TSIT01 Computer Security

Föreläsning 8: Lösenordslagring, Unix, Windows, Virtualisering

Jonathan Jogenfors

Information Coding Group
Department of Electrical Engineering

LINKÖPING UNIVERSITY

Agenda

- Password storage
- Unix authentication
- Windows authentication
- Sandboxing via chroot
- Virtualization

Storing passwords

- When a system requires a password to login, the password must be stored in some way
- Danger: Somebody gets to see the password list
- Examples: LinkedIn security break in 2012 (100 million users affected)
- A growing concern, since passwords are used in apps, sites as well as e-mail

Password storage is a rapidly growing problem

- You can't just store password in plain text!
- Solution: Don't store the password p, store its hash $H(p)$
- When a password p' is entered, compute $H(p')$ and compare with $H(p)$.
- For a secure hash function H, $p = p'$ implies $H(p) = H(p')$
- Danger: Dictionary / Rainbow tables

Hashing passwords is not enough

- Hashed can still reveal identical passwords

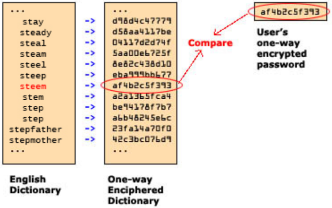
User	Password Hash
Stephen	39e717cd3f5c4be78d97090c69f4e655
Lisa	f567c40623df407ba980bfad6dff5982
James	711f1f88006a48859616c3a5cbcc0377
Harry	fb74376102a049b9a7c5529784763c53
Sarah	39e717cd3f5c4be78d97090c69f4e655

Another weakness of hashed passwords

- For a given hash function, one can pre-compute common passwords
- This is a space-time tradeoff. More space = less time

Using a dictionary for attack

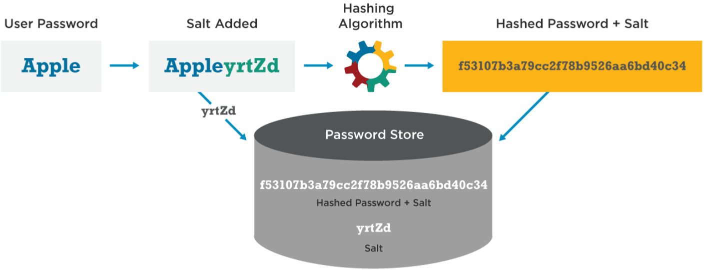
1. Prepare list of common words (i.e. English dictionary)
2. For each word w, compute $H(w)$
3. Store w and $H(w)$
4. Now compare hashes with known hashes



Dictionary attacks are useful

- The size of a dictionary can be reduced with a rainbow table
- Main problem: Hashes are too predictable across users and systems
- Solution: Add a random, public, salt s before hashing
- Password database contains $H(p|s)$ and s
- When checking an entered password p', compute $H(p'|s)$ and compare with database
- Salt should be unique for every user

Hashing and salting



Password hashes are now unpredictable even when using common passwords

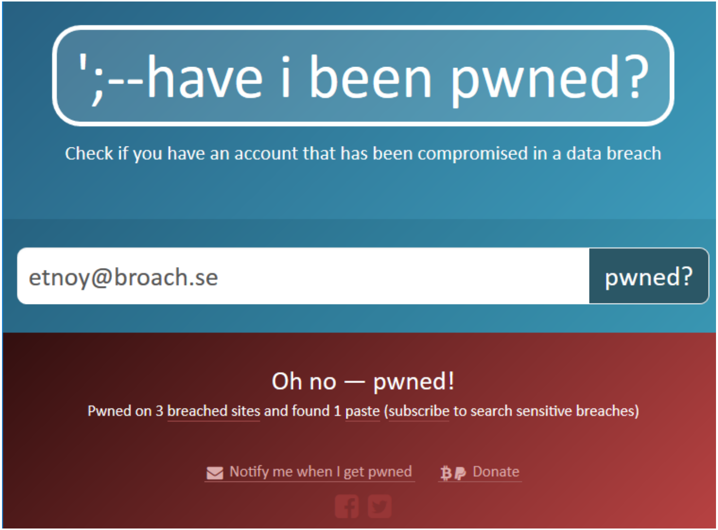
Things are looking better

How Dropbox securely stores your passwords

Devdatta Akhawe | September 21, 2016   0 601 19

It's universally acknowledged that it's a bad idea to store plain-text passwords. If a database containing plain-text passwords is compromised, user accounts are in immediate danger. For this reason, as early as 1976, the industry standardized on storing passwords using secure, one-way hashing mechanisms (starting with Unix Crypt). Unfortunately, while this prevents the direct reading of passwords in case of a compromise, all hashing mechanisms necessarily allow attackers

Dropbox published a very interesting blog post on their approach for storing passwords securely (read it!)


';--have i been pwned?
Check if you have an account that has been compromised in a data breach
etnoy@broach.se
pwned?
Oh no — pwned!
Pwned on 3 breached sites and found 1 paste (subscribe to search sensitive breaches)
Notify me when I get pwned
Donate

Dropbox: In mid-2012, Dropbox suffered a data breach which exposed the stored credentials of tens of millions of their customers. In August 2016, they forced password resets for customers they believed may be at risk. A large volume of data totalling over 68 million records was subsequently traded online and included email addresses and salted hashes of passwords (half of them SHA1, half of them bcrypt).

Compromised data: Email addresses, Passwords

Last.fm: In March 2012, the music website Last.fm was hacked and 43 million user accounts were exposed. Whilst Last.fm knew of an incident back in 2012, the scale of the hack was not known until the data was released publicly in September 2016. The breach included 37 million unique email addresses, usernames and passwords stored as unsalted MD5 hashes.

Compromised data: Email addresses, Passwords, Usernames, Website activity

Plex: In July 2015, the discusison forum for Plex media centre was hacked and over 327k accounts exposed. The IP.Board forum included IP addresses and passwords stored as salted hashes using a weak implementation enabling many to be rapidly cracked.

Compromised data: Email addresses, IP addresses, Passwords, Usernames

Protecting yourself against password leaks

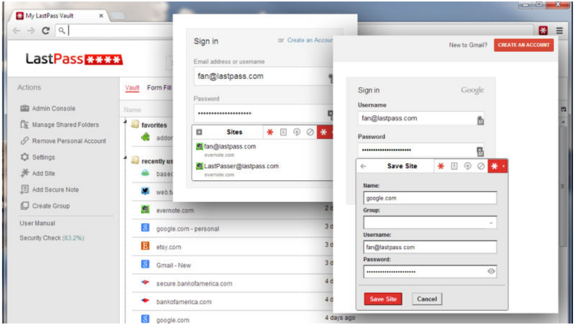


Figure: A password manager such as KeePass and LastPass generates strong, unique passwords for each site

Unix(es)

- Unix started out as a departure from the design philosophy of Multics
- Multics was very complex, Unix was meant to be simple
- Unix is not really used anymore . . .
- . . . but there are many Unix-based systems today
- GNU/Linux, OSX/iOS, Android, FreeBSD, OpenBSD . . .
- Most Unix-like systems inherit the access controls from traditional Unix

Environmental creep in Unix

- Unix was initially used by a small number of skilled and trustworthy people
- Security mechanisms were there to protect from mistakes, not adversaries
- The course books talks of a “success disaster”, where Unix became popular and had to be extended again and again
- Patching new holes is not a viable security method
- . . . but over the years, Unix has improved its reputation

Unix(es)

- Most secure operating systems have a *security architecture*
- Unix has a colorful history of versions
- Generally needs a skilled admin
- Unix systems are extensible and are very widely used, from small embedded devices to extremely large data centers

Configuration in Unix

- System configuration resides in the /etc directory
- Much of the configuration is world-readable, but needs root privileges for writing
- Stored in a file hierarchy, not a database as such
- (although in modern Linux distributions there is a program database that registers what is installed)

Centralized configuration in Unix

- Networked Unix machines are often jointly administered
- User accounts can be administered in a central machine through Network Information Service (NIS or YP), Lightweight Directory Access Protocol (LDAP) or plain Kerberos
- Centralized file systems can be shared via Network File System (NFS)
- . . . however each service often needs individual care

Principals in Unix

- Principals are users and groups, with a *user* or *group identity* (uid or gid)
- uid:s and gid:s are 32-bit numbers
- Some uid:s have special meanings, that may be different between systems (0=root)

/etc/passwd

```
username:password:UID:GID:ID string:home directory:login shell

root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false
jonfo33:x:1000:1000:Jonathan Jogenfors,,,:/home/jonfo33:/bin/bash
```

Principals in Unix

/etc/passwd

```
username:password:UID:GID:ID string:home directory:login shell

root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false
jonfo33:x:1000:1000:Jonathan Jogenfors,,,:/home/jonfo33:/bin/bash
```

- The username can (typically) be eight characters long, and is used for login
- Access control is via uid

Principals in Unix

/etc/passwd

```
username:password:UID:GID:ID string:home directory:login shell

root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false
jonfo33:x:1000:1000:Jonathan Jogenfors,,,:/home/jonfo33:/bin/bash
```

- There is no distinction between uid:s on different systems
- . . . so watch out when you mount removable disks

Authentication in Unix

- Standard user authentication is via password, but other methods are supported
- Modern Unix variants use hashed, salted passwords
- The `/etc/passwd` file is world-readable
- Passwords (hashes) are today stored somewhere more hidden

Authentication in Unix: The `shadow` file

- The shadow file contains the encrypted passwords
- Only readable by root, all other users have no permissions
- Specifies password change intervals and account expiry dates

/etc/shadow

```
username:pw:age:min age:max age:warn-p:inact-p:expiry:reserved
root:$6$saltsalt$yeaRight:15642:0:99999:7:::
daemon:*:15453:0:99999:7:::
bin:*:15453:0:99999:7:::
sys:*:15453:0:99999:7:::
jonfo33:$6$sosalty$youthinkIputmypasswordhere:15642:0:99999:7:::
```

Group principals in Unix

- Groups have their own id:s, gid, which is a number
- Users can belong to more groups than their primary group (user accounts lists them)
- Groups can have passwords, allowing users to directly associate with their permissions

/etc/group

```
group name:group password:GID:list of usernames
root:x:0:
adm:x:4:jonfo33
news:x:9:
lpadmin:x:109:jonfo33,janla64
jonfo33:x:1000:jonfo33
```

Special principals in Unix

- Are used for login, audit logging, I/O access, upgrading (system) programs, and so on
- root can do almost anything
- ...including removing the "almost" in the last sentence
- ...but cannot decrypt users' password
- Others include daemon, bin, sys, games, man, lp, mail, news, uucp, backup, irc, gnats, nobody, syslog, avahi, pulse, speech-dispatcher, haldaemon, ...

Subjects in Unix

- Subjects are processes
- Unix keeps track of active processes, each with a process id, pid
- Each process has a *real* uid/gid, inherited from the parent process
- Each process has an effective uid/gid, inherited from the parent process or the current file being executed (`suid`)
- A process can decide to drop priviliges once started

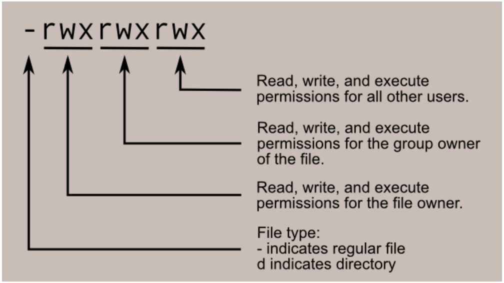
Objects in Unix

- Objects are registered as (i)nodes in a tree
- A node can be a file, a directory or an I/O device, all visible in the "file" tree
- Owner/group is by uid/gid

mode	File type and access rights
uid	Owner
gid	Group
mtime	Modification time
itime	Inode modification time
block count	File size
	Physical location

Some more words on permission bits

```
/home/jonfo33/:
-rw-r-r-        jonfo33  users  tsit02-lecture-05.pdf
-rw----         jonfo33  users  tsit02-exam.pdf
drwxrwx--       jonfo33  icg    ourproject
```



Access control in Unix

- Owners get access according to the “user” bits
- Members of the object’s group get the “group” permission
- All other get the “other” access permissions
- The owner and root can change object permissions
- It is possible to give more access to “others” than to the owning “user”
- The “sticky” bit can handle one special case: job queues and mail directories

Limitations to access control in Unix

- Files have only one owner and one group
- Permissions are read, write, and execute
- All other access rights must be mapped to basic file permissions
- Other operations need to be done through `suid` applications
- Complex security policies are often impractical

Deleting files in Unix

- Deleting a file in Unix means removing the directory item that points to the data of the files
- The actual data is not erased from disk (memory, . . .)
- To actually be removed, the data must be wiped securely before the link is removed
- Secure wipe is to overwrite the data with zeros or noise

Audit logs in Unix

- Unix logs some security-relevant events into the directory /var/log/
- Logging is provided by `syslogd` (or modern replacements such as `syslog-ng`)
- Logs can include valid and invalid login attempts including `sudo`
- Note: `suid` binaries are not logged by default
- The system can be rigged to shut down if it is not possible to log a security-relevant event

Sandboxing in Unix

- There are a few ways we can strengthen access control even further
- The Unix `chroot` system call can create sandboxed "jails"
- Within such a sandbox, a process is restricted to that sandbox
- This can protect against secondary damage from privilege escalation bugs
- Android versions before 4.3 had primitive sandboxes in form of directories with restricted permissions
- Android versions after 4.3 have SELinux-enforced sandboxes

Chroot: limitations and gotchas

- However, a root user can still create device files
- The device file `/dev/mem` gives direct access to the computer memory
- Almost all jail breaking requires root privileges
- From previous lecture: Must protect lower layers from tampering
- Possible solution: No files inside `chroot` with `setuid`/`setgid` permissions, or mount filesystem with the `nosuid` option

Fun with `chroot`: "An evening with Berford in which a Cracker is Lured, Endured and Studied" from 1991.

Windows

- Windows was not designed with security in mind
- Comes from single-user PCs, security was mostly to prevent mistakes
- Security features were added as needed
- This is not a generally viable method
- . . . but over the years, Windows has improved its reputation

Windows

- Windows does have a *security architecture*
- Generally needs a skilled admin (but with a different skillset)
- Lots of development from DOS via Windows 1.0-3.1, 95, 98, 2000, NT, XP and Vista to Windows 7, 8 and 10
- Available mechanisms are less straightforward and much more sophisticated than in Unix
- This opens up for more mistakes as well as it makes far better adjusted controls possible

Configuration: The Windows Registry

- System configuration resides in the Registry in Windows
- Much of the configuration is world-readable, but needs supervisor priviliges for writing
- This is a proper database, and can be accessed via the Registry Editor
- One problem is (was) that many applications assume they can write into the Registry directly

Configuration: The Windows Registry

- Registry integrity is crucial for Windows security
- Registry entries are confusingly called “keys”
- One “key” contains all user profiles
- One “key” defines the local software configuration
- One “key” defines system hardware at startup
- One “key” defines the environment for the current user

Configuration: The Windows Registry

- Registry integrity is crucial for Windows security
- Registry entries are confusingly called "keys"
- A possible problem is the behaviour if one key is missing, e.g., if the key

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet
  \Control\SecurePipeServers\Winreg

  is missing, the registry can be accessed from a remote machine exactly as on the local machine

Centralized configuration in Windows

- Networked Windows machines are often jointly administered in a Domain
- All machines in a Domain share user accounts database and security policy
- Domains can form a hierarchy
- Each Domain has a Domain Controller (DC), acting as a trusted third party in authentication, for example
- Many services are available, configuration is of course needed

Principals in Windows

- Principals are local users, aliases, domain users, groups, or machines, and each have a *security identifier* (SID)
- The SID has the format S-R-I-S-S-. . .-RID, where R is revision, I is identifier authority (48 bit), S is up to 14 subauthority fields, and RID is a relative id, all 32 bit

```
Everyone          S-1-1-0
SYSTEM            S-1-5-18
Administrator     S-1-5-21-<local authority>-500
Administrators    S-1-5-32-544
Domain admins     S-1-5-21-<domain authority>-512
Guest             S-1-5-21-<authority>-501
```

Principals in Windows

- Principals are local users, aliases, domain users, groups, or machines, and each have a *security identifier* (SID)

```
Everyone          S-1-1-0
SYSTEM            S-1-5-18
Administrator     S-1-5-21-<local authority>-500
Administrators    S-1-5-32-544
Domain admins     S-1-5-21-<domain authority>-512
Guest             S-1-5-21-<authority>-501
```

- local users and aliases are administered locally
- domain users and groups are administered by the DC
- groups can be nested, aliases cannot
- SIDs are (statistically) unique, using pseudorandom numbers

Special principals in Windows

- Is used for login, audit logging, I/O access, upgrading (system) programs, and so on
- Administrator can do almost anything
- . . . including removing the "almost" in the last sentence
- . . . but cannot decrypt users' password
- Others include LocalSystem, Administrators, Domain admins, Domain Users, Everyone, Interactive, Network, CreatorOwner, . . .

Priviliges in Windows

Here, "priviliges" are not generic root priviliges, nor access to objects belonging to some group; instead typical priviliges can be:

- Backing up files and directories
- Generating security audits
- Managing and auditing security logs
- Taking ownership of files and other objects
- Enabling computer and user accounts to be trusted for delegation
- Shutting down the system

Subjects in Windows

- Subjects are processes and threads
- Windows stores security credentials in an access token, that contains
  - User SID
  - Group SIDs
  - Privileges (to system resources, the union of the above priviliges)
  - Defaults for new objects (owner SID, group, and DACL, see below)
  - Miscellaneous (session ID and token ID)
- The token does not change once created, which is more efficient, but TOCTTOU may be a problem

Objects in Windows

- Objects are the passive parts in an access operation
- Windows objects can be active, ”executive objects”, like processes and threads
- Registry entries, devices, . . . , are also objects
- Standard file system objects too
- Each object has a security descriptor listing access control data for it

Objects in Windows

Owner SID
Primary Group
DACL
SACL

- (The Primary Group is for POSIX compliance)
- The Discretionary ACL (DACL) determines access properties
- The System ACL (SACL) defines the audit policy

Permissions in Windows

- An D/SACL is a list of Access Control Entries (ACEs), that in Windows contains

Type (allow,deny,monitor)
Inheritance and audit flags
Access mask
ObjectType
InheritedObjectType
SID (that the ACE applies to)

Permissions in Windows

- Standard permissions are modify/read-and-execute/read/write/full-control
- There are also Advanced permissions, for example subdividing "read" into read-data/read-attributes/read-extended-attributes/read-permissions

Folders have a different behaviour from Unix

- In Unix, each directory in the path is checked for access rights
- In Windows, the path is (only) an identifier, so only the DACL of the target object is checked
- A file may be accessable even when a folder (tree) is not

Permissions in Windows

- In Windows, the path is a (only) an identifier, so only the DACL of the target object is checked
- However, access rights can be inherited in Windows
- This is actually the default, but can be chosen per-folder

Order of precedence

- Deny before Allow
- Explicit before inherited
- Parent before grandparent (before great-grandparent)

This makes fine-tuned control possible (via ACLs), while keeping the admin complexity down (via inheritance)

Deleting files in Windows

- Deleting a file in Windows means removing the directory item that points to the data of the files
- The actual data is not erased from disk
- To actually be removed, the data must be wiped securely before the link is removed
- Secure wipe is to overwrite the data with zeros or noise

Audit logs in Windows

- Windows logs security-relevant events in the security log
- The entries are generated by the Security Reference Monitor
- The object SACLs cause such events to be logged
- Other sources typically include valid and invalid login attempts, and privilige use

- A maximum log size can be set
- The system can be rigged to shut down if it is not possible to log a security-relevant event

Formal models and Windows

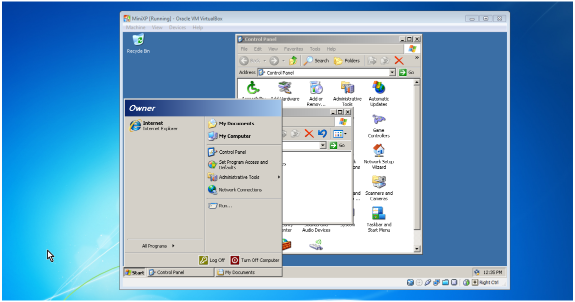
- Windows provides more fine-tuned access control
- Inheritance also plays a (bigger) role
- But the same critique holds; there is no check that the levels really form a hierarchy
- Although Windows UAC attempts to implement Biba, and almost succeeds

Unix vs Windows

- There are differences that you need to be aware of
- Windows is (was) more complicated, but also gives (gave) more control
- Be careful out there

Virtualization

- In the past decade, virtualization has become very popular
- A *virtual machine* (VM) is a simulated computer, which in turn runs an operating system

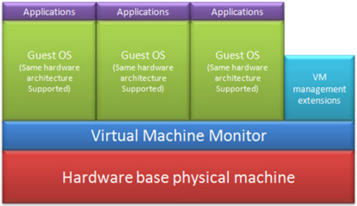


The virtual revolution

- Virtualization took off about a decade ago
- Report from Kapersky in 2012: 69 % of US companies are using server virtualization

Full virtualization

- There are many forms of virtualization, with different architectures
- This lecture will cover *full virtualization*, where the guest OS has no idea it is ran on a VM
- Hypervisor or Virtual Machine Monitor takes care of managing the guests



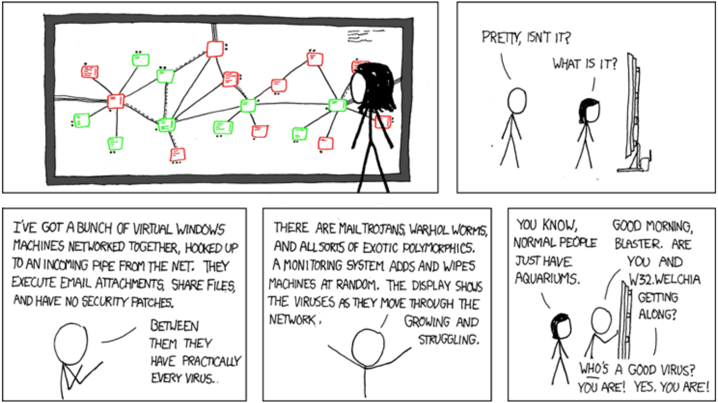
Virtualization and security

- Virtualization allows a higher degree of isolation compared to `chroot`
- One compromised guest system typically does not compromise the host
- However, virtual machines can be complex and hard to analyze
- Virtualization can also increase flexibility, making systems easier to administer

Virtualization reduces administrative burden

- A new OS can be installed, configured, secured and tested in a VM
- The administrator then takes a snapshot of this VM
- This snapshot can then be distributed to many hosts
- If a guest system is compromised, it can be frozen (including RAM contents), which makes forensic analysis much easier
- A virtual machine can be migrated between hosts (often while running!)
- This makes it less of a hassle to install patches and reboot the host server

The Virtual Zoo



Attacks on virtual machines

- Virtual machines are complex, with new possible attack vectors due to the complexity involved
- Simplest attack: VM detection
  - Timing attacks
  - Checking the local descriptor table
- Compromised guest system performs denial of service on host or other guests
  - Overloading
- Guest-to-guest communication
  - Shared memory and resources

Attacks on virtual machines

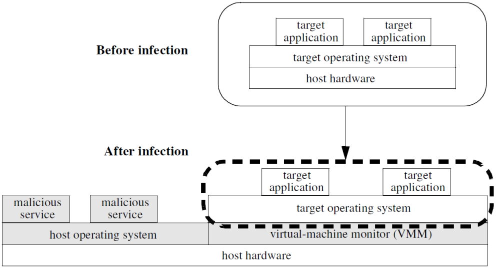
- Virtual machines are complex, with new possible attack vectors due to the complexity involved
- Simplest attack: VM detection
  - Timing attacks
  - Checking the local descriptor table
- Compromised guest system performs denial of service on host or other guests
  - Overloading
- Guest-to-guest communication
  - Shared memory and resources
- Escaping the virtual machine

Virtual machine esacpe

- The most serious type of attack
- Usually grants privileged access to the host machine
- . . . which of course compromises all guests on that host
- Often caused by bugs in the hypervisor
- Example: Cloudburst attack on VMWare in 2009

Virtual machines as malware

- Subvirt was published in 2006 by King et al.
- Compromises a system by installing a virtual machine underneath it



Virtualization paved the way for cloud computing

- Virtual Private Server (VPS): A virtual machine sold as a service
- VPS allows customers root access within their VPS
- Example: Amazon AWS provides the Elastic Compute Cloud (EC2), where VPS instances can be managed and created
- EC2 can be rented in the following ways:
  - On-demand (hourly rates)
  - Reserved (long-term rental)
  - Spot (bid-based, runs jobs only if the spot price is below the bid price)
- Future trend: Centralization of computing

Jonathan Jogenfors

www.liu.se

LINKÖPING UNIVERSITY