Network Security

Course notes

Version 2013.1

Contents

Chapter 1

Firewalls

A firewall is a security control designed to prevent unauthorized access from an external network to the internal (networked) system.

Operation:

- All traffic from inside to outside, and vice versa, must pass through the firewall.
- The firewall allows only authorized traffic to pass through (as defined in the security policy).

Capabilities (both security and non-security):

- Filter traffic.
- Audit traffic.
- Router functions, like NAT, IPsec, VPN.

How to bypass the firewall:

- Uncontrolled network access, like dial-in/out, wireless access (extra problem: rogue access points).
- Physically intruding the perimeter of the system.
- Use of portable devices (PDA, laptop, USB stick)

Filtering strategies:

- Packet filtering

  An IP packet contains: source address, destination address, port number, protocol, etc. Use rules (similar to p278 of the text book) to decide on blocking or allowing (for an explanation of the rules see the text book).

no.	**action**	**prot**	**opt**	**source**	**port**	**dest**	**port**	**comment**
1	block	*	–	*	*	SPIGOT	*	We don't trust them.
2	block	*	–	SPIGOT	*	*	*	We don't trust them.
3	allow	SMTP	–	internal	25	*	*	From our SMTP port
4	allow	SMTP	–	*	*	internal	25	To our SMTP port

  Use *default rule* (can be block or allow) to decide on packets for which there is no rule.

Advantage: simple

Disadvantages:

- Work at IP level, cannot examine upper-layer data, so cannot block application specific commands.
- Limited logging information (source, destination, port number etc.).
- Address spoofing attacks.
- Hard to fully cover organization's security policy.

- Stateful inspection

  Packets are part of an existing stream of traffic (TCP connection). The firewall is able to determine whether a packet is either the start of a new connection, a part of an existing connection, or an invalid packet.

- Application-level Gateway (also known as application layer firewall or application proxy)

  Inspect packets at the level of the application (ftp, telnet, http). Can be used to block websites, viruses, etc.

1.1 Location of a firewall

The firewall can be located:

- Personal firewall (on a computer)
- Between internal and external network (on the gateway)
- Double firewall separating internal network from *Demilitarized Zone* (DMZ) from external network. The DMZ contains the "services" provided by the company to the external network, such as web servers, mail, DNS. (See fig 9.3 p287 of the text book).
  The DMZ is usually treated as no man's land: dangerous for the inside network (viruses/malware in DMZ), and the inside network poses a danger to it (attacks from inside to DMZ). Hence, separate DMZ from internal network with firewall. Similarly, separate DMZ from outside network with firewall.

Chapter 2

Intrusion Detection

Two main sources of attacks on a system:

- intruder
  - internal (misuse of privileges)
  - external (penetrator)
- malware: see Chapter **??**

Intrusion Detection Systems (IDS) / Intrusion Prevention Systems (IPS) provide a “first line of defense”: they will reduce intrusions but definitely not eliminate them all.

We can categorise the attacker using his skills and reasons to attack:

- Newbie/tool kit user (no specialized knowledge, use of-the-shelf hacking tool).
  E.g.: Security testing firms, curious students.

- Hacker (fun, showing weaknesses, increase status). Note: even if the hacker does not damage the system/data, the company may make a lot of cost to recover from the incident (direct, as well as indirect costs).
  E.g.: Kevin Mitnick.

- Criminals (for profit): hacking is becoming an industry with specializations and a market for hacking services.
  E.g.: You can buy valid credit card numbers online.

- Cyber terrorists (for damage, intimidation).
  E.g.: Anonymous.

- Insider, e.g. disgruntled employee.
  E.g.: Terry Childs (see a.o. Wikipedia).

Using this, we can tailor the Security Controls to the appropriate intruder. Example security controls are:
IPS/IDS, logs, principle of least privilege, authentication, managerial (close down accounts of ex-employees), back up, ...

2.1 Concepts of Intrusion detection

Notion 1 *Intrusion detection =*
A security service that monitors and analyzes *system* events *for the purpose of finding, and providing (real-time) warning of* attempts to access system resources *in an* unauthorized manner.

Logical components of IDS:

- sensor: collect data
- analyzers: determine if intrusion has occurred
- user interface: view output and control system

Often, Intrusion Detection is not a yes/no question. The system (and operator) have to decide given an event whether it is regular behaviour or intruder behaviour (see figure p182 text book).

Four different situations in detecting a possible attack.

	Positive	Negative
True	a real attack triggers an alarm	real attack, but no alarm
False	a regular event triggers an alarm	regular event and no alarm

Events
IDS base their operation on the observed "events", which can contain:

- Subject: initiator of action (e.g. sjouke).
- Action: which operation (e.g. login, read).
- Object: on which the action is performed (e.g. file, program, printer).
- Exception-condition: is the action performed successfully? (e.g. attempt to read a file which is not readable).
- Resource usage: amount used of some resource (e.g. processor time, number of pages printed).
- Time-stamps: time at which the action took place.

Example: from p185.
Consider the following command: `COPY GAME.EXE TO <library>GAME.EXE`
This command can give rise to the following three events:

Subject	**action**	**object**	**exception**	**resouce**	**time-stamp**
Smith	execute	<library> `COPY.EXE`	0	CPU = 000002	11012345
Smith	read	<Smith> `GAME.EXE`	0	RECORDS = 0	11023456
Smith	execute	<library> `COPY.EXE`	write-viol	RECORDS = 0	11034567

2.2 Approaches to Intrusion Detection

Two approaches to detect problems:

- Anomaly detection: find abnormal pattern/behavior.
  - Threshold detection: determine threshold for frequency of occurrence of various events.
  - Profile based: determine profile for each user and detect changes.
- Signature detection: define which behavioral patterns are typically those of an intruder.

Example of measures for anomaly detection (from p188 of the book).

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 to break-in by password guessing
Failures to login from specified terminals	Operational	Attempted break-in
	Program Execution Activity	
Execution frequency	Mean and standard deviation	May detect intruders, who are likely to use different command, or a successful penetration by a legitimate user, who has gained access to privileged commands.
Program resource utilization	Mean and standard deviation	An abnormal value might suggest injection of a virus or Trojan horse, which performs side effects that increase I/O or processor utilization.
Execution denials	Operational model	May detect penetration attempt by individual user who seeks higher privileges.
	File Access Activity	
Read, write, create, delete frequency	Mean and standard deviation	Abnormalities for read and write access for individual users may signify masquerading or browsing.
Records read, written	Mean and standard deviation	Abnormality could signify an attempt to obtain sensitive data by inference and aggregation.
Failure count for read, write, create, delete	Operational	May detect users who persistently attempt to access unauthorized files.

Example of heuristics for signature detection:

1. Users should not read files in other users' personal directories.
2. Users must not write other users' files.
3. Users who log in after hours often access the same files they used earlier.
4. For users, access to disks happens via the operating system's (disk) drivers.
5. Users should not be logged in more than once to the same system.
6. Users do not make copies of system programs.

2.3 Base Rate Fallacy

Notion 2 (Fallacy) *A fallacy is a "false or misleading belief/notion."*

Any intrusion detection system is an imperfect test of the system – it will fail to detect some malicious events (false negatives) and detect some benign events as being malicious (false positives).

Any imperfect test suffers from the base rate fallacy: if the base rate of the events that are to be detected is very low, detection error can make the test useless.

a. Low base rate b. High base rate

Figure 2.1: Low vs. high base rate of event.

Consider, for example, the two base rates shown in Figure 2.1. In Figure 2.1a, we see a situation where there are few events which we are looking for (depicted in red). Figure 2.1b shows a situation in which most events are the ones we are interested in.

Now suppose we use a test which is less than perfect: for a "red" event, there is a 90% chance the test results in "red", while for a "blue" event, there is a 90% chance the test results in "not red". Since the test is not perfect, it will make errors. But if the base rate is very low, then the amount of errors it makes can be (far) greater than the amount of correct results it gives.

Example 2.3.1 (Testing for red with different base rates) *Consider there are 99 "blue" events for each "red" event (i.e. the base rate of "red" events is 1%).*

- *Of every 100 events, 99 are "blue".*
- *Of these 99, 9.9 will be tested as "red".*
- *So, at best, testing 100 events will result in 10 events labelled "red", **only one of which is actually red**.*

So, in this setting, the test (which is 90% accurate), is only correct **10%** ***of the time(!)**.*

If the base rate is closer to 50%, things change:

- *Of every 100 events, 50 are "blue" and 50 are "red".*
- *Of the blue events, 5 will be tested as "red".*
- *Of the red events, 45 will be tested as "red".*

- *So, testing 100 events will result in 50 events labelled "red", of which 45 events (90%) are actually red!*

Finally, consider a situation like Figure 2.1b: there are 99 "red" events for each "blue" event. In this case:

- *Of every 100 events, 1 is "blue" and 99 are "red".*
- *At worst, the one blue event will be mistakingly labelled "red" – but this only happens 10% of the time.*
- *Of the 99 red events, 89 will be tested as "red".*
- *So, worst case, testing 100 events will result in 90 events labelled "red", of which 89 events (98.89%) are actually red!*

Note that in this last setting, the test is good at finding red events, but not good at finding 'not red" ones – most of the not-red labels are incorrect.

As the above example illustrates, the performance of a non-perfect test depends on the *incidence* of the tested-for event – that is, the base rate of the event. Depending on the base rate, the *exact same test* can perform very well (98.89% correct) or rather bad (10% correct).

This is further illustrated by the below example.

Example 2.3.2 (Base Rate Fallacy) *Consider regular (R) and intruder (I) events. Suppose an IDS will ring an alarm for 99 out of 100 I events. For only 1 out of 100 R events it will ring a (false) alarm.*

Suppose further that we have a computer system on which 1,000,000 R events take place in an hour plus 100 I events.

We observe an alarm from the IDS. What is the chance that this event is an intruder event?

Answer: The system will ring an alarm for 99 of the I events and for 10,000 of the R events. Given an alarm, the chance that this is an intruder event is 99/10,099 which is roughly 1 percent. So the chance of a false alarm is 99 percent.

In conclusion: to understand how well a not-perfect test will perform in the real world, knowing only its accuracy does not suffice. In addition, we need to have an idea of the ratio of positive vs. negative events.

False positives/false negatives

An event that tests positively, but is "negative", is called *false positive* – the test classifies this event as "positive", but that is false. Similarly, a *false negative* is an event which the test classifies as "negative", while it is actually positive. For example, for a spam filter, we have:

	not spam	*spam*
tagged "not-spam"		false negative
tagged "spam"	false positive	

Calculations taking base rate into account

This section illustrates (by means of an example) how to answer questions related to the base rate fallacy.

Example 2.3.3 (Base Rate Fallacy calculations) *Consider a spam filter that has an accuracy of 99%: 99% of all spam messages is tagged as spam, while 99% of not-spam is not tagged. We let this filter tag 1,000,000 messages, of which 5% is actually spam, and the rest isn't. With this, we can compute the answer to questions such as the following:*

- *How many spam mails are there in reality?*
- *How many spam mails will be tagged as spam?*
- *How many* non-*spam mails will be tagged as spam?*
- *How many spam mails will be tagged as* non-*spam?*
- *etc.*

To answer this, we fill in the following table:

	not spam	*spam*	***sum***
not tagged			
tagged			
sum			***1,000,000***

Since 5% of all mail is spam, we know that there are 5% $\times 1,000,000 = 50,000$ *spam mails in the entire collection. Hence, the remaining* $1,000,000 - 50,000 = 950,000$ *mails are not spam.*

	not spam	*spam*	***sum***
not tagged			
tagged			
sum	*950,000*	*50,000*	***1,000,000***

Moreover, we know that 1% of the spam mails will not be tagged, therefore 1% $\times 50,000 = 500$ *spam mails will not be tagged. The remaining* $50,000 - 500 = 49,500$ *spam mails will be tagged.*

Similarly, 1% of non-spam mails will *be tagged. Hence, 1%* $\times 950,000 = 9,500$ *non-spam mails will be tagged. The remaining* $950,000 - 9,500 = 940,500$ *mails won't be tagged. If we fill in these numbers, the table looks like this:*

	not spam	*spam*	***sum***
not tagged	*940,500*	*500*	
tagged	*9,500*	*49,500*	
sum	*950,000*	*50,000*	***1,000,000***

Now we can simply sum the rows, and find that there are $940,500 + 500 = 941,000$ *mails that are not tagged, and the remaining* $9,500 + 49,500 = 59,000$ *mails are tagged.*

In this example, the Base Rate Fallacy *is that the base rate (of spam) is so low (5%), that the tiny errors the filter makes lead to inaccurate results. In particular, the chance that a tagged message is a false positive is* $9,500/59,000 = 0.1610\ldots = 16.1\%$ *(!).*

Because the base rate of spam is low, the number of false positives is quite high.

2.4 Honey pots

Honeypot: fake resource serving to fool an intruder.

- Examples:
  - file/database with fake information
  - computer system
  - network (virtual, simulated)
  - e-mail address (spamtrap)
- Purposes:
  - Divert attacker from accessing criticial systems
  - Collect information about the attacker's activity
  - Encourage the attacker to stay on the system long enough for administrators to respond