

Access Control and Privacy Policies (4)

Email: christian.urban at kcl.ac.uk

Office: SI.27 (1st floor Strand Building)

Slides: KEATS (also home work is there)



two weeks ago: buffer overflow attacks

Buffer Overflows

As a proof-of-concept, the following URL allows attackers to control the return value saved on the stack (the vulnerability is triggered when executing `"/usr/sbin/widget"`):

```
curl http://<target ip>/post_login.xml?hash=AAA...AAABBBB
```

The value of the `"hash"` HTTP GET parameter consists in 292 occurrences of the `'A'` character, followed by four occurrences of character `'B'`. In our lab setup, characters `'B'` overwrite the saved program counter (`%ra`).

Discovery date: 06/03/2013

Release date: 02/08/2013

<http://pastebin.com/vbiG42VD>

Backdoors

D-Link router flaw lets anyone login through
"Joel's Backdoor":

If you tell your browser to identify itself as Joel's
backdoor, instead of (say) as Mozilla/5.0
AppleWebKit/536.30.1 Version/6.0.5, you're in
without authentication.

"What is this string," I hear you ask?
You will laugh: it is

```
xmlset_roodkcableoj28840ybtide
```

October 15, 2013

<http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/>

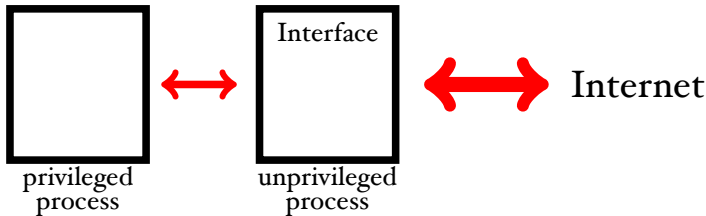
Access Control in Unix

- access control provided by the OS
- authenticate principals (login)
- mediate access to files, ports, processes according to **roles** (user ids)
- roles get attached with privileges

principle of least privilege:
programs should only have as much privilege as they need

Access Control in Unix (2)

- the idea is to restrict access to files and therefore lower the consequences of an attack



Access Control

- **Discretionary Access Control:**

Access to objects (files, directories, devices, etc.) is permitted based on user identity. Each object is owned by a user. Owners can specify freely (at their discretion) how they want to share their objects with other users, by specifying which other users can have which form of access to their objects.

Discretionary access control is implemented on any multi-user OS (Unix, Windows NT, etc.).

Access Control

- **Mandatory Access Control:**

Access to objects is controlled by a system-wide policy, for example to prevent certain flows of information. In some forms, the system maintains security labels for both objects and subjects (processes, users), based on which access is granted or denied. Labels can change as the result of an access. Security policies are enforced without the cooperation of users or application programs.

This is implemented today in special military operating system versions (SELinux).

Discretionary Access Control

In its most generic form usually given by an Access Control Matrix of the form

	/mail/jane	edit.exe	sendmail
jane	r, w	r, x	r, x
john	\emptyset	r, w, x	r, x
sendmail	a	\emptyset	r, x

access privileges: **r**ead, **w**rite, **e**xecute, **a**ppend

Mandatory Access Control

- Restrictions to allowed information flows are not decided at the user's discretion (as with Unix `chmod`), but instead enforced by system policies.
- Mandatory access control mechanisms are aimed in particular at preventing policy violations by untrusted application software, which typically have at least the same access privileges as the invoking user.

Simple example: Air Gap Security. Uses completely separate network and computer hardware for different application classes.

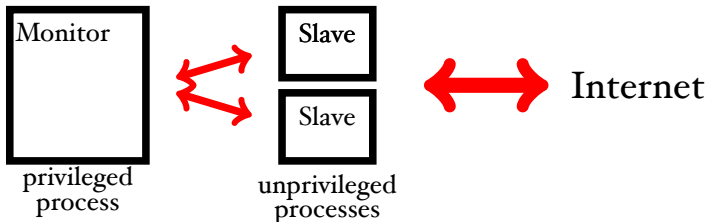
Setuid and Setgid

The solution is that unix file permissions are 9 +
2 Bits: **Setuid** and **Setgid** Bits

- When a file with setuid is executed, the resulting process will assume the UID given to the owner of the file.
- This enables users to create processes as root (or another user).
- Essential for changing passwords, for example.

```
chmod 4755 fobar_file
```

Privilege Separation in OpenSSH



- pre-authorisation slave
- post-authorisation
- 25% codebase is privileged, 75% is unprivileged

Network Applications

ideally network application in Unix should be designed as follows:

- need two distinct processes
 - one that listens to the network; has no privilege
 - one that is privileged and listens to the latter only (but does not trust it)
- to implement this you need a parent process, which forks a child process
- this child process drops privileges and listens to hostile data
- after authentication the parent forks again and the new child becomes the user

Other Problems

There are things you just cannot solve on the programming side:

- for system maintenance you often have cron-jobs cleaning /tmp
 - attacker:
mkdir /tmp/a; cat > /tmp/a/passwd
 - root:
rm /tmp/*/*:
 - attacker:
rm /tmp/a/passwd; rmdir /tmp/a;
ln -s /etc /tmp/a

Security Levels

Unix essentially can only distinguish between two security levels (root and non-root).

- In military applications you often have many security levels (top-secret, secret, confidential, unclassified)

Security Levels

Unix essentially can only distinguish between two security levels (root and non-root).

- In military applications you often have many security levels (top-secret, secret, confidential, unclassified)
- Information flow: Bell — La Padula model
 - read: your own level and below
 - write: your own level and above

Security Levels (2)

- Bell — La Padula preserves data secrecy, but not data integrity

Security Levels (2)

- Bell — La Padula preserves data secrecy, but not data integrity
- Biba model is for data integrity
 - read: your own level and above
 - write: your own level and below

Access Control in 2000

According to Ross Anderson (1st edition of his book), some senior Microsoft people held the following view:

Access control does not matter. Computers are becoming single-purpose or single-user devices. Single-purpose devices, such as Web servers that deliver a single service, don't need much in the way of access control as there's nothing for operating system access controls to do; the job of separating users from each other is best left to application code. As for the PC on your desk, if all the software on it comes from a single source, then again there's no need for the operating system to provide separation. (in 2000)

Research Problems

- with access control we are back to 1970s

Going all the way back to early time-sharing systems we systems people regarded the users, and any code they wrote, as the mortal enemies of us and each other. We were like the police force in a violent slum.

— Roger Needham

Research Problems

- with access control we are back to 1970s
- the largest research area in access control in 2000-07 has been “Trusted Computing”, but thankfully it is dead now
- a useful research area is to not just have robust access control, but also usable access control — by programmers and users (one possible answer is operating system virtualisation, e.g. Xen, VMWare)

Research Problems

- with access control we are back to 1970s
- the largest research area in access control in 2000-07 has been “Trusted Computing”, but thankfully it is dead now
- a useful research area is to not just have robust access control, but also usable access control — by programmers and users (one possible answer is operating system virtualisation, e.g. Xen, VMWare)
- electronic voting

Mobile OS

- iOS and Android solve the defence-in-depth problem by **sandboxing** applications
- you as developer have to specify the resources an application needs
- the OS provides a sandbox where access is restricted to only these resources

Security Theatre

Security theatre is the practice of investing in countermeasures intended to provide the feeling of improved security while doing little or nothing to actually achieve it.

Bruce Schneier

Security Theatre

- for example, usual locks and strap seals are security theatre



From: Ross Anderson
<Ross.Anderson@cl.cam.ac.uk>
To: cl-security-research@lists.cam.ac.uk
Subject: Tip off
Date: Tue, 02 Oct 2012 13:12:50 +0100

I received the following tip off, and have removed the sender's coordinates. I suspect it is one of many security vendors who don't even get the basics right; if you ever go to the RSA conference, there are a thousand such firms in the hall, each with several eager but ignorant salesmen. A trying experience.

Ross

I'd like to anonymously tip you off about this product:

<http://www.strongauth.com/products/key-appliance.html>

It sounds really clever, doesn't it?

...

Anyway, it occurred to me that you and your colleagues might have a field day discovering weaknesses in the appliance and their implementation of security. However, whilst I'd be willing to help and/or comment privately, it'd have to be off the record ;-)

Schneier: Step 1

What assets are you trying to protect?

This question might seem basic, but a surprising number of people never ask it. The question involves understanding the scope of the problem. For example, securing an airplane, an airport, commercial aviation, the transportation system, and a nation against terrorism are all different security problems, and require different solutions.

Schneier: Step 1

What assets are you trying to protect?

This question might seem basic, but a surprising number of people never ask it. The question involves understanding the scope of the problem. For example, securing an airplane, an airport, commercial aviation, the transportation system, and a nation against terrorism are all different security problems, and require different solutions.

You like to prevent: “It would be terrible if this sort of attack ever happens; we need to do everything in our power to prevent it.”

Schneier: Step 2

What are the risks to these assets?

Here we consider the need for security. Answering it involves understanding what is being defended, what the consequences are if it is successfully attacked, who wants to attack it, how they might attack it, and why.

Schneier: Step 3

How well does the security solution mitigate those risks?

Another seemingly obvious question, but one that is frequently ignored. If the security solution doesn't solve the problem, it's no good. This is not as simple as looking at the security solution and seeing how well it works. It involves looking at how the security solution interacts with everything around it, evaluating both its operation and its failures.

Schneier: Step 4

What other risks does the security solution cause?

This question addresses what might be called the problem of unintended consequences. Security solutions have ripple effects, and most cause new security problems. The trick is to understand the new problems and make sure they are smaller than the old ones.

Schneier: Step 5

What costs and trade-offs does the security solution impose?

Every security system has costs and requires trade-offs. Most security costs money, sometimes substantial amounts; but other trade-offs may be more important, ranging from matters of convenience and comfort to issues involving basic freedoms like privacy. Understanding these trade-offs is essential.