 EST
using RSA key ID 8D98A413
gpg: Good signature from "course test (test key for demo) <csci40@mail.dce.harvard.edu>"
cmd (~):

cat of signature file of "testfile1sign"

cmd (~): **cat testfile1sign**
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World
This is line five (5) of this file.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.7

iQCVAwUBP8u0S+nwDzqNmKQTAQIzEAQAiYTHo
PS4GZMUjFyzltigG2nWXuI3867oYyvPp/D9q+jTR6O
PapnwowXpgqJIZn0mluxMoTO0pSkygcC3ILqo0o4
W5z6BN8ykfdXoyDMCuh4+n133OgjjYS/lylrq9org+
gEw9nn4Chyyq5LvbHwgo1B6fr1ml+HGi4P4PvwdCDM=
=ZQMq
-----END PGP SIGNATURE-----
cmd (~):

cat of signature file of changed "testfile1"
Note the added character at "Hello World"

cmd (~):
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

this is a test file to be used in the networks and
protocols class...
abcdefghijklmnopqrstuvwxyz1234567890
Hello World ?
This is line five (5) of this file.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.0.7

iQCVAwUBP8u0S+nwDzqNmKQTAQIzEAQAiYT
HoPS4GZMUjFyzltigG2nWXuI3867oYyvPp/D9q+
jTR6OPapnwowXpgqJIZn0mluxMoTO0pSkygcC
3ILqo0o4W5z6BN8ykfdXoyDMCuh4+n133OgjjY
S/lylrq9org+gEw9nn4Chyyq5LvbHwgo1B6fr1ml
+HGi4P4PvwdCDM=
=ZQMq
-----END PGP SIGNATURE-----
cmd (~):

Failure of Signature Verification

```
cmd (~): gpg --verify testfile1badsign
gpg: Warning: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
gpg: Signature made Mon Dec 01 16:36:11 2013 EST
                    using RSA key ID 8D98A413
gpg: BAD signature from "course test (test key for demo)
                    <csci40@mail.dce.harvard.edu>"
cmd (~):
```

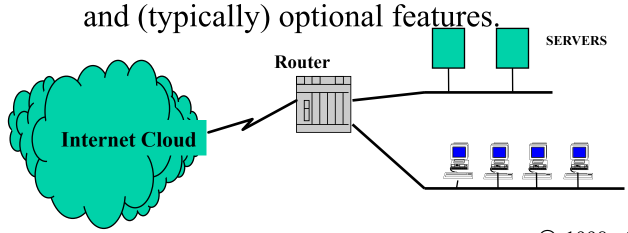

Digital Signatures

- A digital signature should "prove" that a message came from a specific user (lets call them UserA) and that the message has not changed
- One way to produce a digital signature
  - UserA computes a one-way hash function on the contents of the message
  - UserA encrypts the hash code using their private key
  - The encrypted hash code is appended to the message and the combination is sent to UserB
  - UserB computes the same hash function on the contents of the message
  - UserB then decrypts the received hash code with UserA's public key
  - If the hash codes match, the message came from UserA and the message was not changed in transit

Security Provided by Routers and Firewalls

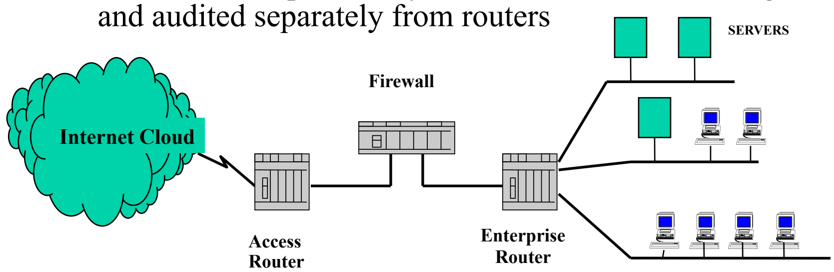
First Lets Talk About Router Based Security

- For many years routers have used access control lists (ACL) to filter packets based on one or more criteria:
  - Source and/or destination IP address
  - transport layer protocol (UDP vs TCP)
  - application protocol (SSH, SIP, HTTP, DNS, etc.)
  - protocol state information
  - plus other criteria
- Access lists in routers are difficult to maintain and are different than routing policies. Current routers now include "router based firewalls" and these are separate and (typically) optional features.



Basic Firewall Architecture

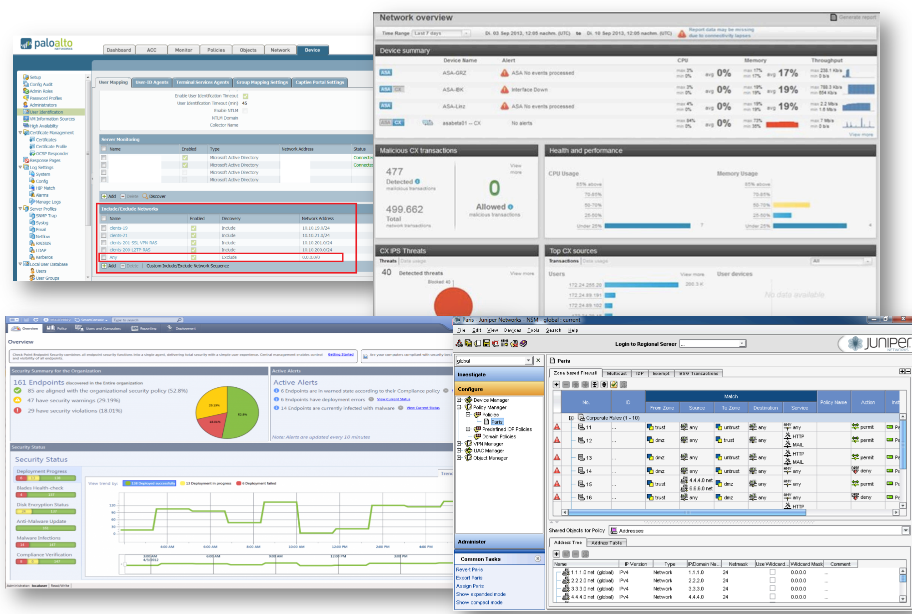
- Firewalls are dedicated network devices (or separate software) that isolate the external network from the internal/enterprise network.
- Firewalls are also used to isolate and manage different networks within the same enterprise. This is important since there are different types of users, each with different privileges and responsibilities.
- Division of responsibility: firewalls should be managed and audited separately from routers



Firewall Functionality (part one)

- Firewalls implement a security policy for an enterprise based on a defined set of access and other rules
- These rules are typically called Policies and are defined by a detailed ACL and policy configuration system. They are typically managed by a GUI.
- ACLs define what is allowed into the enterprise network from the outside world, as well as what is allowed out of the network. Deny should be the default policy.
- Firewalls can implement access security using many different technical approaches. Unfortunately, these approaches are presented in a confused and contradictory way by vendors
  – Packet based filter/forward decisions (too simple)
  – Stateful Inspection
  – Application Layer Gateways (ALG)
  – and many others

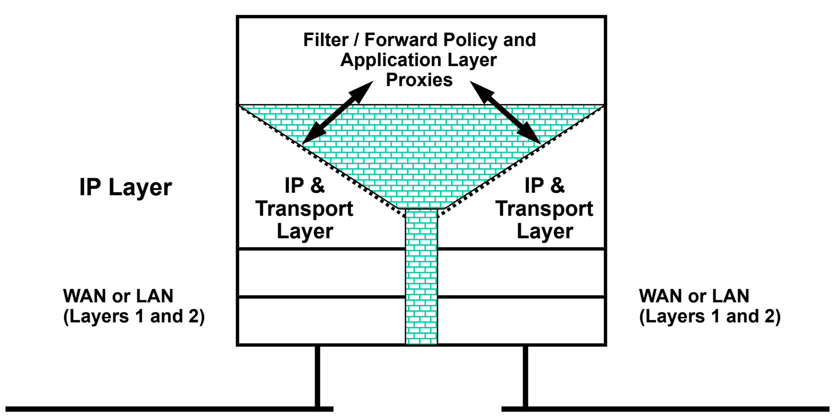
Common Firewall GUI Management Consoles



Firewall Functionality (part two)

- Firewalls access decisions can be based on:
  - Source and/or destination address, port info, etc.
  - The specific user
  - Type of pplication layer protocols (SSH, FTP, HTTP, SIP/VoIP, etc) and the user
  - Application layer URLs and MIME types, and Deep Packet Inspection
  - Major differences exist between inbound versus outbound traffic, etc. etc.
- Firewalls can also provide user authentication and VPNs:
  - User passwords or one time passwords with secure IDs or tokens
  - This functionality is in addition to the functionality noted above

Simplified Firewall Schematic

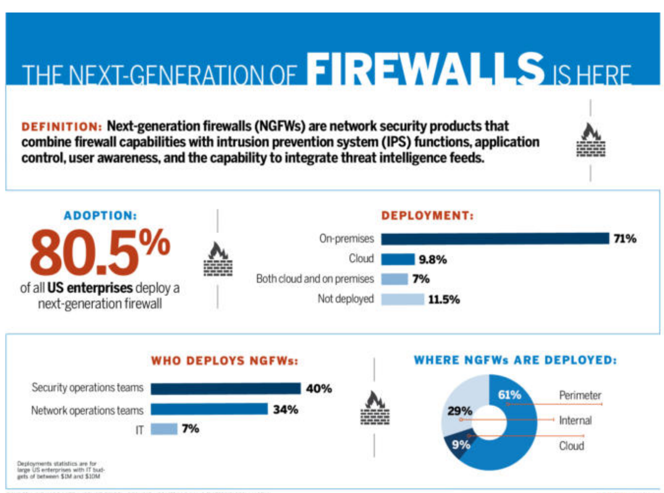


© 1998 - 2017 L. Evenchik

Firewall Functionality (part three)

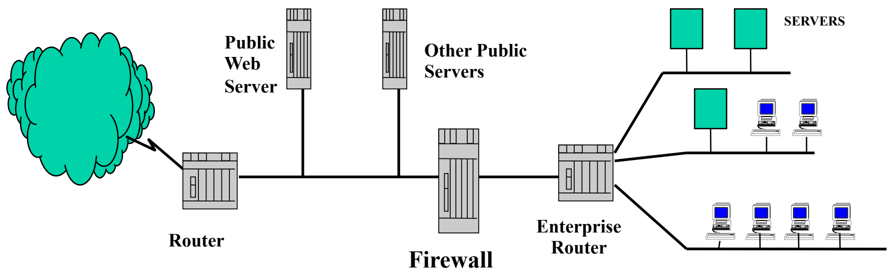
- Firewalls also provide NAT and service redirection
- Application Layer Gateways (ALG) can terminate application layer sessions for individual users and then create new sessions, depending upon security policy.
- Firewalls can provide secure tunnels with encryption for remote users (aka VPNs)
- Firewalls should provide extensive logging, reporting, management and alarms
- Firewalls features are being added constantly. For example: virus checking, spam filters, QoS, use of AI, etc.
- Firewalls can also be located at the ISP or in the Cloud. Managed security is also an option.
- Firewalls are not a complete security solution

The Marketing of Firewalls is Constantly Changing

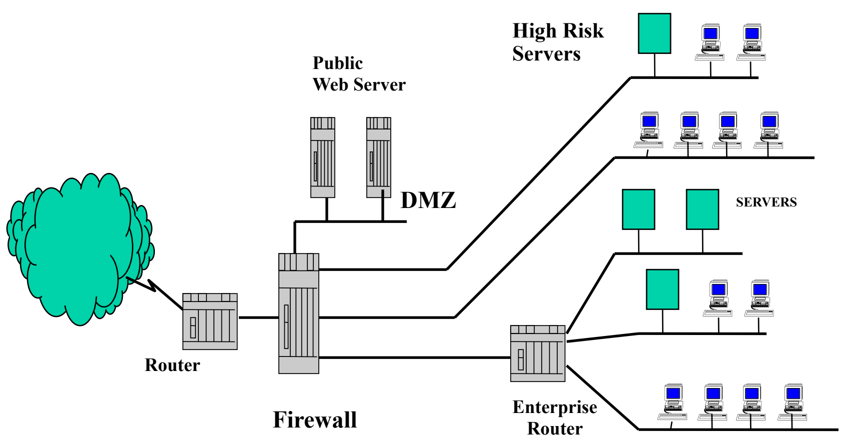


Source: Network World

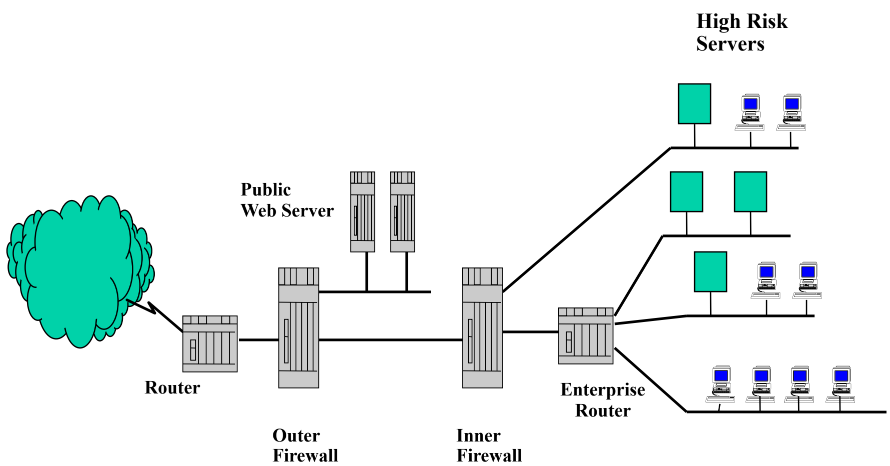
Simple Firewall Architecture - option 1



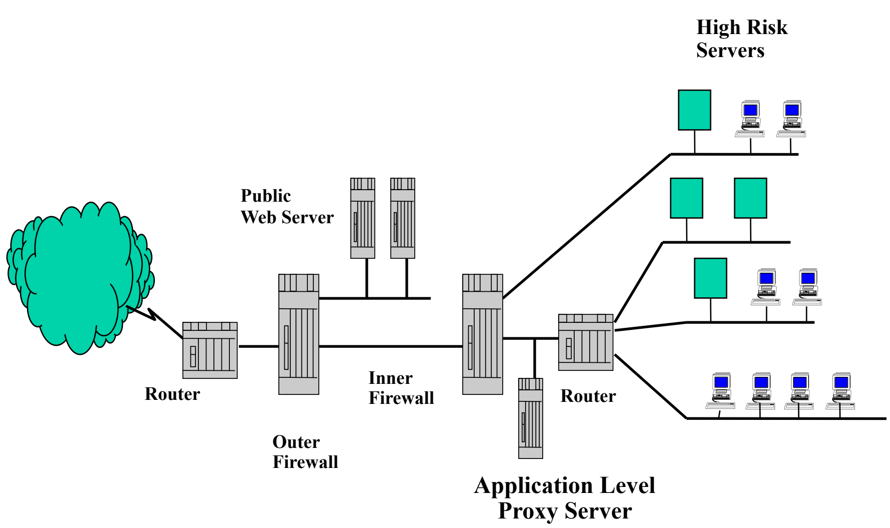
Simple Firewall Architecture - option 2



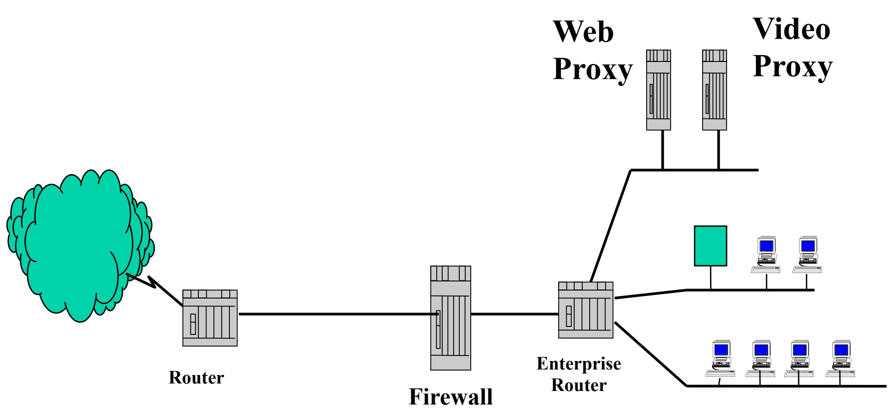
Firewall Architecture - option 3



Firewall Architecture – option 4

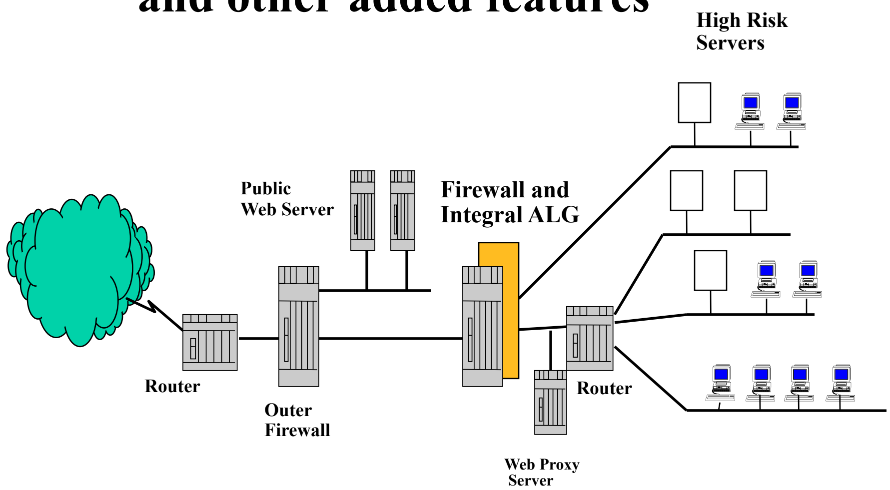


Firewalls and Proxies are Different



© 1998 - 2017 L. Evenchik

Firewall with Integral ALG and other added features



Virtual Private Networks and IPsec

Virtual Private Networks (1)

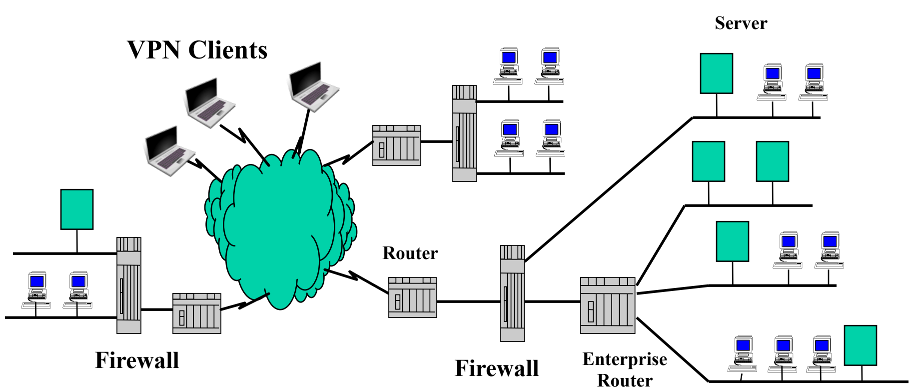
- VPNs provide an encrypted channel over an insecure link. VPNs can be implemented as part of a firewall or as stand-alone software or hardware.
- Two generic types of VPNs: tunnel mode and transport mode.
- Tunnel mode provides a secure encrypted tunnel between different sites for all the users at that site. Tunnel mode is typically done between firewalls (or other edge devices.)
- Sites can be connected together via the public internet or private network circuits (leased or owned.)
- A single access circuit can provide both VPN service and access to the public Internet at the same time.

© 1998 - 2017 L. Evenchik

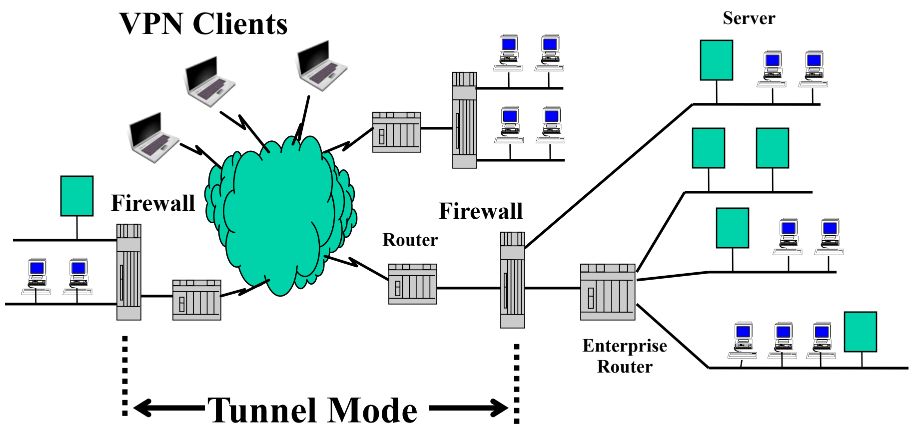
Virtual Private Networks (2)

- Transport mode is typically used to connect an individual user to a specific host. The encapsulated payload can be any application layer protocol (using UDP or TCP.)
- Individual users can be supported by VPN software for access via cable or xDSL and even dial-up. VPN software runs on clients and mobile devices.
- Tunnel mode can also be done between a client on a laptop or mobile devices and a firewall for access to all devices behind the firewall.
- Multiple protocols are available to set up VPNs, both proprietary and IETF protocols such as IPSec.

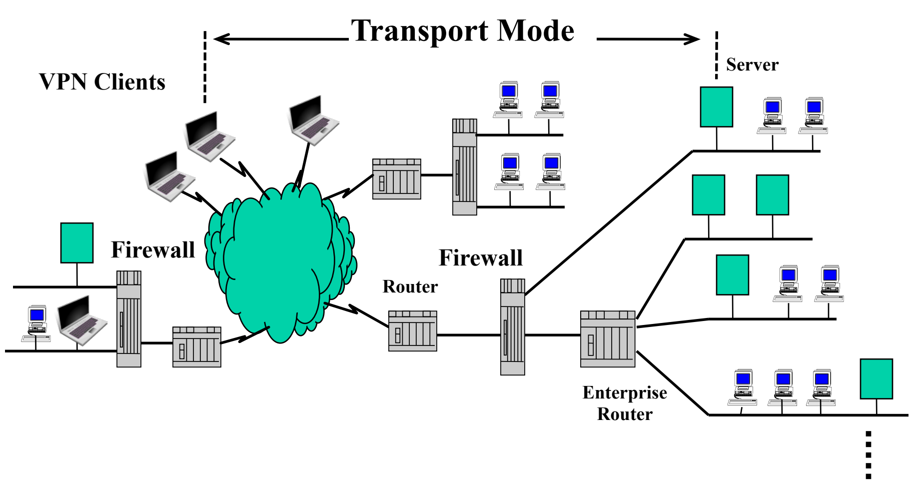
VPN Architecture



VPN Architecture - Tunnel Mode



VPN Architecture - Transport Mode



IPsec

- Developed by a working group of the IETF
- Provides confidentiality, integrity and authentication for IP packets, or UDP/TCP
- Both Tunnel and Transport Mode supported, In tunnel mode, the payload encapsulates IP datagrams. In transport mode, the payload typically encapsulates TCP or UDP.
- Authentication Header (AH) - crypto checksum of contents
- Encapsulated Security Protocol (ESP) - provides confidentiality of contents plus authentication. **This is the common approach today.**
- Key management and exchange is separate
- IPv6 requires that IPSec be supported
- See RFC 4301 as your starting point.

AH Header

```
TRANSPORT MODE
AH is inserted after IP header and before any upper layer protocol

    BEFORE APPLYING AH
            ----------------------------
      IPv4  |orig IP hdr  |     |      |
            |(any options)| TCP | Data |
            ----------------------------
    AFTER APPLYING AH
            ---------------------------------
      IPv4  |orig IP hdr  |    |     |      |
            |(any options)| AH | TCP | Data |
            ---------------------------------
            |<------- authenticated ------->|
                 except for mutable fields

TUNNEL MODE
Use of AH in either hosts or security gateways
          ------------------------------------------------
    IPv4  | new IP hdr* |    | orig IP hdr*  |    |      |
          |(any options)| AH | (any options) |TCP | Data |
          ------------------------------------------------
          |<- authenticated except for mutable fields -->|
          |           in the new IP hdr

    Source RFC 2402
```

ESP Header

```
TRANSPORT MODE
ESP is inserted after IP header and before any upper layer protocol

                 BEFORE APPLYING ESP
            ----------------------------
      IPv4  |orig IP hdr  |     |      |
            |(any options)| TCP | Data |
            ----------------------------

                 AFTER APPLYING ESP
            -------------------------------------------------
      IPv4  |orig IP hdr  | ESP |     |      |   ESP   | ESP|
            |(any options)| Hdr | TCP | Data | Trailer |Auth|
            -------------------------------------------------
                                |<----- encrypted ---->|
                          |<------ authenticated ----->|

TUNNEL MODE
Use of AH in either hosts or security gateways
        -----------------------------------------------------------
  IPv4  | new IP hdr* |     | orig IP hdr*  |   |    | ESP   | ESP|
        |(any options)| ESP | (any options) |TCP|Data|Trailer|Auth|
        -----------------------------------------------------------
                            |<--------- encrypted ---------->|
                      |<------------ authenticated ---------->|
Source RFC 2406
```

One Minute Wrap-Up

- Please do this Wrap-Up at the end of each lecture.
- Please fill out the form on the website.
- The form is anonymous (but you can include your name if you want.)
- Please answer three questions:
  - What is your grand "Aha" for today's class?
  - What concept did you find most confusing in today's class?
  - What questions should I address next time
- **Thank you!**