Cyber Security Technologies

Session 13 – User & Network Authentication / Access Control

Shawn Davis
ITMS 448 – Spring 2016

Slides contain original content from Davis, S. and Lidinsky, W. Slides may also contain content from Ch.3 & 4 of Stallings, W. and Brown, B, Computer Security 2ed; Pearson Education, Inc. 2012

Today

- Logon to your RADISH Win8.1 VM
- Open your Kali VM

Overview

Part I – User Authentication

Part II – Password Cracking Hands On

Part III – Network Authentication

Part IV – Access Control

Part I

User Authentication

Authentication

- Confirming an identity in order to access an object

Profess Identity

Verify Identity

Object

Authentication Process

- Identification Step
  - Presenting something that identifies the user
- Verification Step
  - Presenting or generating authentication information that matches previously stored authentication information

Four Means of Authenticating a User

- **Something you know:**
  - **Password, Pin, Security Question Answers**
- Something you have:
  - Token, Smartcard, Physical Key
- Something you are:
  - Fingerprint, Retina, Iris, Facial Geometry, Hand Geometry
- Something you do:
  - Voice Pattern, Handwriting, Typing Rhythm

Traditional Password Authentication

- Admin creates user account on a server and assigns a user ID and password for the user
- Password is hashed with a one-way function and stored in a protected file on the server
- User enters User ID and password and submits it to authenticate
- Server takes password and performs one-way function to generate hash
- Generated hash is compared against the stored hash
- If they match, user is allowed access to the server

Options to Crack a Password

- Online Attack
  - Use the system's logon mechanism to attempt to crack the password by continually submitting guesses
  - Easy for security admins to catch in logs
- Offline Attack
  - Steal the protected password list and crack the hashes offline
  - Easier for attacker to hide their tracks

Question

- So if hashes are non-reversible one-way functions, how do attackers "crack" them offline once they are stolen???
  - They simply use tools that make many guesses at the password using different methods
  - Each guess is hashed and then that hash is compared against the hash in the stolen file
  - If they match, the password is "cracked"

Methods to Crack a Password

- Dictionary Attack:
  - Crack program has a large wordlist of most every word in the dictionary
  - Will not crack the password if it is not in the wordlist
  - Can use rules to enhance guesses
    - Capitalize first letter of word
    - Append words with numbers
    - Replacing S with $, a with @, O with 0, i with !, etc.
  - Fast

Methods to Crack a Password

- Brute Force Attack:
  - Tries every combination
    - a, b, c, d…z; aa, ab…az; ba, bb…bz, etc.
  - Will eventually crack any password
  - Much slower and may take hundreds of years depending on length and complexity of password

Methods to Crack a Password

- Lookup Table:
  - Takes a wordlist and pre-calculates all of the hashes and stores them in a table
  - Very fast
  - Tables are huge
  - Example is crackstation.net
    - Extracted every word from Wikipedia and from many password lists
    - For MD5 and SHA1 hashes, they have a 15-billion entry lookup table

CrackStation

- Everyone open a browser and go to crackstation.net
- Enter this hash:
  - 7c6a180b36896a0a8c02787eeafb0e4c
  - Enter the captcha and hit “Crack Hashes
- What was the type of hash and the password of the hash???
  - MD5 / password1

Methods to Crack a Password

- Reverse Lookup Table:
  - So far the cracking methods haven't been concerned with the usernames
  - This method uses a Lookup Table but also keeps track of which usernames are mapped to which hashes
  - When a password hash is cracked, all users that had that same password are displayed
  - Very fast

Methods to Crack a Password

- Rainbow Table:
  - Doesn't include all of the pre-calculated hashes
  - Uses reduction function to reduce table size that is considerably smaller than a Lookup table
  - If you are interested, this is a good link to learn the details about how the reduction function works:
    - http://kestas.kuliukas.com/RainbowTables/

Tools to Crack Passwords

- Online Attack
  - THC Hydra (Dictionary, Brute Force)
- Offline Attack
  - John the Ripper (Dictionary, Brute Force)
  - Cain and Abel (Dictionary, Brute Force)
  - hashcat (Dictionary, Brute Force
  - crackstation.net (Lookup Table)
  - Rainbow Crack (Rainbow Table)
  - OphCrack (Rainbow Table)

Password Guessing

- Realize that often you can simply determine a user's password or the answers to their password reset questions
- How might you do this???
  - Information provided on Social Media sites
  - Social Engineering
  - Looking under their keyboard tray for a post-it
- Users often also use the same password on multiple sites.
  - Crack one and have them all

Scenario

Stolen Hashed Passwords	Cracked Passwords So Far
7c6a180b36896a0a8c02787eeafb0e4c	password1
7b4c4fd444fff31be2750b0d78a53788	beth33
6004d8e974c5a9f0d369f20d532e1f71	david1982#.%
7c6a180b36896a0a8c02787eeafb0e4c	password1

- Attacker used a lookup table or rainbow table to crack these passwords
- What could have stopped the attacker from using those two methods???
  - Using a random salt!

What is a Salt?

- A random string prepended or appended to the password before it is hashed
- Should you use the same salt for each password or a different salt?
  - A different salt
- Why???
  - If the same salt were used, the attacker could simply create a new lookup table with hashes pre-calculated with that salt

Salting (Cont.)

- Also, you should have a longer salt (8 characters or more)
- Why??
  - If a small salt such as 2 or 3 digits was used, the attacker could simply brute force it as well
- So now we know that a longer random salt can effectively stop lookup and rainbow tables

Salting (Cont.)

- Can a random and long salt also stop the use of dictionary and brute force cracking methods???
  - No
- Why Not???
  - In an online attack, the authentication function will add the correct salt automatically to the dictionary or brute force guesses
  - In an offline attack, the attacker already has the salt in the stolen file that contains the password hashes

Unix Passwords

- Unix passwords are encrypted with the Unix *crypt()* function
  - crypt() is a hash function based upon DES
  - It is <u>not</u> the Unix crypt utility
    - Rather weak encryption
  - crypt() really should have been called something else
- *crypt()* hashes the plaintext password
  - Uses one of several "salts"
    - Salts are given as an argument

Old Unix Passwords

- Old Unix (not Linux) passwords were stored in the ***/etc/passwd*** text file as part of the account information for each user
- The account info format for each user looks like this:
- loginName **:** passwordHash **:** uid **:** gid **:** userInfo **:** homeDirectory **:** loginProg

Old Unix Passwords

- Examples

```
lidinsky:Qp47caKps8tN:123:123:Bill Lidinsky:/home/lidinsky:/bin/bash
student:$1$3MkY0$edjui.5DE7DYn/F8V25S:14076:0:99999:7:::
root:$l$w6jJr$7jsFOg7za9J//ul9Wul:14076:0:99999:7:::
```

- Items are separated with a ":"
- The password is hashed

A Security Problem

- The ***/etc/passwd*** file must be readable globally because info in it is used by many Unix tools and applications
  - **This presented a serious security problem**
- What to do?
  - Could have changed many many tools and applications
  - Better idea: Create an **/etc/shadow** file

Current Unix & Linux Passwords

- Similar to old Unix but
- The password entry in ***/etc/passwd*** is replaced with an "x"
- The hashed password is put into ***/etc/shadow*** file

Current Unix & Linux Passwords

- So in /etc/passwd we have

  loginName : **x** : uid : gid : userInfo : homeDirectory : loginProg

- Examples of */etc/passwd* accounts

```
student:x:1001:1001:student:/home/student:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

Current Unix & Linux Passwords

- In the ***/etc/shadow*** file there is for each login name the following line format:

loginName : passwordHash : *#days_since_last_changed*
:*#days_before_may_be_changed* : *#days_after_which_must_be_changed*
:*#days_to_warn_user_of_expiring* : *#days_after_expired_account-disabled* :
#days_since_account_disabled : *reserved_filed*

- Examples

```
lidinsky:Qp47caKps8tN:723:0:99999:7:::
student:$1$3MkY0$edjui.5DE7DYn/F8V25S:14076:0:99999:7:::
root:$l$w6jJr$7jsF0g7za9J//ul9Wul:14076:0:99999:7:::
```

Linux Password Hash Format

- `$1$w6jJr$7jsFOg7za9J//ul9Wul`

- $ is delimiter
- 1st field is hashing algorithm used:
  - $1 = MD5
  - $2 = Blowfish
  - $5 = SHA-256
  - $6 = SHA-512

Linux Password Hash Format

- `$1$w6jJr$7jsFOg7za9J//ul9Wul`

- 2nd field is salt value
  - `w6jJr`
- 3rd field is hash of salt and plaintext password together
  - `7jsFOg7za9J//ul9Wul`

Linux Passwords

- Why are hashed passwords are more secure in the /etc/shadow file???
  - By default, only the root user may access it.

Hand's On

- In Kali, open the terminal
- By default, you are logged in as what user?
  - Root, so you can do whatever you want
- View the /etc/passwd and /etc/shadow files
- What hashing algorithm was used to create the hashes?

```
btaylor:$6$hEKsRCAa$EbYpSDWhgX.NypIAOEEq3o.pUrufr.SqeJwCk2xWawbTddlXZYO8IhwNSO6r
UNkFZlk.wFm9puQd55a2Z13610:16732:0:99999:7:::
```

  - $6 – SHA-512

Linux Passwords

- To steal the passwords for an offline attack, one simply has to copy the shadow file to a remote system
- What methods could an attacker use to do so?
  - SCP, FTP, HTTP, HTTPS, Email, etc.
  - IM, P2P, Tweets
  - Echo Requests
  - Stego files
- Interesting paper on exfiltration techniques: http://udspace.udel.edu/bitstream/handle/19716/10145/Ryan_VanAntwerp_thesis.pdf?sequence=1

Windows Passwords

- Windows OSs may hash a password twice using two different algorithms
  - LM or LMHash
  - NT or NTHash

LM Windows Passwords

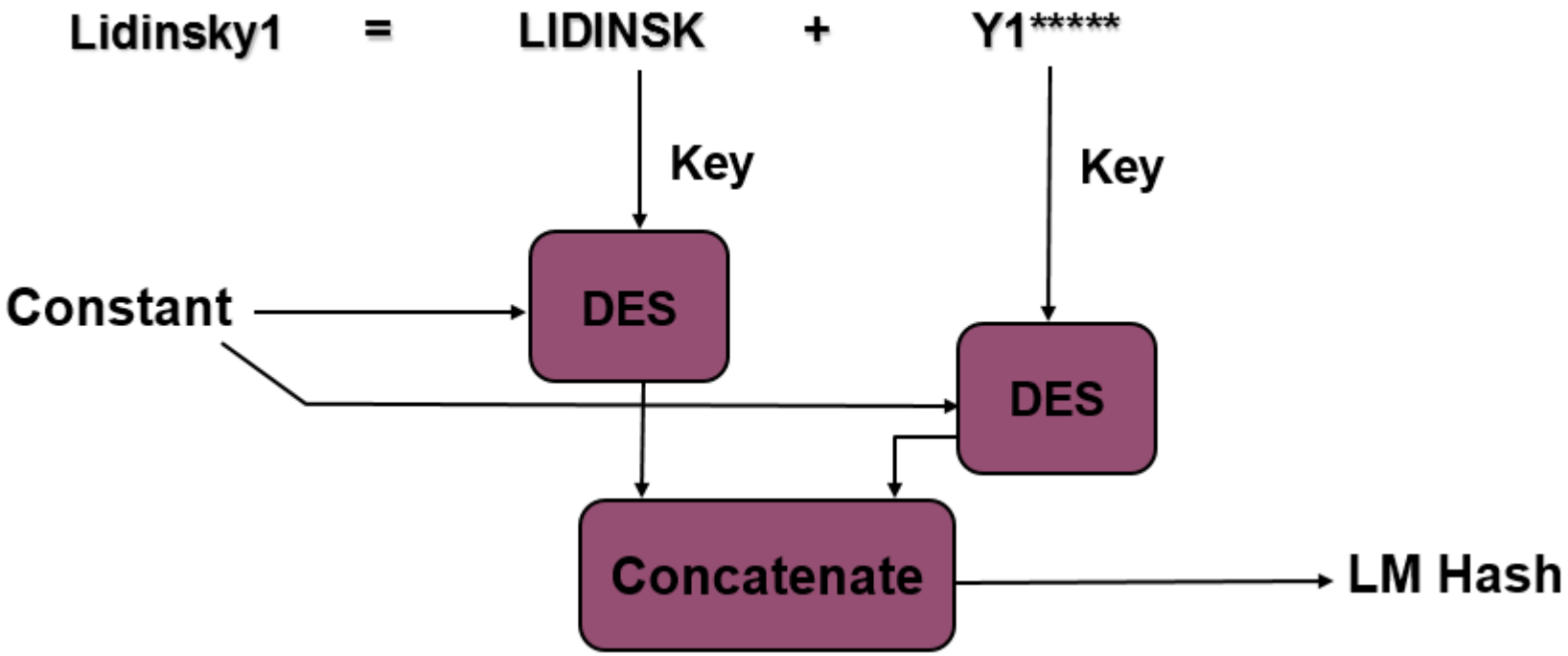
- LM allows alphanumeric characters and symbols (👍)
- LM converts all password plaintext characters to upper case before hashing (👎)
- Pads out plaintext passwords to 14 characters (👎)
- Divides modified plaintext passwords into two 7-byte parts
- Hashes each 7-byte part separately (👎)
- Concatenates the two hashes to form the LM hash
- Notes
  - Use a fixed salt across all Windows OSs (👎)
  - Number of password characters must be ≤ 14

LM Windows Passwords

- Passwords ≤ 14 characters or bytes are padded with nulls
- To each 7-byte part an odd parity byte is added
- This 8-byte value becomes a DES key
- DES is used with a fixed salt to create each 8-byte hash
- The two hashes are concatenated together to create the 16-byte LM hash
- LM provides rather weak hashing

LM Hash Generation

- Converted to upper case
- Padded with NULL to 14 characters
- Separated into two 7 character strings which are hashed separately

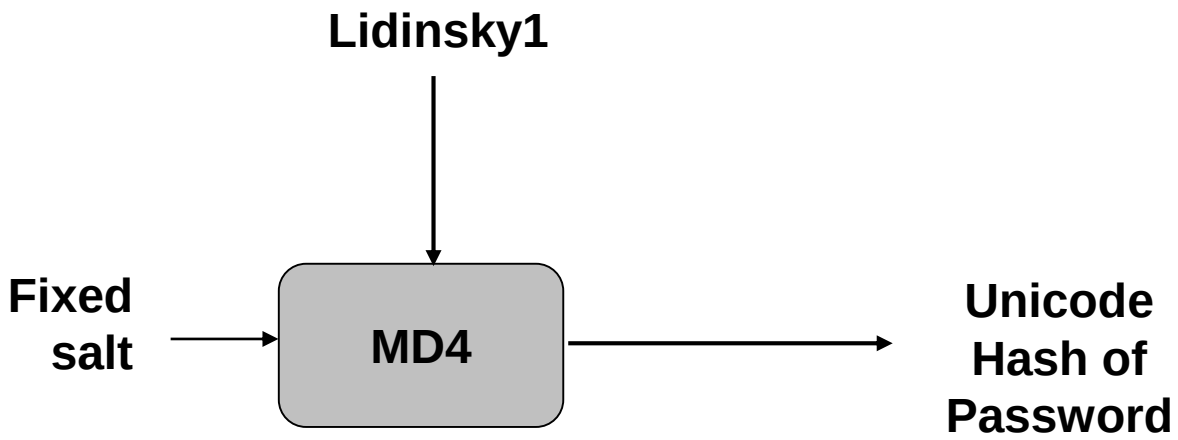


NT Hash Windows Passwords

- NT hash allows alphanumeric characters & symbols (👍)
  - Characters and symbols are represented as Unicode
- Keeps both lower and upper case characters (👍)
- Does not split up the plaintext password (👍)
- But still uses a fixed salt (👎)
- Creates a 16-byte hash

NT Hash Generation

- Just hash the password
- Then store it



Windows Passwords

- From Win2K forward the two hashes are both calculated and stored in the SAM (Security and Account Manager) part of the registry
  - There are ways to eliminate the LM hash
  - But then backward compatibility is compromised
- Windows stores encrypted password hashes in the Windows Registry
  - Unlike Linux, password files don't exist in Windows
- Need special tools to extract passwords and other account information
  - e.g., pwdump, pwdump2, pwdump3, samdump…

Windows Passwords

- Both LMHash and NTHash are enabled in windows OSs up to XP and 2003 server
  - But LM can be disabled
- In Vista, Win7 and 2008 server, LM is disabled by default
  - But LM can be enabled for backward compatibility

No Salt Implications

- The same plaintext password will yield the same hash on all Windows OSs
- A password cracker can therefore create a large lookup table and break passwords on any Windows computer

Why are the Following Bad Passwords???

- Computer6
  - Cracker can use wordlist and add rule to append numbers
- sdavis10
  - Cracker might have a rule that tries all usernames as the password and appends numbers
- asdfjkl;
  - Crackers often have a rule for letters in close proximity on the keyboard

Why are the Following Bad Passwords???

- #9dk34D.39@@kduv2349Blc2
  - No user could remember that and would probably write it down

Why is the Following a Good Password???

- R3ady4tHaWknd?1
  - 15 Characters
    - Many Rainbow and Lookup Tables only go to eight Characters
    - A brute force cracker on a single desktop PC would take about 335 billion years to crack this password
  - Not a dictionary word
  - Contains Uppercase, Lowercase, Numbers, Symbols
    - Character set is 94

Is This a Good Password???

- ThisIsTheMostSecurePasswordEverInTheWorld
  - Yes
  - 40 Characters
    - Many Rainbow and Lookup Tables only go to eight Characters
    - A brute force cracker on a single desktop PC would take about 179 secdecillion years to crack this password
  - Not a dictionary word
  - Contains Uppercase, Lowercase
    - Character set is 52
- Crackers are starting to build wordlists with longer phrases though
- Hashcat can crack password lengths of 55 characters

Password Cracking Defenses

- User education
- Password vulnerability scanner
- Password blacklist
- Password complexity and length requirements
- Salt and secure file that contains hashes
- Periodic required password changes
- Account lockout after predetermined number of failed authentication attempts
- Use multifactor authentication (biometrics, token, etc.)

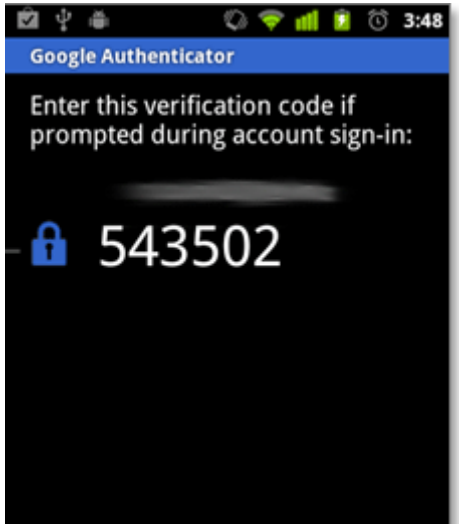
Multifactor Authentication

- Requires more than one authentication method to verify identity of user
- Examples:
  - Password & Fingerprint
  - Password & Token
  - Credit Card & Pin
  - Voice Scan & Iris Scan
  - Physical Key & Key Code

Four Means of Authenticating a User

- Something you know:
  - Password, Pin, Security Question Answers
- **Something you have:**
  - **Token, Smartcard, Physical Key**
- Something you are:
  - Fingerprint, Retina, Iris, Facial Geometry, Hand Geometry
- Something you do:
  - Voice Pattern, Handwriting, Typing Rhythm

Hardware & Software Tokens



How Do Tokens Work?

- Requires Token and Backend Authentication Server
- User receives Token that contains unique seed file which contains symmetric key
- Seed file also loaded on Authentication Server
- Both Token and Server use pseudo-random number generator and both start on same number
- Both token and server change their number at fixed intervals (such as every 60 seconds)
  - Accurate time is important
- Sequence of number change based on secret RSA Algorithm and the seed value used to initialize the token

How Do Tokens Work? (Cont.)

- After initial setup is complete, no communication or synchronization between token and server occurs
  - User must manually enter code as second factor
- Some Tokens have an additional PIN that must be entered

Can Tokens Be Attacked???

- Yes, an attack occurred in 2011 that required RSA to replace almost all of the 40 million hardware Tokens that were in use
- Any ideas of what the attackers needed to do to compromise the Tokens???

RSA Attack

- Hackers sent phishing emails to RSA employees with subject line "2011 Recruitment Plan" with attached Excel file
- Excel sheet contained 0-day flaw in Adobe Flash in order to install a backdoor with Poison IVY RAT
- Attackers harvested passwords and pivoted around until they gained access to and stole RSA data
  - This data may have been the seed files and serial numbers to Tokens
- RSA's "secret" algorithm was already disclosed publicly and wasn't so secret

RSA Attack (Cont.)

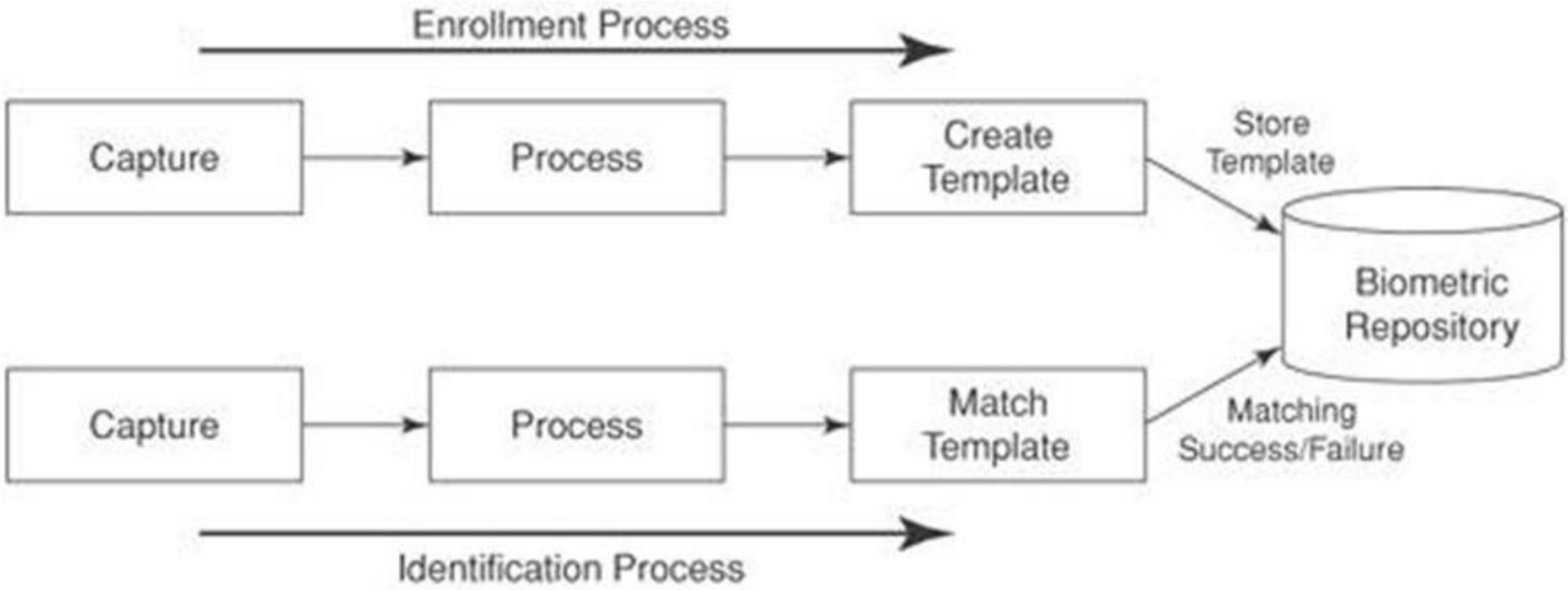
- Attackers could use seed, Tokens, and serial numbers to impersonate users
- RSA replaced all Tokens

Biometrics

Four Means of Authenticating a User

- Something you know:
  - Password, Pin, Security Question Answers
- Something you have:
  - Token, Smartcard, Physical Key
- **Something you are:**
  - **Fingerprint, Retina, Iris, Facial Geometry, Hand Geometry**
- **Something you do:**
  - **Voice Pattern, Handwriting, Typing Rhythm**

Biometric Process



http://www.globalsecurity.org/security/systems/biometrics.htm. 6.12.2010

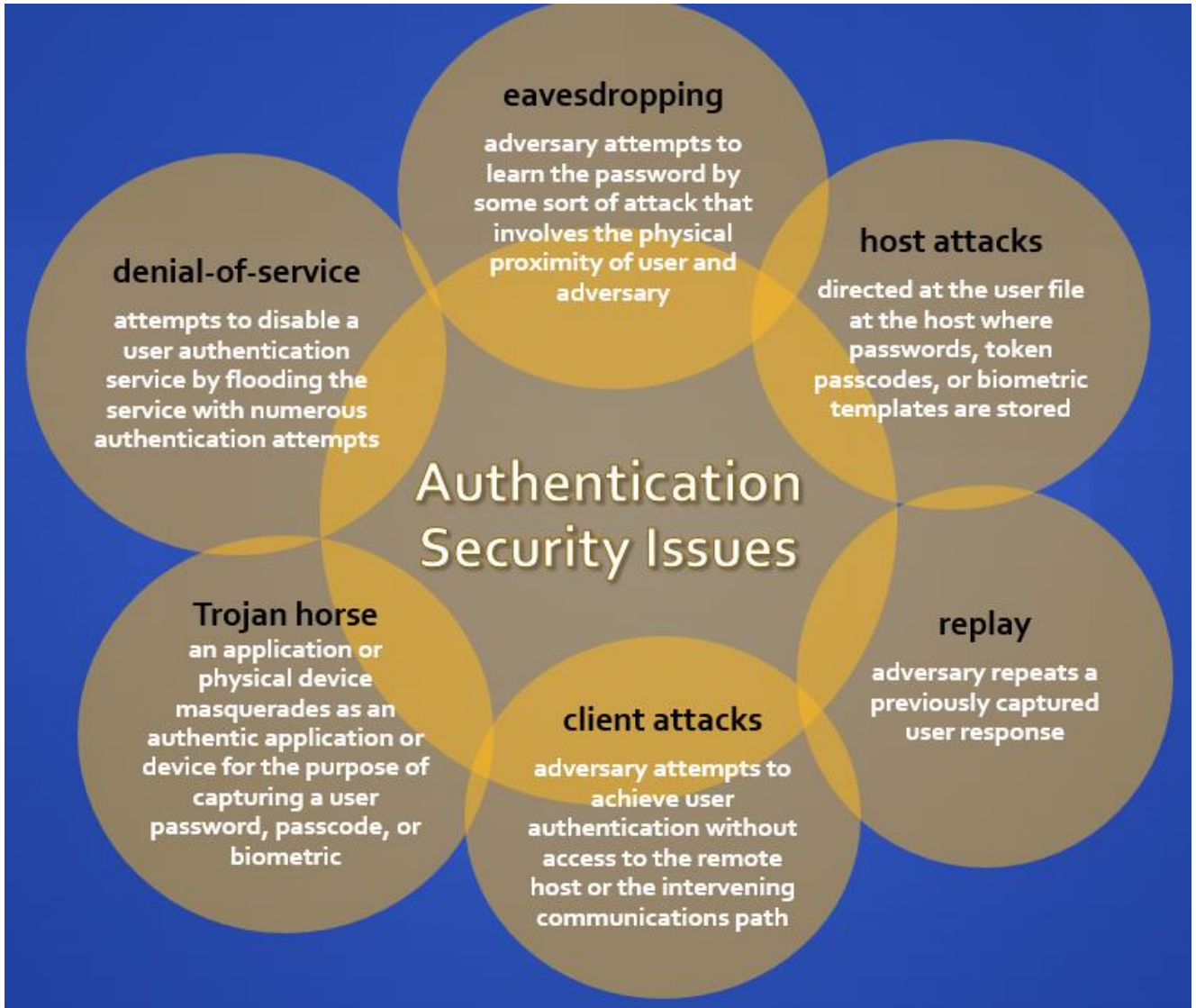
Biometric Attacks and Defenses

- Fingerprint
  - Attack: Use clay, gummy bears, gelatin, etc. to capture fingerprint and use on sensor
  - Defense: Sensor detects moisture, electrical charge, etc.
- Iris
  - Attack: Use a high resolution photograph to trick the sensor
  - Defense: Use lighting to check if a real Iris dilates or not.

Biometric Attacks and Defenses (Cont.)

- Facial Recognition
  - Attack: Use high resolution photograph to trick the sensor
  - Defense: Use a 3D sensor
- Voice Recognition
  - Attack: Replay attack with voice recording for passphrase
  - Defense: Detect unnatural utterances

General Authentication Security Issues



General Authentication Security Defenses

Attacks	Authenticators	Examples	Typical defenses
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts, theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response

General Authentication Security Defenses

Eavesdropping, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication

General Authentication Security Defenses

Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

Part II

Password Cracking Hands On

Exercise 1: Cracking Old Unix Passwords

- Open a terminal in Kali Linux
- Kali 2.0's john has a bug so we need to upgrade it first
- **apt-get --only-upgrade install john**

Exercise 1: Cracking Old Unix Passwords

- **echo e91e6348157868de9dd8b25c81aebfb9 > hashfile**
- **john –format:raw-md5 hashfile**

- What is the cracked password and how long did it take???
  - security (less than 1 second)

Exercise 1: Cracking Old Unix Passwords

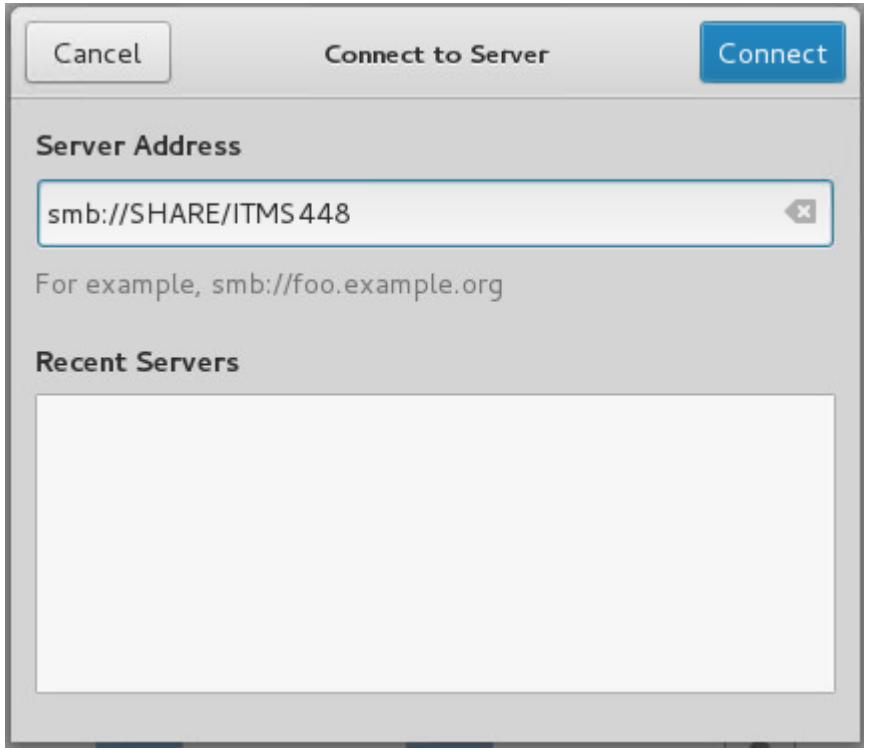
```
root@KLY-IR105:~# echo e91e6348157868de9dd8b25c81aebfb9 > hashfile
root@KLY-IR105:~# john -format:raw-md5 hashfile
Created directory: /root/.john
Loaded 1 password hash (Raw MD5 [128/128 SSE2 intrinsics 12x])
security         (?)
guesses: 1  time: 0:00:00:00 DONE (Sat Nov 14 02:32:10 2015)  c/s: 21000  trying
: roger - snapple
Use the "--show" option to display all of the cracked passwords reliably
root@KLY-IR105:~#
```

Exercise 2: Cracking Present Day Linux Passwords

- You are an attacker that breached a system and stole the passwd and shadow files and uploaded them to your site to be cracked offline
- The files are on the M: Drive
- If mounted, you should see an itms448 share link on the Desktop and go ahead and double-click on it
- If it is not mounted, follow the steps in the next four slides

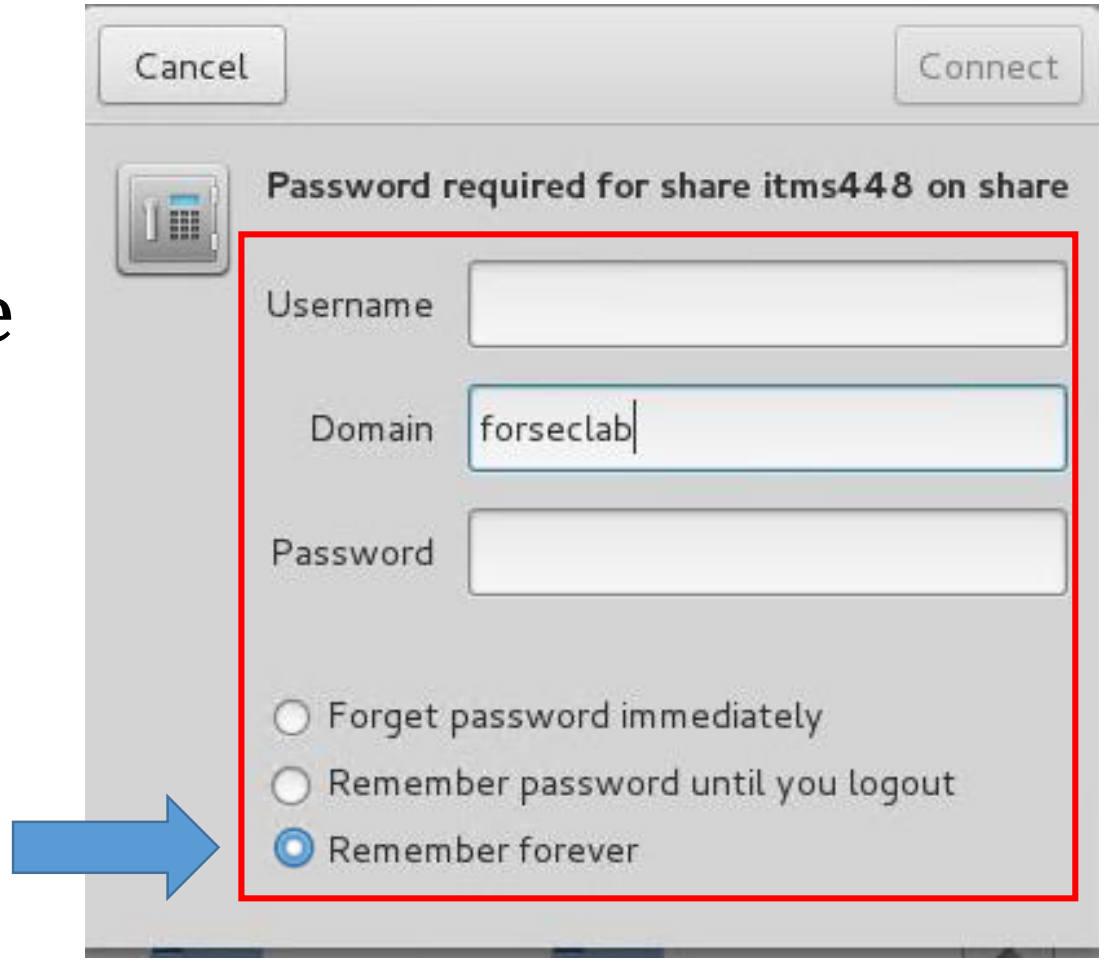
Kali Linux VM access to "M: Drive"

- Click on *Places* and then *Computer*
- Select *Connect to Server* on the bottom left
- Server Address:
  - smb://SHARE/ITMS448
  - Hit Connect



Kali Linux VM access to "M: Drive"

- In a few seconds an authentication window will appear
- Enter your RADISH username and password
- The Domain is forseclab
- Choose "Remember forever"
- Hit Connect

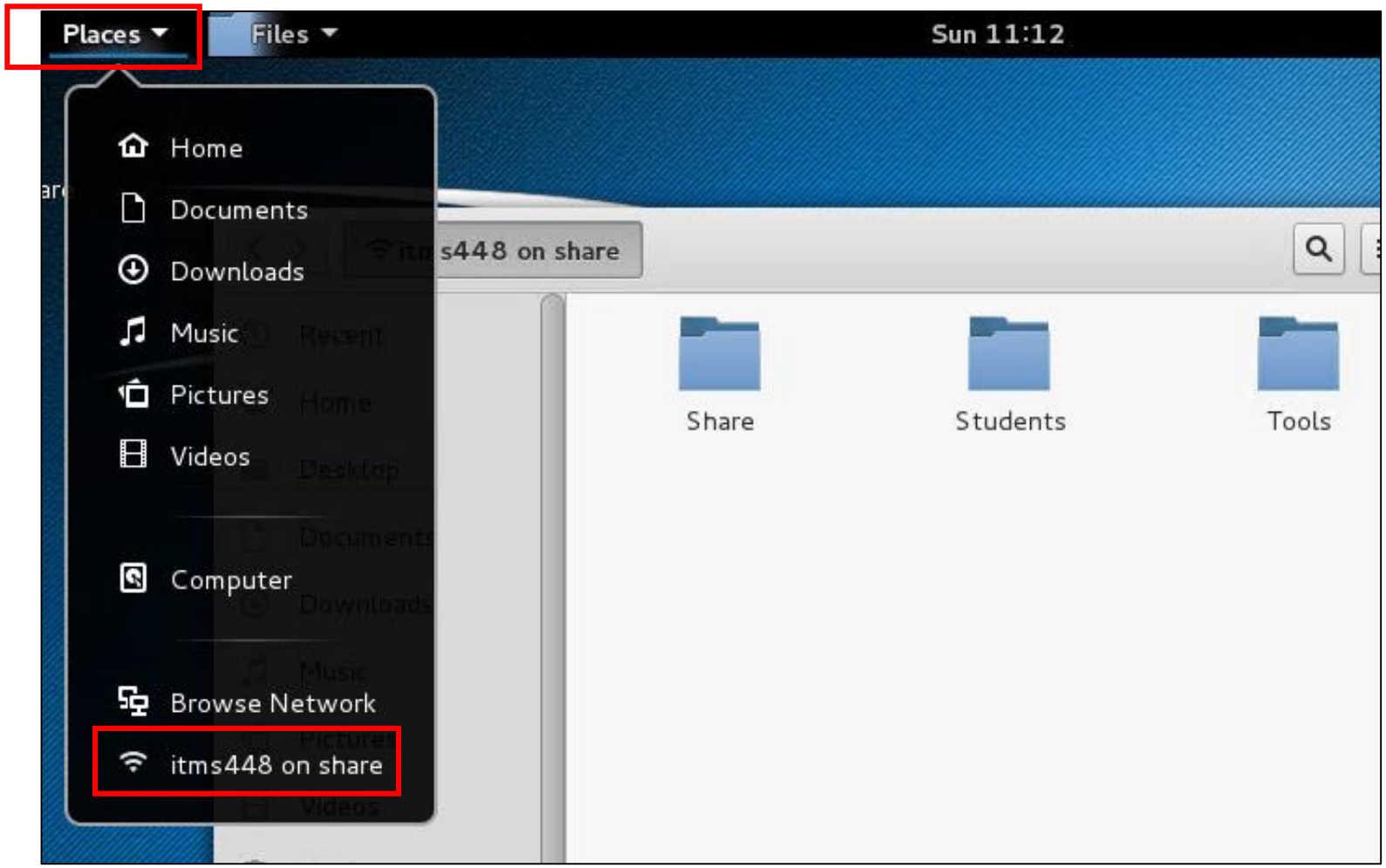


Kali Linux VM access to “M: Drive”

- Enter your Kali root user password which is:
- toor
- Hit Unlock

Kali Linux VM access to “M: Drive”

- Share will then be mounted:



Kali Linux VM access to "M: Drive"

- In the itms448 share, go to Tools and open the Password In-Class Labs folder
- Drag the LinuxLab folder to your desktop
- Open a terminal
- **cd Desktop/LinuxLab**
- **ls**
- You should now see a folder called adminuser which you will use in the next exercise

Exercise 2: Cracking Present Day Linux Passwords

- If you are familiar with john the ripper, feel free to try to crack the password on your own
- Otherwise, follow along

Exercise 2: Cracking Present Day Linux Passwords

- **cd adminuser**
- **ls**
- **cat passwd**
- **cat shadow**
- What is the name of the user?
  - adminuser
- What type of hashing was used?
  - $6 = SHA-512
- What is the salt?
  - 9QAJYzZF

Exercise 2: Cracking Present Day Linux Passwords

- Now, we need to combine the passwd and shadow files into a format that john can crack
- We will use the unshadow command to do this
- Make sure you are still in the adminuser directory
- **unshadow passwd shadow > /usr/share/john/filetocrack**
- Now, change directories to where john is kept
- **cd /usr/share/john**
- **cat filetocrack**

Exercise 2: Cracking Present Day Linux Passwords

- At this point let's use john to try to crack the password
- We are going to use a dictionary wordlist for this attempt
- Use **ls** to see the files in the john directory

Exercise 2: Cracking Present Day Linux Passwords

- Run john to try and crack the password
- **john –w:password.lst filetocrack**
- Did it crack the password???
  - No

Exercise 2: Cracking Present Day Linux Passwords

- We are pretty sure that the password is in the password list but it didn't seem to crack
- What else could we try with the same list???
  - Try john's mangling ruleset!
  - Located in /etc/john/john.conf

john's Default Rules List

```
# Wordlist mode rules
[List.Rules:Wordlist]
# Try words as they are
:
# Lowercase every pure alphanumeric word
-c >3 !?X l Q
# Capitalize every pure alphanumeric word
-c (?a >2 !?X c Q
# Lowercase and pluralize pure alphabetic words
<* >2 !?A l p
# Lowercase pure alphabetic words and append '1'
<* >2 !?A l $1
# Capitalize pure alphabetic words and append '1'
-c <* >2 !?A c $1
# Duplicate reasonably short pure alphabetic words (fred -> fredfred)
<7 >1 !?A l d
# Lowercase and reverse pure alphabetic words
>3 !?A l M r Q
# Prefix pure alphabetic words with '1'
>2 !?A l ^1
# Uppercase pure alphanumeric words
-c >2 !?X u Q M c Q u
# Lowercase pure alphabetic words and append a digit or simple punctuation
<* >2 !?A l $[2!37954860.?]
```

john's Default Rules List

```
# Words containing punctuation, which is then squeezed out, lowercase
/?p @?p >3 l
# Words with vowels removed, lowercase
/?v @?v >3 l
# Words containing whitespace, which is then squeezed out, lowercase
/?w @?w >3 l
# Capitalize and duplicate short pure alphabetic words (fred -> FredFred)
-c <7 >1 !?A c d
# Capitalize and reverse pure alphabetic words (fred -> derF)
-c <+ >2 !?A c r
# Reverse and capitalize pure alphabetic words (fred -> Derf)
-c >2 !?A l M r Q c
# Lowercase and reflect pure alphabetic words (fred -> fredderf)
<7 >1 !?A l d M 'l f Q
# Uppercase the last letter of pure alphabetic words (fred -> freD)
-c <+ >2 !?A l M r Q c r
# Prefix pure alphabetic words with '2' or '4'
>2 !?A l ^[24]
# Capitalize pure alphabetic words and append a digit or simple punctuation
-c <* >2 !?A c $[2!3957468.?0]
# Prefix pure alphabetic words with digits
>2 !?A l ^[379568]
# Capitalize and pluralize pure alphabetic words of reasonable length
-c <* >2 !?A c p
```

john's Default Rules List

```
# Lowercase/capitalize pure alphabetic words of reasonable length and convert:
# crack -> cracked, crack -> cracking
-[:c] <* >2 !?A \p1[lc] M [PI] Q
# Try the second half of split passwords
-s x**
-s-c x** M l Q
```

Exercise 2: Cracking Present Day Linux Passwords

- Now we can run the crack again with the ruleset
  - **john -w:password.lst --rules filetocrack**
  - Your system should take around 00:02:30 to crack it
- While we are waiting, take a look at the common passwords shown in the dictionary wordlist
- Open a new terminal:
  - **gedit /usr/share/john/password.lst**
- What is the cracked password when done?
  - Volcano2

Example of Writing a Custom Rule for John

```
# Shawn Custom Rules
[List.Rules:Append3Numbers]
# Capitalize dictionary words and appends 3 digits
-c <* >2 !?A c $[0123456789]$[0123456789]$[0123456789]
~
~
```

```
root@kali:/usr/share/john# john -w:lower.lst --rules:Append3Numbers tocrack
```

```
Cirrculation002  (salesadmin)
guesses: 1  time: 0:00:32:51 DONE (Fri Apr 11 22:36:51 2014)  c/s: 482  trying:
```

- Great slide deck on custom rules for john.conf:

https://www.owasp.org/images/a/af/2011-Supercharged-Slides-Redman-OWASP-Feb.pdf

Openwall Wordlists

- You can purchase large wordlists here:
  - http://www.openwall.com/wordlists/
- You can download free wordlists here:
  - http://download.openwall.net/pub/
- Of course you can also just make your own

John Incremental Mode

- Also, in/etc/john/john.conf are the incremental modes for brute force attacks
- Examples
  - All characters
  - All characters for 6, 7, 8, and 15 characters
    - Used to be there but you will add the 6 character mode for the homework
  - All alpha characters
  - All numerical characters
  - Etc.

John Notes

- John keeps track of what passwords have been cracked.
- If you need to start over, delete the john.pot file which is located in a hidden .john folder off of your user's directory
- Example for root user
- **rm /root/.john/john.pot**

Windows Passwords

- We won't be demoing this today but all that is necessary is to:
  - Grab passwords from the registry with newer version of pwdump
  - Use your cracking tool of choice that works with LM and/or NT hashes

Part III

Network Authentication

Windows Infrastructure

- Domain Controller (DC)
  - System running Windows Server OS and has Active Directory Domain Services installed as role
- Active Directory (AD)
  - Uses database to authenticate and authorize users based on policy as well as LDAP for modification and queries
- Lightweight Directory Access Protocol (LDAP)
  - Protocol used to query and modify items in directory services providers such as AD

AD Domain Trees

- Root domain = chase.com
- Other contiguously named domains can be part of same tree as root domain:
  - sales.chase.com
  - us.sales.chase.com
  - Etc.
- Non-contiguously named domains can be part of same AD forest:
  - chase.com
  - chaserewards.com

AD Domain Objects

- AD can control and manage users, computers, and groups within network
  - User Accounts
    - Credentials to access network objects
  - Computer Objects
    - Specific functions allowed to be performed on computers in network
  - Security Groups
    - All members receive same permissions

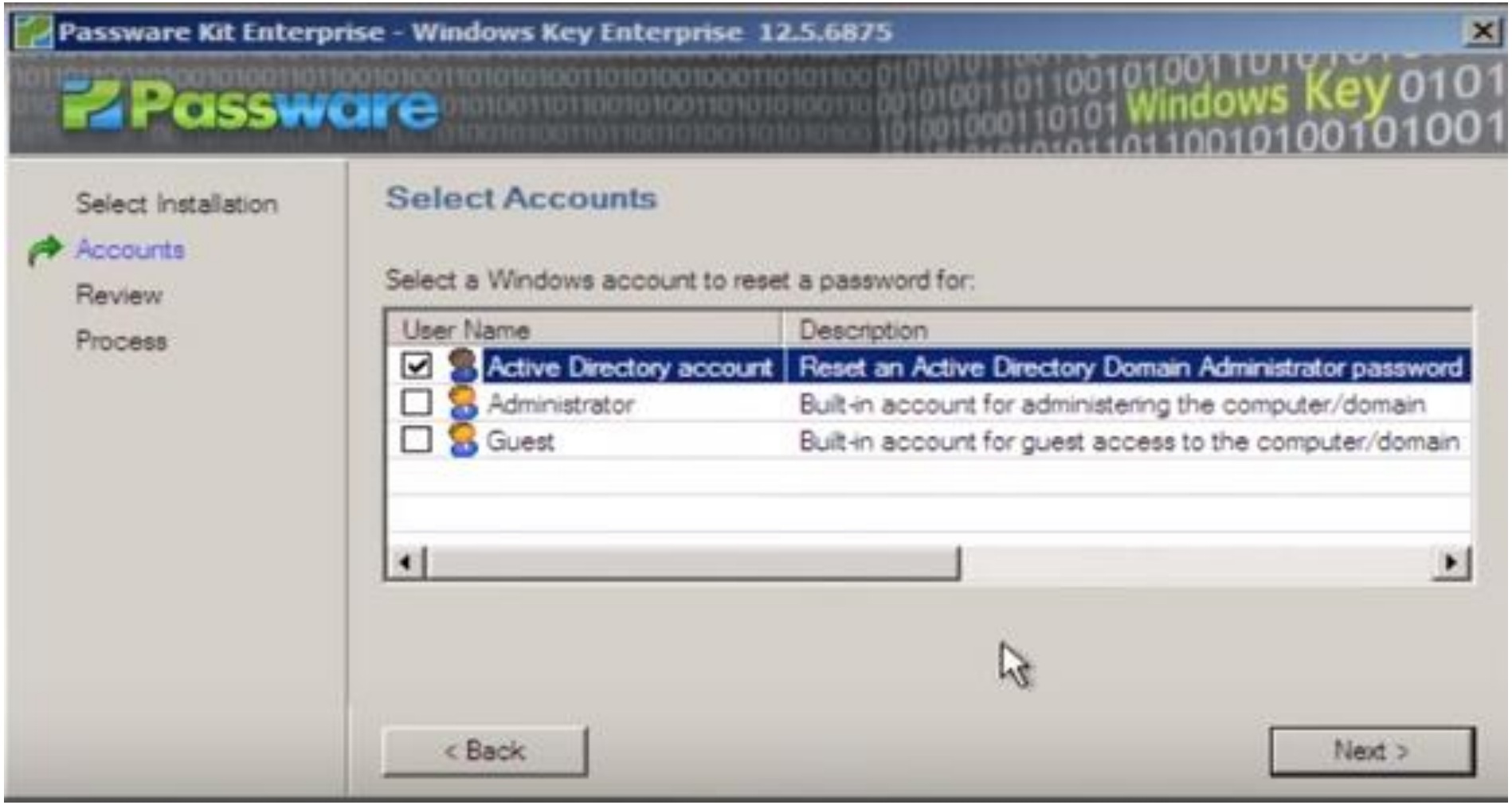
Domain Controller Security

- Domain Controller with AD needs to be most secure
- An attacker who takes over the DC essentially owns the entire Windows infrastructure
  - All Admin Accounts, User Accounts, etc.
- Internal DC needs to be physically protected and needs to be protected by perimeter firewall with no public access to the DC

Domain Controller Physical Attack Scenario

- An attacker could just walk into a data center and physically steal a DC
- Attacker loads Passware Winkey Enterprise Boot Disk into optical drive
- Selects option to reset AD domain admin password

Domain Controller Physical Attack Scenario



Domain Controller Physical Attack Scenario

- Remove Passware disk and reboot
- Logon with blank password
- Use pwdump to extract all user accounts and password hashes on the domain
- Crack password hashes

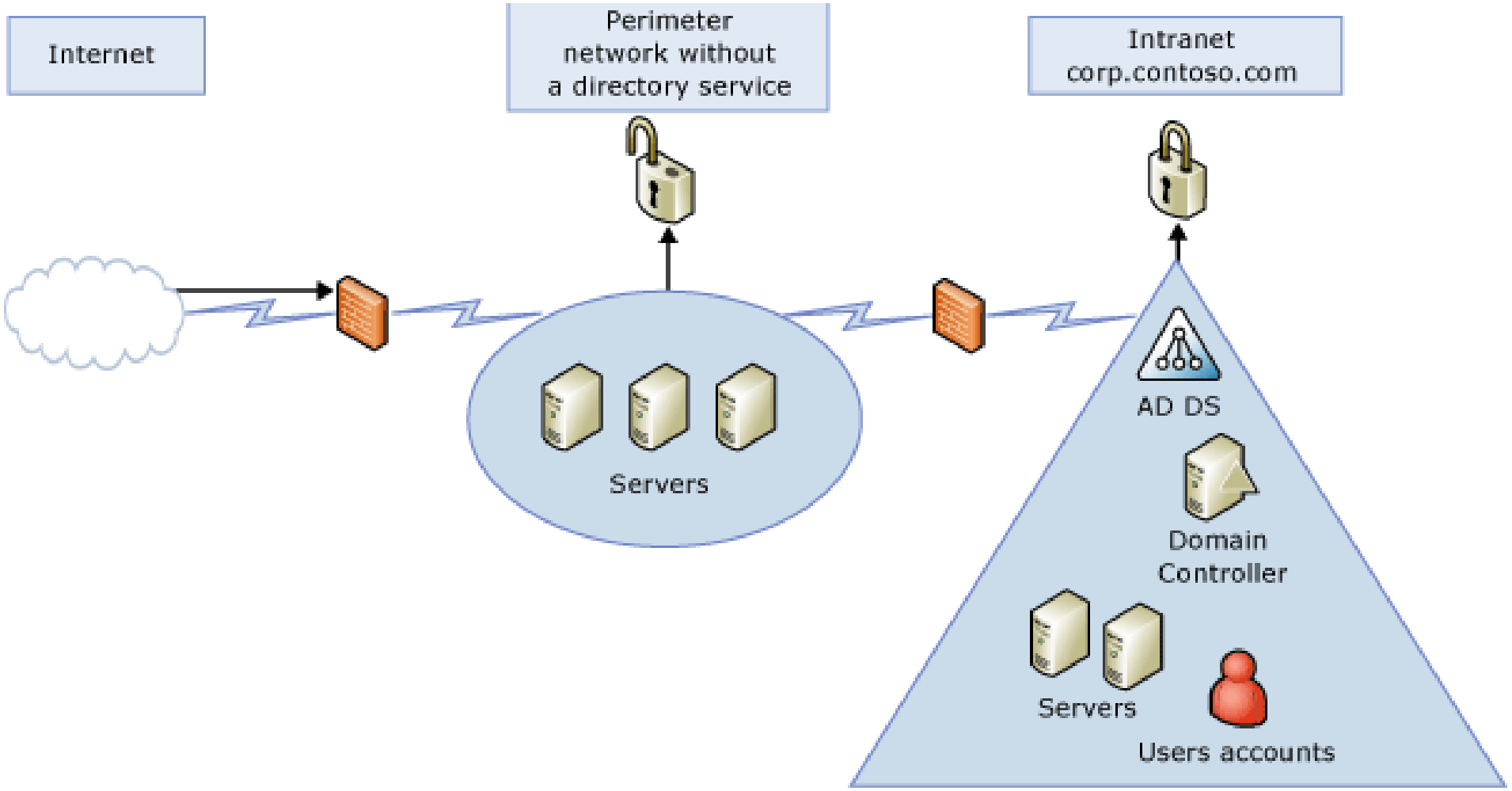
Domain Controller – Internal & DMZ Access

- Internal DC therefore needs to be physically protected but also needs to be protected by perimeter firewall with no public access to the DC
- What do we do about authentication to systems on the DMZ (web servers, etc.) if there is no access there to the internal DC?
- Also, what if we have a small branch office that doesn't have an IT staff and requires authentication as well?

One Problem…

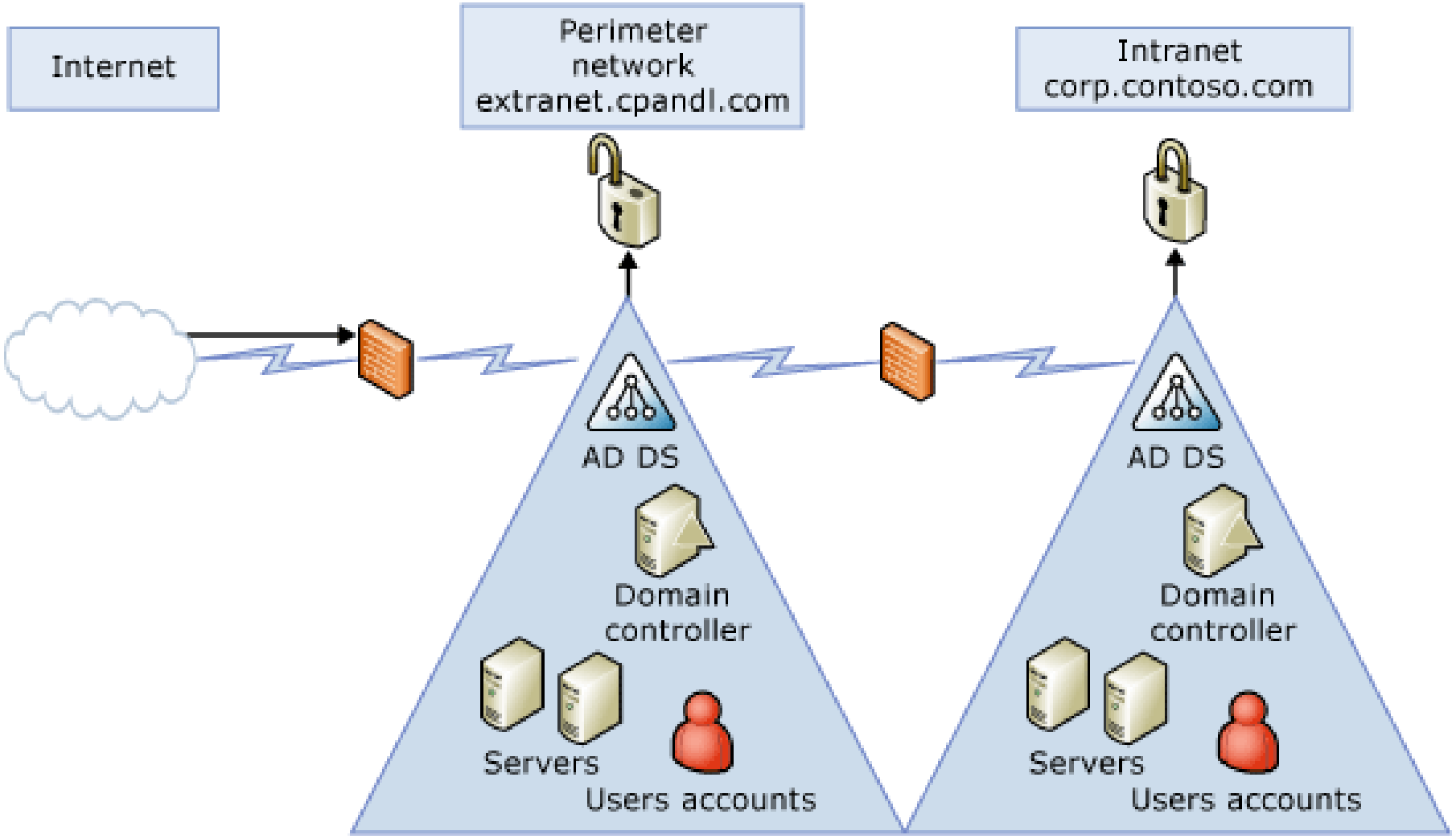
- What would happen if we placed a second DC server on the DMZ or in the branch office that fully replicates with the internal DC??
  - Attackers could penetrate that DMZ and then crack all of the passwords that were stored on the internal DMZ since they are replicated
  - Attackers could physically steal the DMZ DC or branch office DC and crack the passwords
- What other options do we have???

DMZ Scenario 1 – No AD



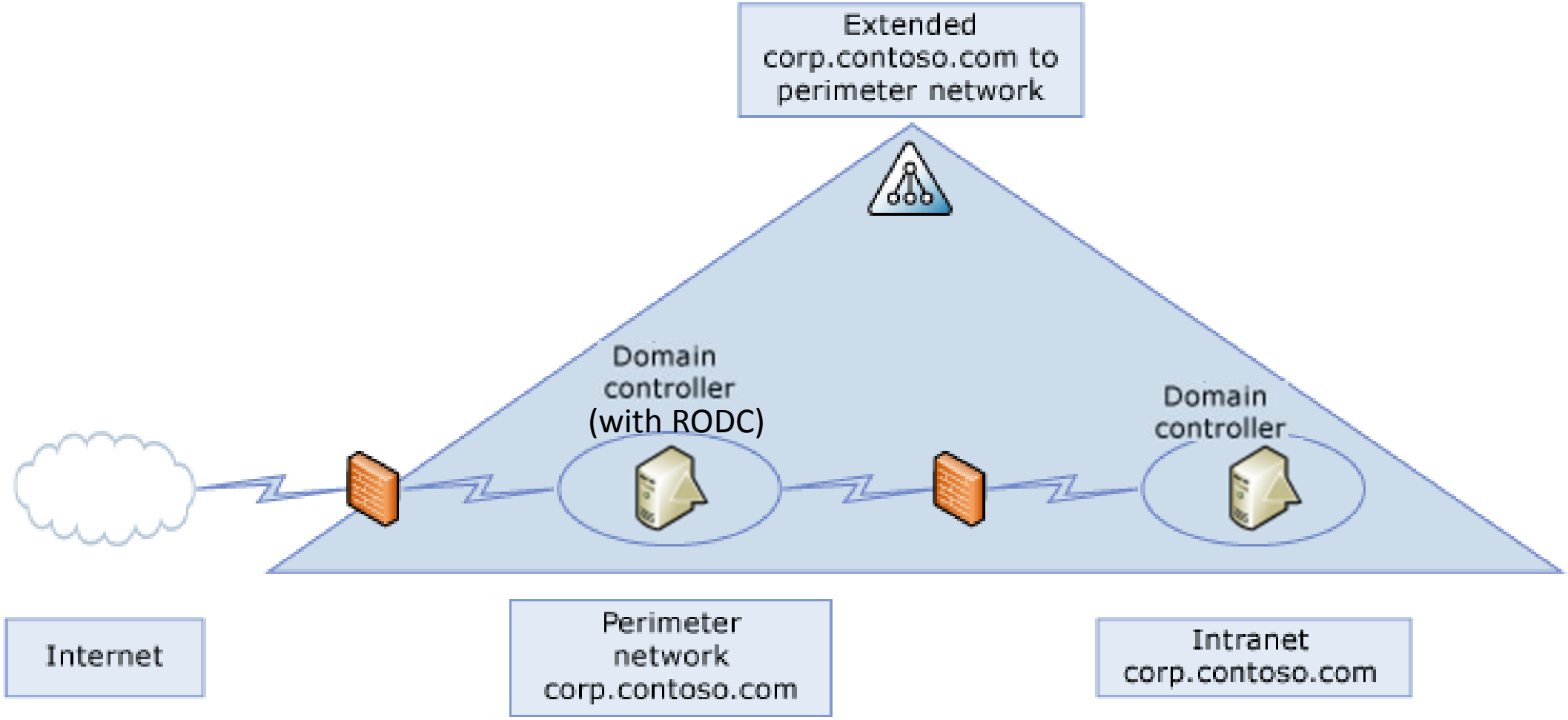
http://blog.windowsserversecurity.com/2011/04/20/active-directory-domain-services-in-the-perimeter-network-part-1/

DMZ Scenario 2 – Isolated Forest



http://blog.windowsserversecurity.com/2011/04/20/active-directory-domain-services-in-the-perimeter-network-part-1/

DMZ Scenario 3 – Extended Forest with RODC

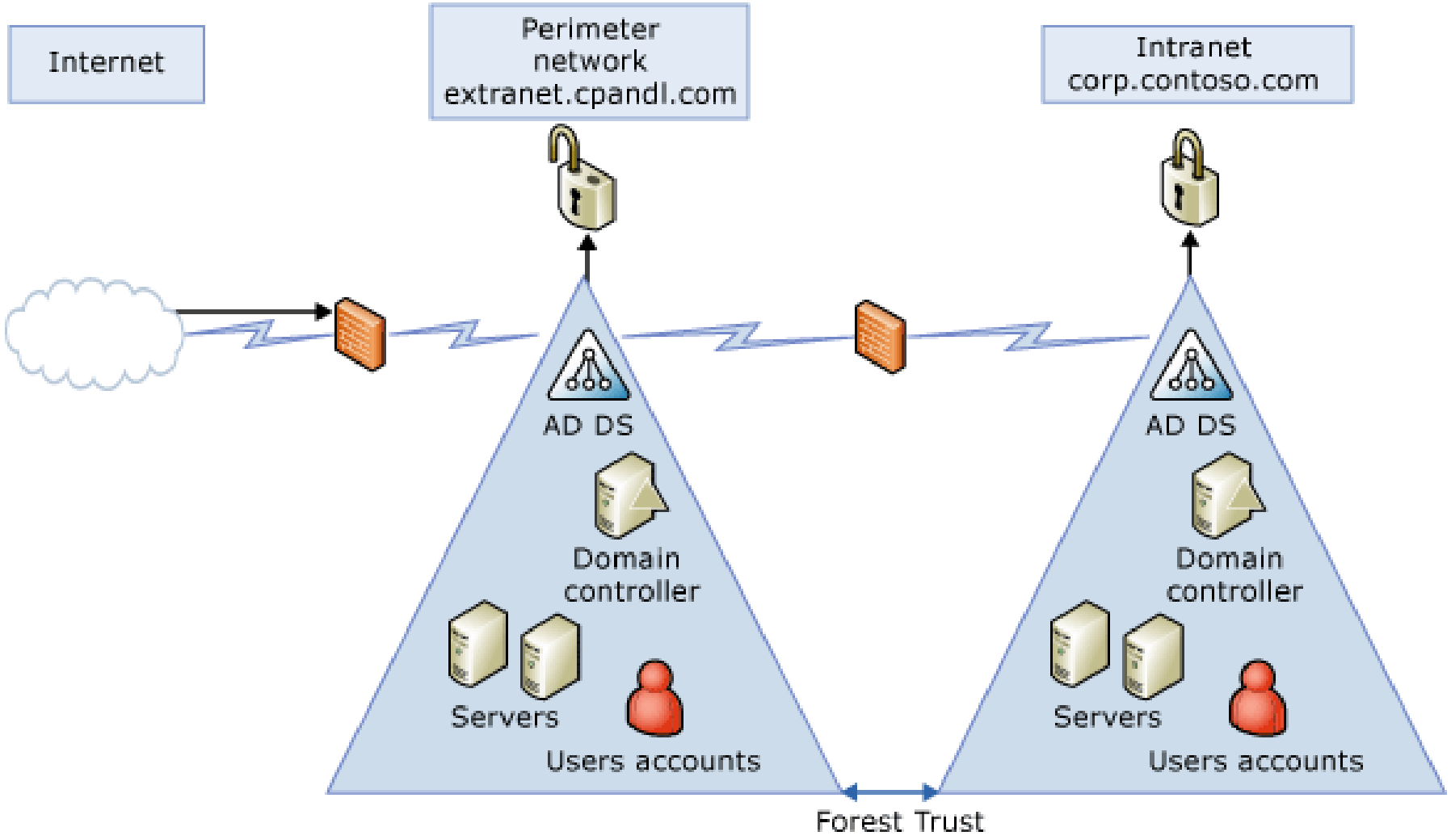


http://blog.windowsserversecurity.com/2011/04/20/active-directory-domain-services-in-the-perimeter-network-part-1/

Read-Only Domain Controller (RODC)

- If DMZ DC or branch office DC is stolen:
  - No internal user/computer passwords stored on RODC
    - You should disable password caching on RODC as well
  - Isn't writable and won't propagate changes back to internal DC

DMZ Scenario 4 – Forest Trust Model



http://blog.windowsserversecurity.com/2011/04/20/active-directory-domain-services-in-the-perimeter-network-part-1/

Forest Trust Model

- One of the best designs, but costly to administer
- Similar to Isolated Forest Model in that a unidirectional trust can be set between the two forests
  - Could let internal IT users remotely access DMZ web server
  - DMZ web server local users could not access internal network

Internal Network Authentication

- Now, let's look at a technology to allow internal logged on users to access services
  - Servers
  - Web Application
  - File Shares
  - Etc.

Kerberos Overview

- Initially developed at MIT
- Protocol for accessing network resources with tickets as opposed to having to locally store passwords for each resource
- Uses symmetric-key crypto and involves trusted 3rd party
- Can be used within Windows Active Directory (AD) Domain or UNIX/Linux Realm

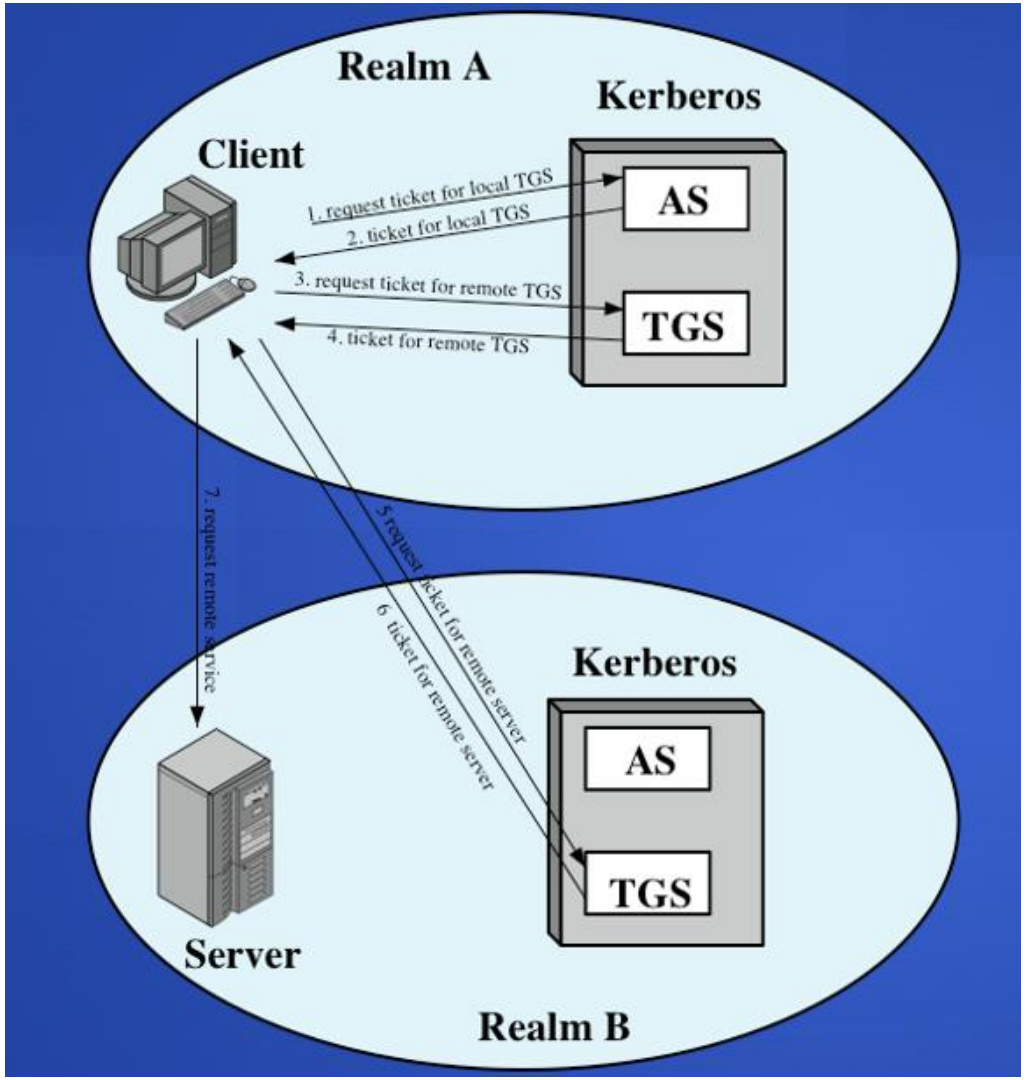
Kerberos Realm

- Network of clients and servers under a single administrative organization
- Kerberos environment consists of:
  - Key Distribution Center (KDC) contains:
    - User Database
    - Authentication Server (AS)
    - Ticket Granting Server (TGS)
  - Kerberos-Aware Application Servers
  - Clients registered with KDC

Multiple Kerberos Realms

- Must be mutual trust between Kerberos servers in different realms
- Each server must share a secret key

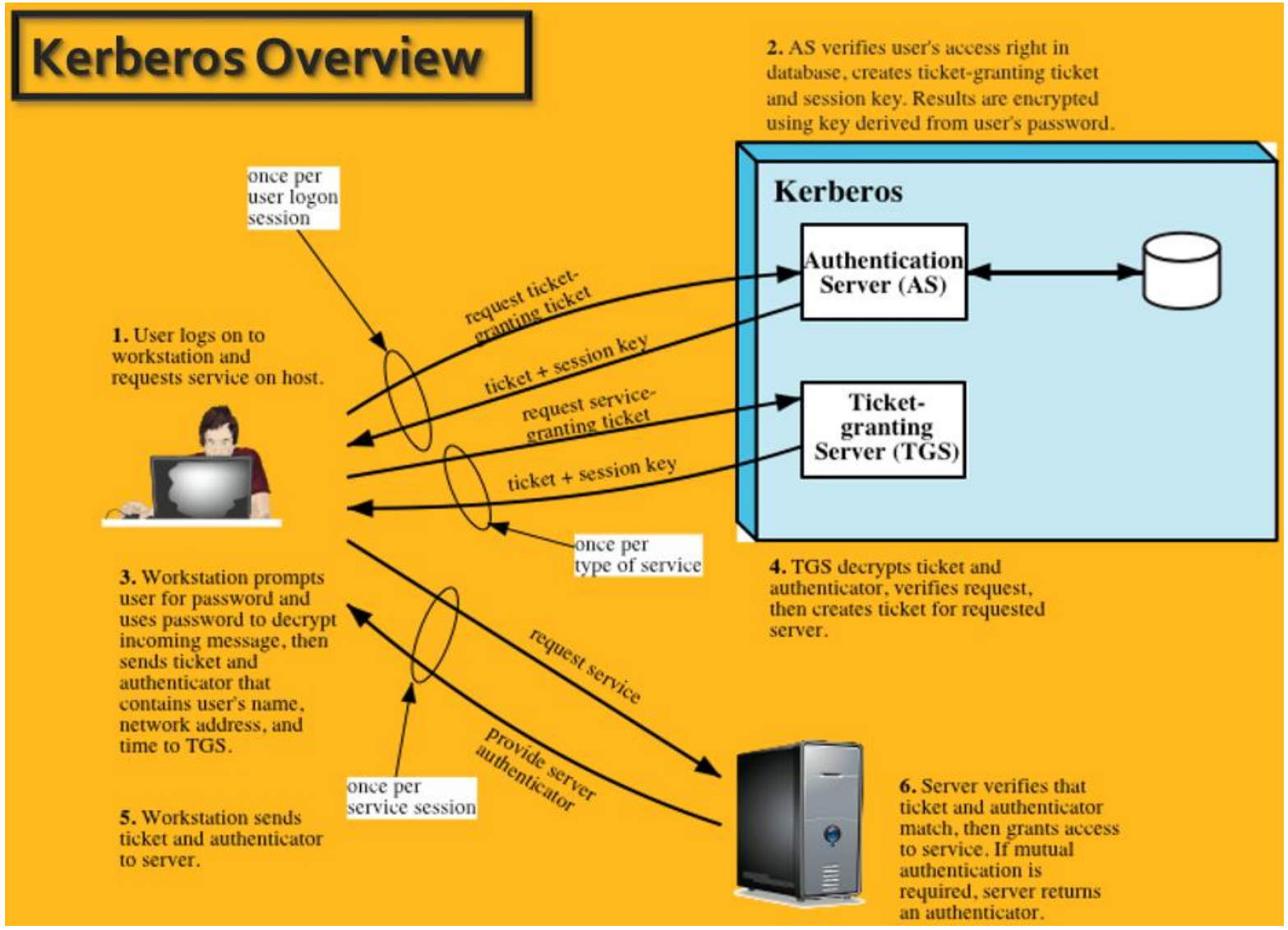
Kerberos Realms



Kerberos Requirements

- Database must have users and access rights entered
- All systems must use time synchronization to accurately time-stamp ticket expiration to prevent replay attacks
- AS needs to be able to send messages to Kerberos-aware Application Servers in a secure manner
  - Physically distribute secret keys to AS and the application servers on the realm/domain

Kerberos Process Diagram



Kerberos Process

1. User logs on to client device locally and sends request to the KDC to access to a Kerberos-aware Application Server
2. AS checks its database to find password of the requesting user
3. AS creates a ticket-granting ticket (TGT) and encrypts it using the secret key that was previously distributed physically to AS
4. AS also creates a one-time session key

Kerberos Process (Cont.)

4. Both the TGT and the one-time session key are additionally encrypted using an encryption key derived from the user's password into a message
5. The message is then sent back to the user's client device
6. Client device prompts user for his or her password to generate the key to decrypt the message and recover the TGT (that remains encrypted) and the session key
   - TGT can now be used by client for entire logon session to request additional tickets to access application servers

Kerberos Process (Cont.)

- Note:
  - TGT contains:
    - User's ID
    - AS' ID
    - Timestamp (prevents replay attacks)
    - Expiration (Might be 8-10 hours)
    - Copy of same session key that was attached in message
  - User (or an attacker) cannot decrypt or modify the TGT since it was encrypted with the secret key that only the AS and application servers have
  - Remember, TGT doesn't allow access to application server

Kerberos Process (Cont.)

- Note:
  - At this point the attacker could just sniff the network and capture a copy of the TGT and wait for the legitimate user to sign out
  - Attacker could then spoof the victim's IP address and send a request to the TGS with the TGT to access an application server.
  - What is stopping this from working???
    - The encrypted session key that the AS originally created, shared with the client which shared with the TGS is not known to the attacker (unless they have the user's password or credential)

Kerberos Process (Cont.)

- Now, the user can use the TGT to request service granting tickets from the TGS to access application servers throughout the logon session

7. Client device sends message (that includes application server ID) to the TGS along with an authenticator
   - Authenticator contains user ID and IP address and is encrypted with the session key
8. TGS decrypts the TGT with the secret key that was previously distributed to it
9. TGS decrypts the authenticator with the session key

Kerberos Process (Cont.)

10. TGS checks to make sure lifetime of TGT hasn't expired and compares user ID and IP address from decrypted authenticator against the user ID and IP address from the message to make sure they match
11. If they do, TGS is assured that sender of the ticket is indeed the ticket's real owner.
    - Note, this doesn't prove anyone's identity but is a secure way to distribute keys
    - Authenticator proves client's identify because it can only be used once and has a short lifetime

Kerberos Process (Cont.)

12. TGS now sends a service-granting ticket (that is encrypted with the secret key previously distributed) and a new session key to the client.
    - Entire message is encrypted with old session key so that client can recover the message.
13. Client sends service-granting ticket and authenticator (encrypted with new session key) to the application server or resource the user wants to access

Kerberos Process (Cont.)

14. If mutual authentication is required:
    - Server replies to client with value of timestamp from the authenticator, incremented by 1 and encrypted with the session key.
    - Client decrypts message with session key to recover incremented timestamp and ensure message was created by the server
15. Lastly, the client and application server share a secret key to be used to encrypt any future communications or to exchange a new session key

Kerberos Versions 4 and 5

- Version 4
  - Used DES for encryption
  - End of life
- Version 5
  - Supports AES
  - Authentication forwarding
    - Client can access a server and then that server can access another server on behalf of the client

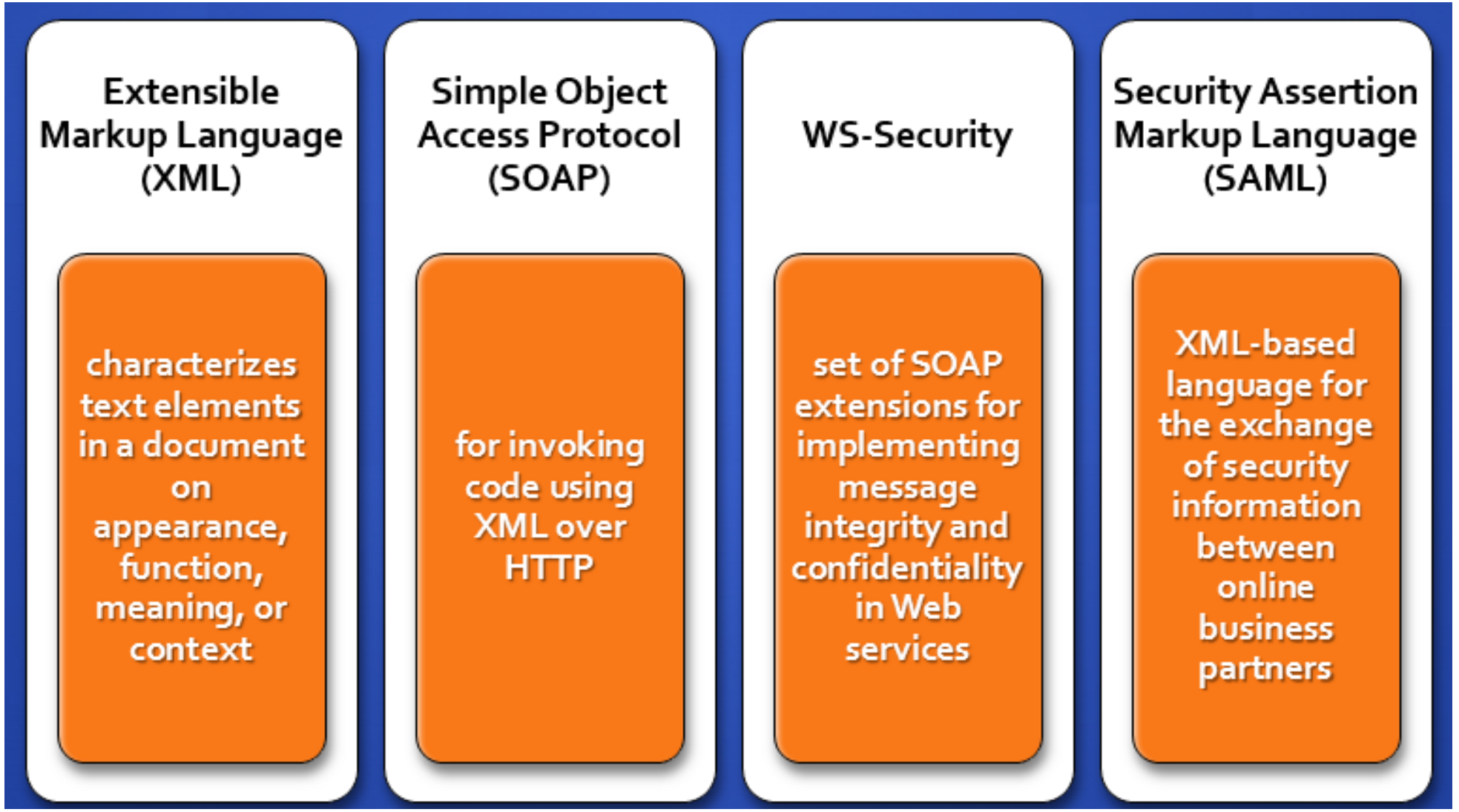
Kerberos Vulnerabilities

- Password guessing
- Availability
  - Failover KDC recommended
- DoS, Buffer Overflow, etc.
  - Keep patched

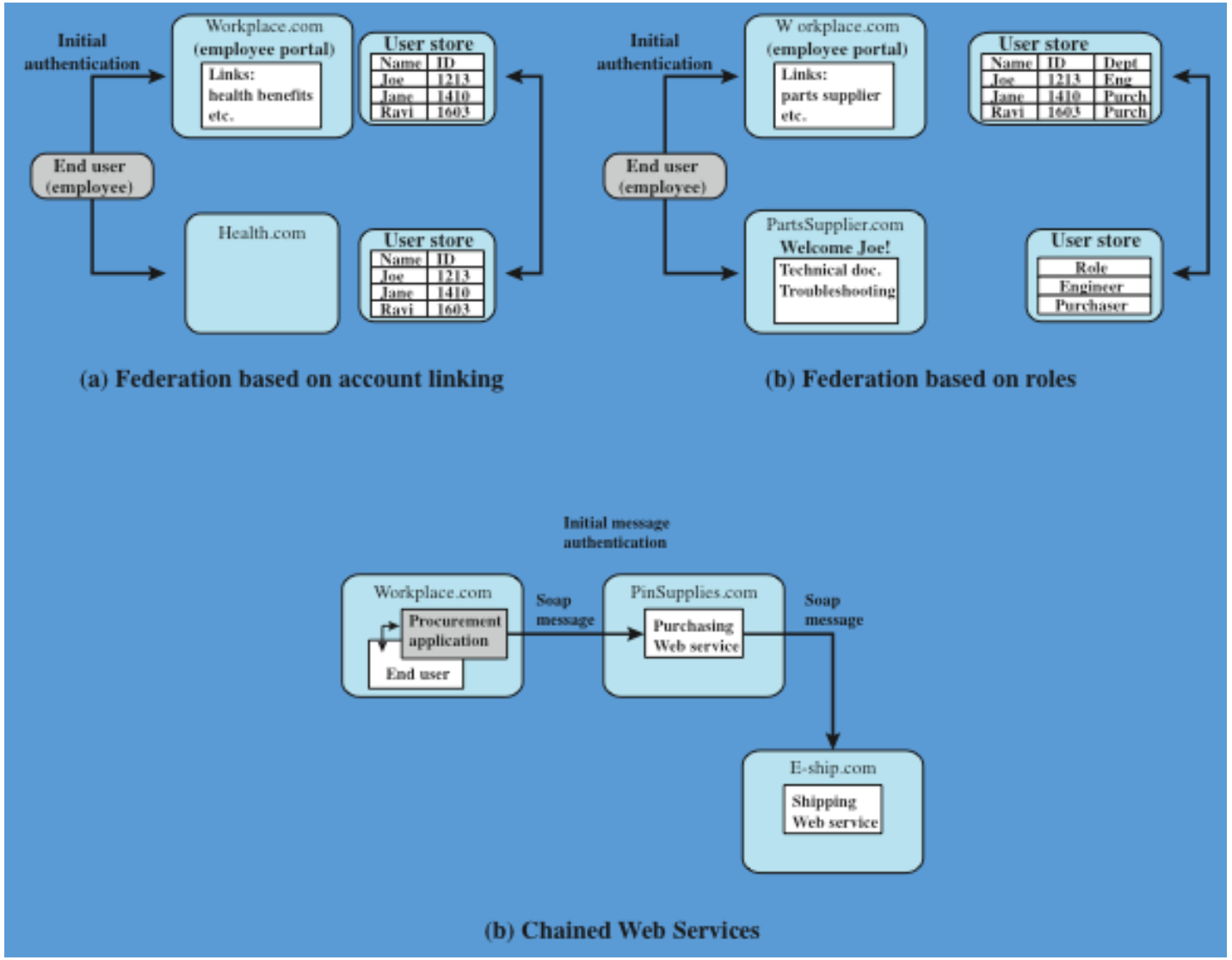
Federated Identity Management

- Common identify management scheme across multiple enterprises supporting even millions of users

Federated Identity Management Standards



Federated Identity Scenarios



Single Sign On (SSO)

- Subset of Federated Identity Management
- User can log on with single set of credentials to access multiple systems
- Generally authenticates user with Kerberos but authorizes user by using LDAP to access Active Directory
- Why would SSO enhance security???
  - Users only need to remember one set of credentials
  - Less likely they will be writing them down

Part IV

Access Control

Access Control

- After authentication, authorization is used to determine is the user has permission to access a resource
- Access Control are the measures taken to restrict access to a resource

Access Control Models

- Mandatory Access Control (MAC)
- Discretionary Access Control (DAC)
- Rule Based Access Control (RBAC)
- Role Based Access Control (RBAC)

Mandatory Access Control (MAC)

- Primarily used by govt. and system admin manages user access by using labels with classifications (top secret, confidential, etc.) and categories (department, project, etc.)
- Example:
  - SELinux – Developed by NSA

Discretionary Access Control (DAC)

- Users control access to own data
- Uses Access Control Lists (ACLs)
- Examples:
  - Windows NTFS
    - Uses UAC with administrator account for highest access
    - Permissions inherited by applications users are accessing
    - Malware can take advantage of this trait
  - Linux File Permissions and ACLs
    - Uses sudo with root account for highest access

Rule Based Access Control (RBAC)

- Access based on approved policies configured by rules
- Example:
  - Rules on a firewall or IDS/IPS

Role Based Access Control (RBAC)

- Uses roles to manage access rights and permissions for users
- All users in same role could have same access rights and permissions
- Examples:
  - Microsoft Project Server
    - Admins, Executives, Project Mgrs, Team Members, etc.
  - Microsoft Active Directory
    - Admins, Backup Operators, Server Operators, etc.

Part V

Linux Access Control Hand's On

Linux Hand's On

- Open up a terminal in Kali
- We have used the **useradd** command in here before but there are more detailed options available

useradd Options

- **-c** "*comment*"
  Typically user's full name.
- **-d** *home_dir*
  Home directory for user.
- **-D**
  Save supplied information as default information.
- **-e** *expire_date*
  Account expiration date (YYYY-MM-DD).
- **-f** *days_till_expired*
  - Number of days till account expires.
  - **-1** disables this option
  - **0** disable accounts after account expired.

More **useradd** Options

- **-g** *group*
  User account's primary group.
- **-G** *grouplist*
  User account's supplemental groups.
- **-k** *skel_dir*
  Directory containing initial configuration files & login scripts.
- **-m**
  Automatically create home directory & copy skeleton files.
- **-M**
  Do not create new user home directory.
- **-n**
  Do not create a new group that matches user name.

Even More **useradd** Options!

- **-o**
  - Used with **-u** option
  - Creates account with same UID as another account.
- **-p** *password*
  Enter an encrypted password for account.
- **-s** *shell*
  Specify command shell for account.
- **-u** *user_id*
  Specify the UID for the account.

Linux Hand's On

- First let's see what groups are currently available
- **cat /etc/group**
- Now let's create a group called sales
- **groupadd sales**
- Run the prior cat command to see if the group was created successfully
- Let's also see what the useradd defaults are
- **useradd -D**

Linux Hand's On

- Figure out how to do the following in one **useradd** command:
  - Create a user called bduggan
  - Automatically create their home directory
  - Set their password as secretpass
  - Set an expiration date of 2015-11-30
  - Set their supplemental group to sales

- Hint: Use **man useradd**

Linux Hand's On

- **useradd bduggan –m –p secretpass –e 2015-11-30 –G sales**
- Let's make sure bduggan is in the bduggan primary group (by default) and the sales group
- **groups bduggan**
- Let's verify that a home directory was created for bduggan
- **ls /home**

Linux Hand's On

- Now, let's verify that the account expiration was set for Nov 30, 2015
- **chage –l bduggan**
- Notice that there is no password expiration
- Remember, we noticed there is no password expiration set by default:
- **useradd –D**

Linux Hand's On

- Use the **chage** command to set the following options for bduggan (use the man page)
  - Minimum password age is 5 days
  - Maximum password age is 60 days
  - Account is locked out 5 inactive days after password expiration
  - User is warned 2 days before password expiration
  - Change explicit expiration date to 2016-11-30

Linux Hand's On

- **chage -m 5 -M 60 -I 5 -W 2 –E 2016-11-30 bduggan**

```
root@KLY-IR105:/usr/share/john# chage -l bduggan
Last password change                                    : Nov 15, 2015
Password expires                                        : Jan 14, 2016
Password inactive                                       : Jan 19, 2016
Account expires                                         : Nov 30, 2016
Minimum number of days between password change          : 5
Maximum number of days between password change          : 60
Number of days of warning before password expires       : 2
```

- Can use -1 after -E to remove an explicit account expiration date

Linux Hand's On

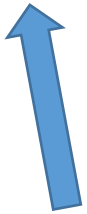
- The **usermod** command can be used to modify pre-existing user accounts
- The **userdel** command can be used to delete an account. The **-r** option will also remove the home directory and contents

Linux Hand's On

- Switch to bduggan's account
- **su bduggan**
- **cd ~**
- **pwd** and verify you are in /home/bduggan
- **touch testfile1** to create an empty file
- **ls -l**

Linux Hand's On

- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1



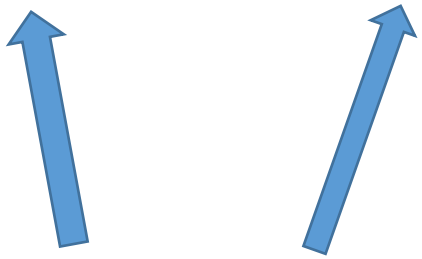
File Type:

- indicates test6 is a regular file

d would indicate a directory (ls –lr)

Linux Hand's On

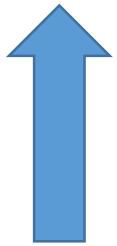
- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1



File Permissions (9 bits) rw- r-- r--

Linux Hand's On

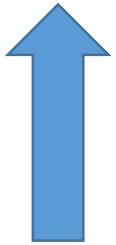
- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1



Owner of the testfile1 file

Linux Hand's On

- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1



Group assigned to the test6 file

Linux Hand's On

- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1

Linux Hand's On

- rw- r-- r-- 1 bduggan bduggan 0 Nov 15 00:33 testfile 1

Understanding File Permissions & Ownership

- Permissions consist of nine bits
- Groups of three bits each:
  - first three bits apply to owner user's permission
  - middle three bits apply to group assigned to file
  - last three bits apply to all others (everyone else)
- Only the owner of the file can change permissions of it

File Permission Bit Assignment

rwx rwx rwx

User Group Others

(User is essentially the “owner” of the file)

File Permission Types

- r = read permission
- w = write permission
- x = execute permission

Understanding Linux rwx Permissions

Permission	File	Directory
Read	View what's in the file.	See what files and subdirectories it contains.
Write	Change the file's content, rename it, or delete it.	Add files or subdirectories to the directory. Remove files or directories from the directory.
Execute	Run the file as a program.	Change to the directory as the current directory, search through the directory, or execute a program from the directory. Access file meta data (file size, time stamps, and so on) of files in that directory.

Permission Bits Can Be Set:

1. Via Letters
   - Handy for changing permissions recursively since you can change bits selectively.
2. Via Numbers
   - Good for changing all permission bits at once.

1. Changing Permissions Via Letters

- Change permissions via letters with **chmod** command
  - Turned on with plus (+) and off with minus (-)
  - Bit sets are designated by
    u = user g = group o = other a = all users

(u = user is really the “Owner”)

Changing Permissions Via Letters (cont.)

- Example 1:
  - The file, testfile1, has the current permissions:

    rw- r-- r--
  - We want to grant execute to user, group, and other
  - **chmod a+x testfile1**
  - **ls -l** to view permission change

  - The file, testfile1, now has the permissions:

    rwx r-x r-x

Changing Permissions Via Letters (cont.)

- Example 2:
  - The file, testfile1, has the current permissions

    rwx r-x r-x
  - We want to take away the group execute permission
  - **chmod g-x testfile1**
  - **ls -l**
  - The file, apple, now has the permissions,

    rwx r-- r-x

2. Changing Permissions Via Numbers

- Each permission is assigned a number.
  r = 4 w = 2 x = 1 no permission = 0

- To determine the number for each bit set, the numbers are added.
  Example: File owner has rw- permissions or 6
  because 4+2+0=6

Permission Numbers (cont.)

- Bit sets for owner, group, & others are combined

- Example: rwx rw- r--

	owner	group	other
because	(4+2+1)	(4+2+0)	(4+0+0)
Permissions =	7	6	4

Examples:

	r--	r-x	rwx
Permissions =	4	5	7
	rw-	rw-	--x
Permissions =	6	6	1
	r-x	---	-wx
Permissions =	5	0	3

chmod Example: Via Numbers

- Change permissions numerically with **chmod** as well
  - **chmod 777 testfile1**
  - Changes permissions to rwx rwx rwx

Changing File Ownership

- regular user - cannot change ownership of a file or directory belonging to another user.
- root user - can change ownership of a file or directory belonging to another user.

- Type **exit** to return to root user
- **cd /home/bduggan**

Changing File Ownership

- **ls –l**

```
-rwxrwxrwx 1 bduggan bduggan 0 Nov 15 00:33 testfile1
```

- To change owner of a file or directory:
  chown *username filename*
- Let's change the owner of testfile1 to the root user
- **chown root testfile1**
- **ls -l**

```
-rwxrwxrwx 1 root bduggan 0 Nov 15 00:33 testfile1
```

Changing File Ownership (Cont.)

- To change user and group of a file or directory chown *username:groupname filename*
- Let's change the owner to bduggan and group to sales:
- **chown bduggan:sales testfile1**
- **ls -l**

```
-rwx rwx rwx  1 bduggan sales 0 Nov  15 00:33 testfile1
```

Permissions vs. ACLs

- In addition to permissions, Linux also allows the setting of Access Control Lists
- If regular users can use chmod to change permissions on file/directories they own, why use ACLs???

Permissions vs. ACLs

- Create a user named jerry with **adduser jerry**
- Choose jerrypass as password and Enter through defaults and choose y at the end
- You should still be in /home/bduggan
- **su bduggan**
- Let's change the perms on testfile1
- **chmod 700 testfile1**

```
-rwx------ 1 bduggan sales 0 Nov 15 00:33 testfile1
```

Permissions vs. ACLs (cont.)

- Let's say jerry isn't a part of any groups but bduggan needs him to be able to read the previously mentioned file. (jerry currently falls under the "everyone else" category which does not have read permission on testfile1)
- bduggan can simply set an ACL to allow jerry to read the file but no one else!

View ACL Permissions with **getfacl**

- To see full current ACL permissions on file, use **getfacl testfile1**

```
$ getfacl testfile1
# file: testfile1
# owner: bduggan
# group: sales
user::rwx
group::---
other::---
```

Setting ACLs with **setfacl**

- Syntax: setfacl -m u:*username*:*perms filename*
- Explanation:
  - m indicates setting an ACL.
  - u indicates setting ACL permission for a user.
  - *username* is the account to assign the ACL permission.
  - *perms* are the ACL permissions, options allowed read(r), write(w), execute (x).
  - *filename* is the file setting ACL permissions on.

Managing ACLs with **setfacl**

- **setfacl –m u:jerry:r testfile1**
- **ls -l**
  - A plus sign (+) indicates ACL permissions are set.

```
-rwxr-----+ 1 bduggan sales 0 Nov 15 00:33 testfile1
```

Managing ACLs with **setfacl**

- **getfacl testfile1**

```
$ getfacl testfile1
# file: testfile1
# owner: bduggan
# group: sales
user::rwx
user:jerry:r--
group::---
mask::r--
other::---
```

Even though jerry is not in the owner group, he can still read this file due to the ACL

Part VI

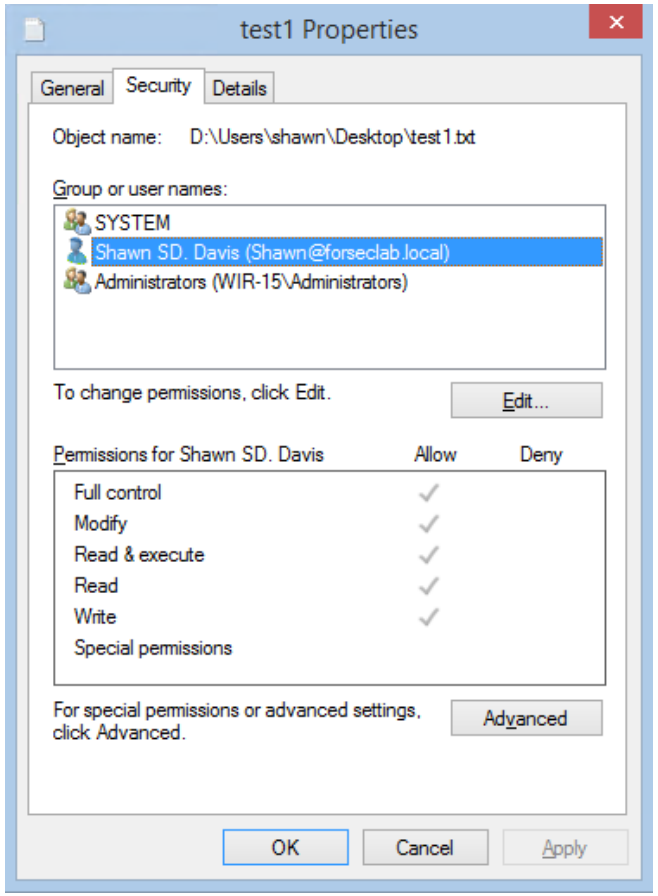
Windows Access Control Hand's On

Windows Hand On

- Go back to your RADISH Win 8.1 VM
- We will cover briefly granting permissions on a file
- Create a file named test1 on your desktop
- Right click on it and select “Properties”
- Choose the “Security” tab

Windows Hand On

- Select your user to see their “Allow” permissions:

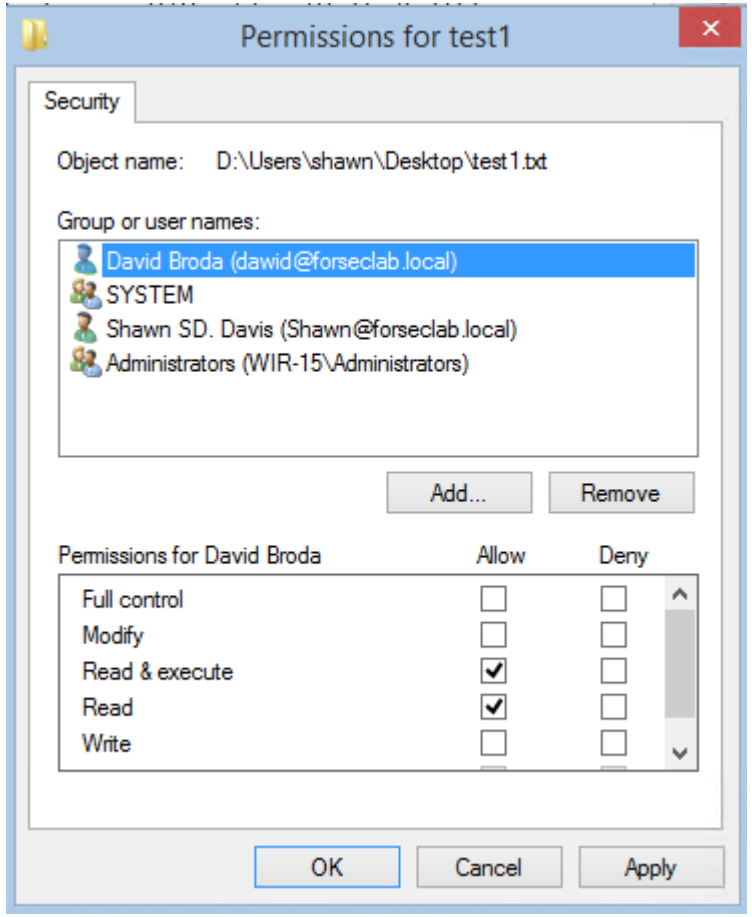


Windows Hand On

- Select “Edit”
- Select “Add” to grant a new user permissions to the file
- Select “Advanced”
- Type Dawid in the “Starts with” text box and select “Find Now”
- Select Dawid Broda at the bottom and hit “OK”
- Hit “OK” again

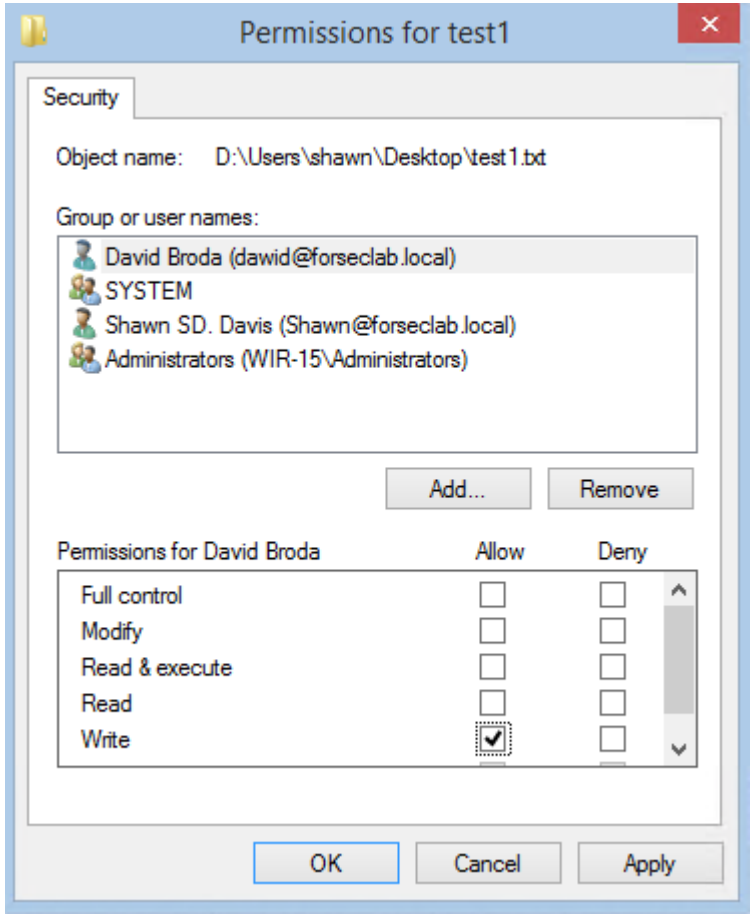
Windows Hand On

- Select David to see his “Allow” permissions:



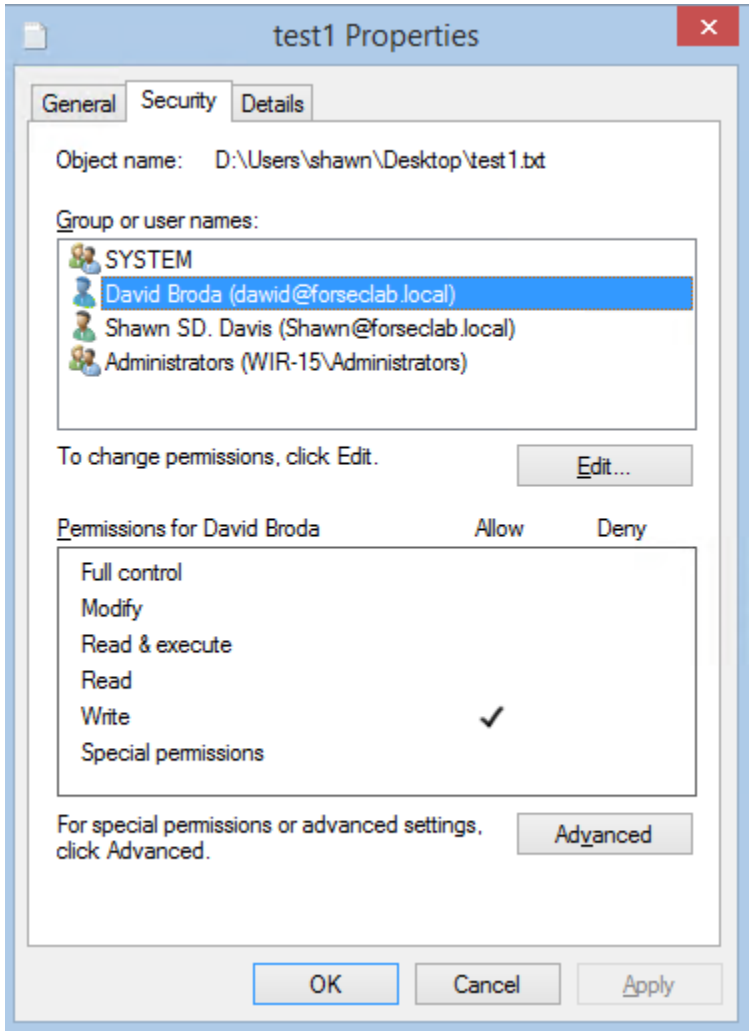
Windows Hand On

- Change permissions so he can only write to the file and hit “OK”



Windows Hand On

- Select “David” again
- Hit “OK” to save the changes



Windows Permissions

Permission	Meaning for Folders	Meaning for Files
Read	Permits viewing and listing of files and subfolders	Permits viewing or accessing of the file's contents
Write	Permits adding of files and subfolders	Permits writing to a file
Read & Execute	Permits viewing and listing of files and subfolders as well as executing of files; inherited by files and folders	Permits viewing and accessing of the file's contents as well as executing of the file
List Folder Contents	Permits viewing and listing of files and subfolders as well as executing of files; inherited by folders only	N/A
Modify	Permits reading and writing of files and subfolders; allows deletion of the folder	Permits reading and writing of the file; allows deletion of the file
Full Control	Permits reading, writing, changing, and deleting of files and subfolders	Permits reading, writing, changing and deleting of the file

https://msdn.microsoft.com/en-us/library/bb727008.aspx

Windows Permissions

- Read is the only permission needed to run scripts. Execute permission doesn't matter.
- Read access is required to access a shortcut and its target.
- Giving a user permission to write to a file but not to delete it doesn't prevent the user from deleting the file's contents. A user can still delete the contents.
- If a user has full control over a folder, the user can delete files in the folder regardless of the permission on the files.

https://msdn.microsoft.com/en-us/library/bb727008.aspx

Windows Note

- Additional access control can be configured:
  - Locally through Group Policy Objects (GPO) with lusrmgr
  - With the Group Policy Management Console (GPMC) on a domain Server
- Both are outside of the scope of this course

Homework

- Homework 13 due on Sunday 4/17 before midnight
- Remember that your Individual Project PowerPoint slides and Videos are due before class next week (uploaded to Blackboard before 5:30pm)