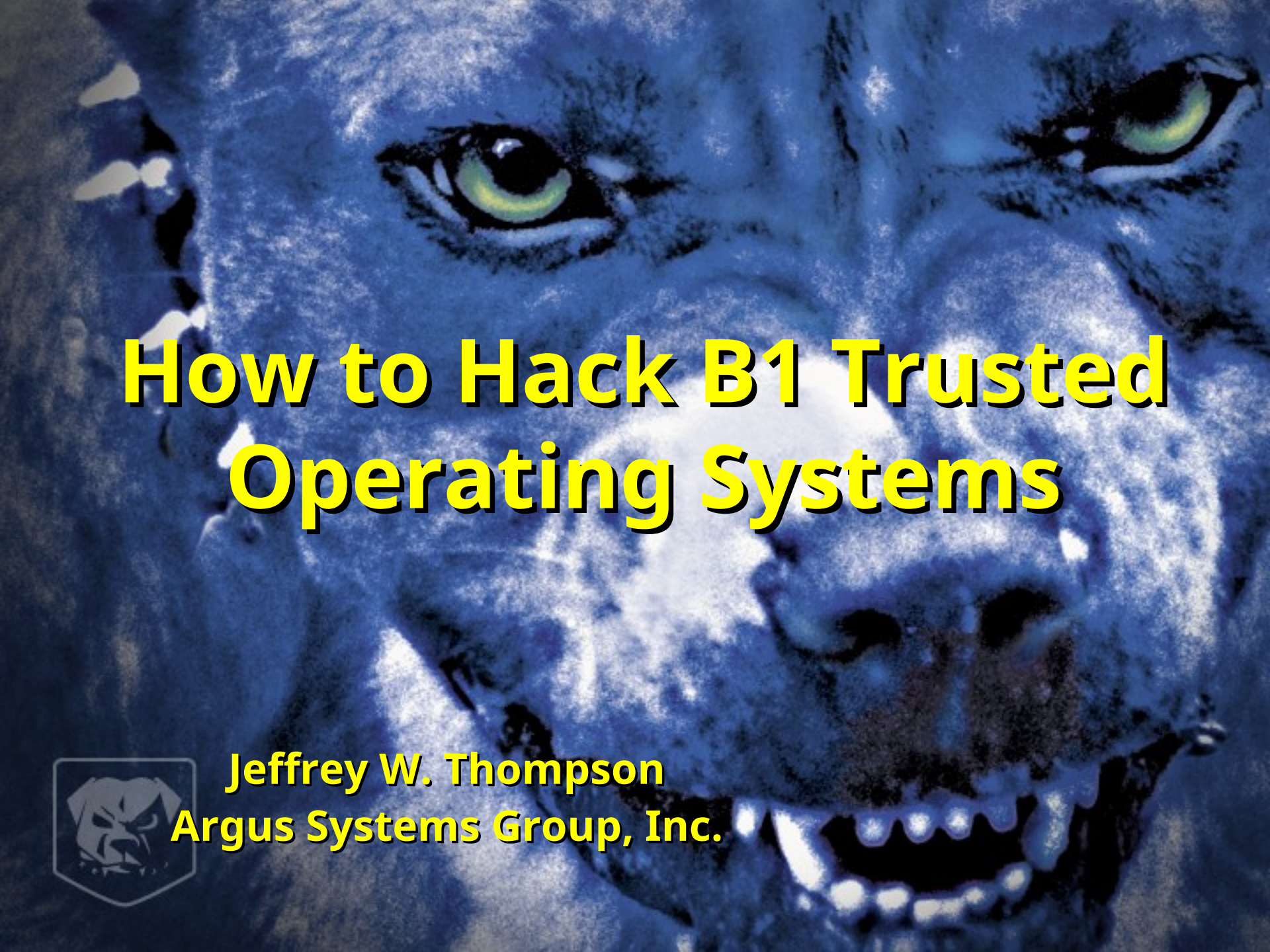
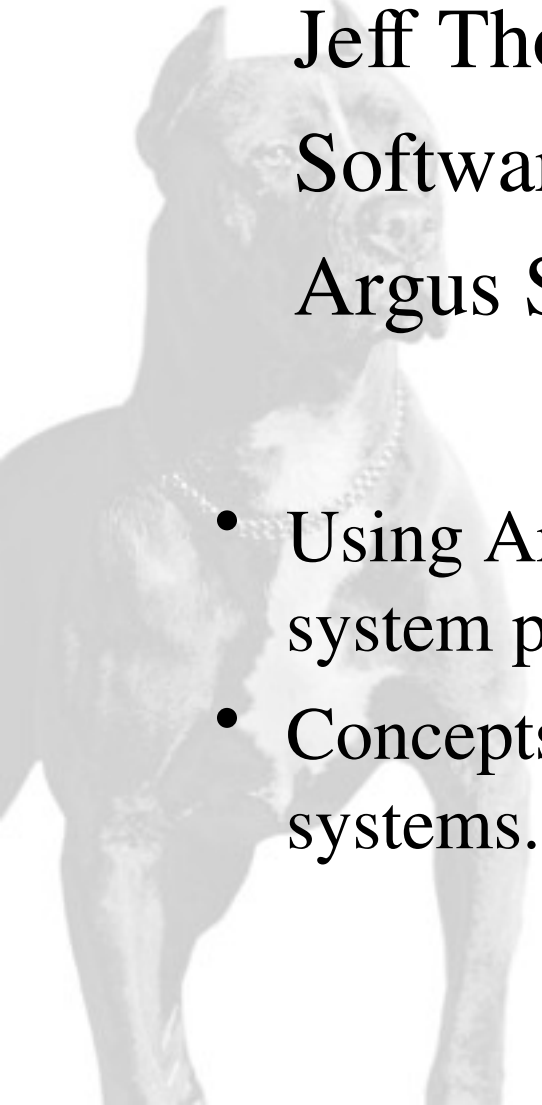


How to Hack B1 Trusted Operating Systems

Jeffrey W. Thompson
Argus Systems Group, Inc.

Introduction



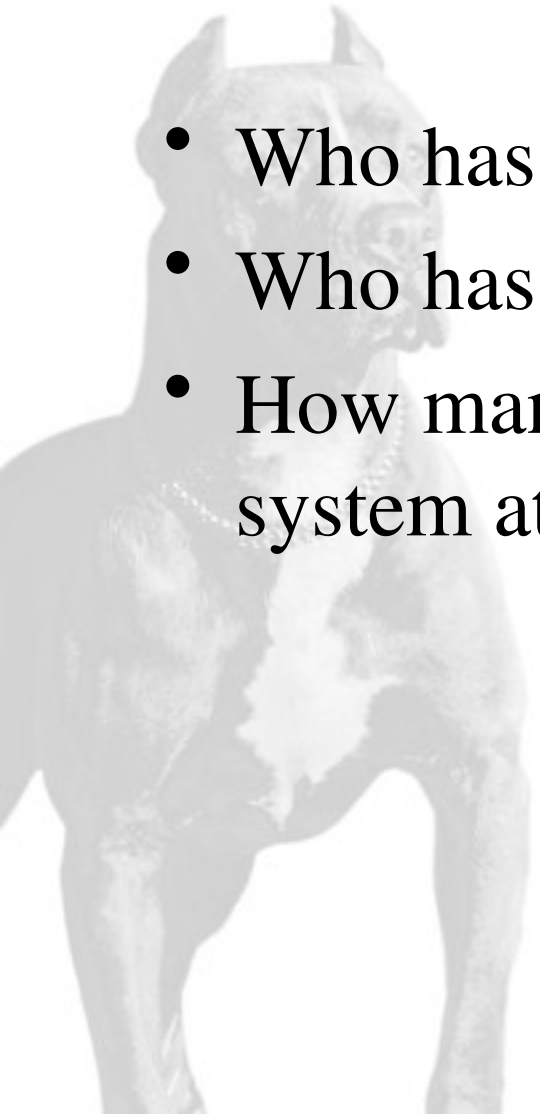
Jeff Thompson (Mythrandir)
Software Evangelist and Visionary
Argus Systems Group, Inc.

- Using Argus' PitBull .comPack trusted operating system product suite as a reference
- Concepts are easily translatable to other TOS systems.

Company Profile

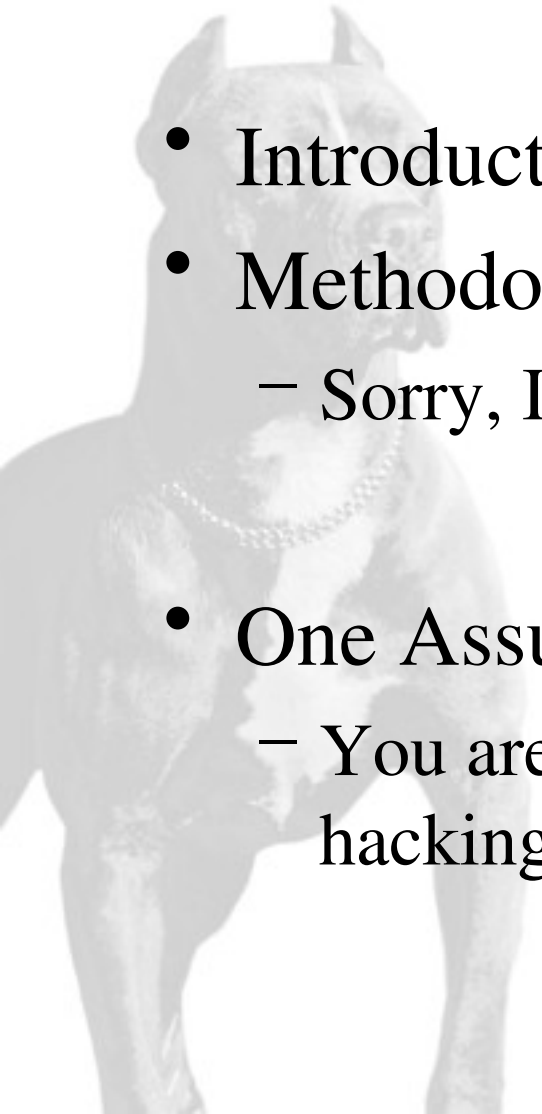
- Argus Systems Group, Inc. is an international provider of Internet security software and engineering services providing E-commerce systems security solutions
- Dedicated to providing security solutions necessary to advance new ways of conducting business over the Internet

Quick Questions



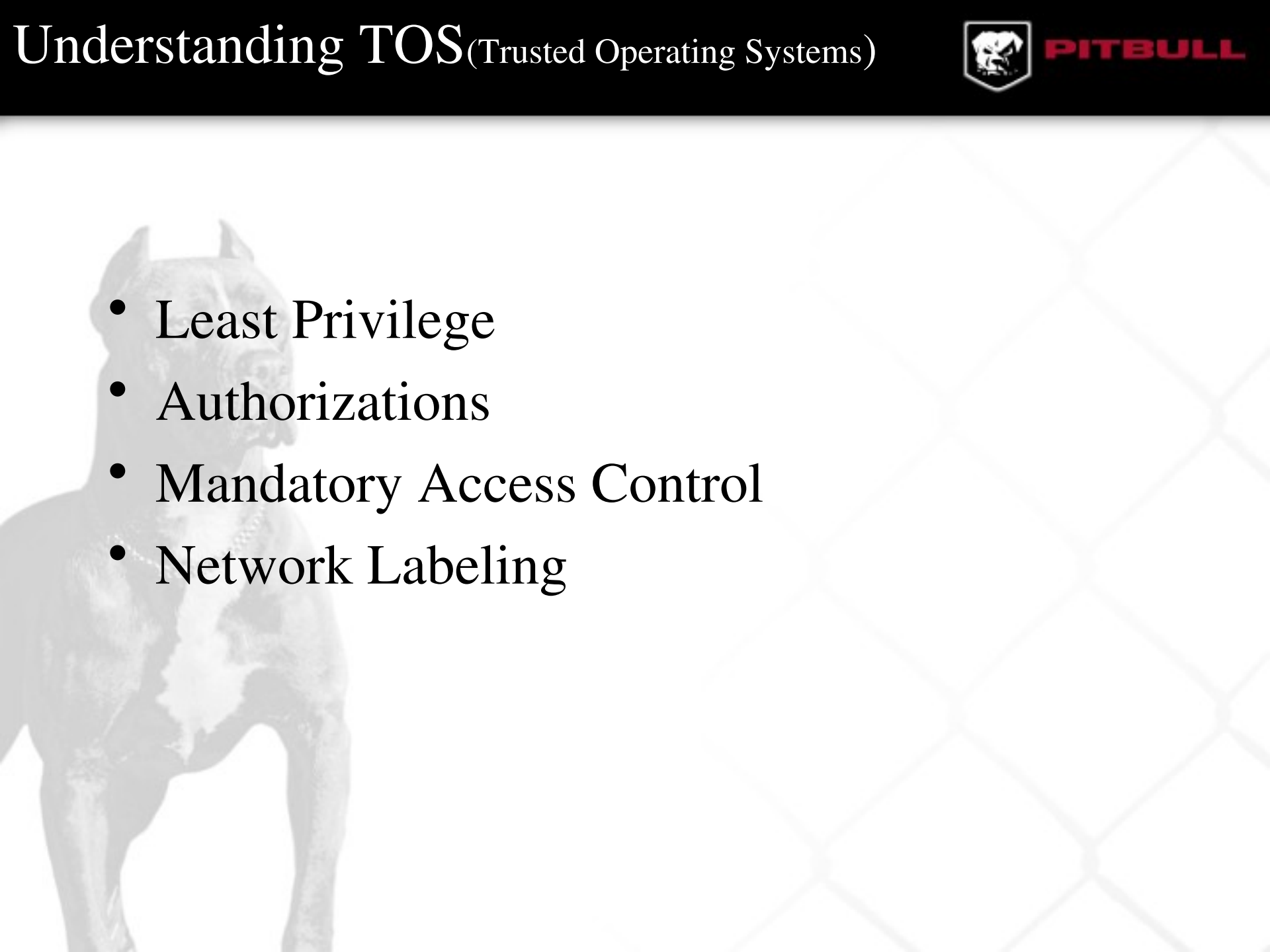
- Who has ever heard of TOS?
- Who has used it?
- How many have tried to hack the PitBull B1 system at 10.20.1.41 for CtF?

Overview



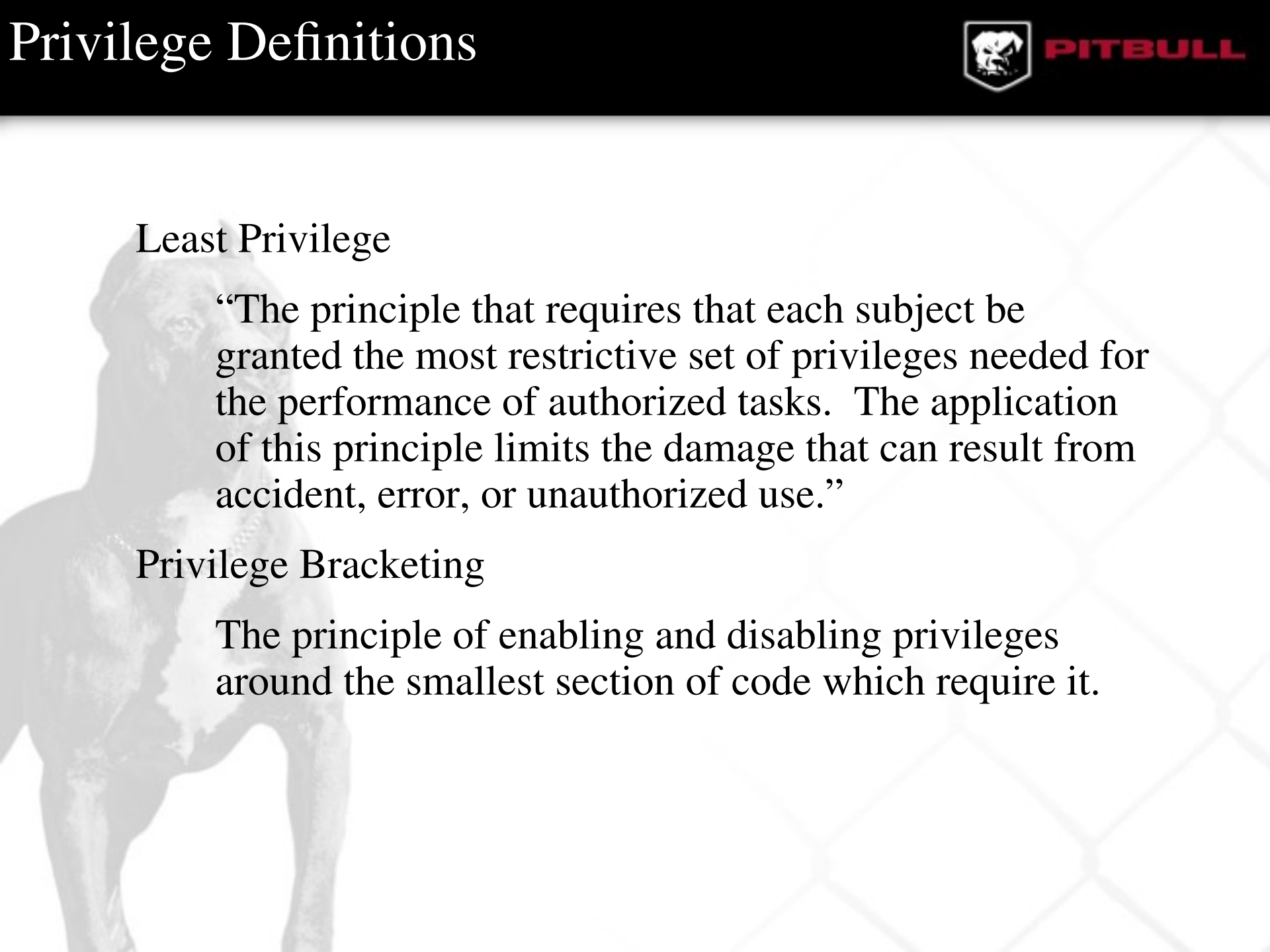
- Introduction to Trusted Operating Systems
- Methodologies for Hacking TOS
  - Sorry, I won't be giving out any –1 day warez
- One Assumption
  - You are already intimately familiar with hacking regular old vanilla OSs

Understanding TOS(Trusted Operating Systems)



- Least Privilege
- Authorizations
- Mandatory Access Control
- Network Labeling

Privilege Definitions



Least Privilege

> "The principle that requires that each subject be granted the most restrictive set of privileges needed for the performance of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use."

Privilege Bracketing

> The principle of enabling and disabling privileges around the smallest section of code which require it.

Privileges

- Traditional UNIX has one privilege: root.
- Argus has divided the root privilege into many sub-privileges, for example:

  PV_FS_MOUNT
  PV_DAC_R

- Argus includes new privileges, for example:

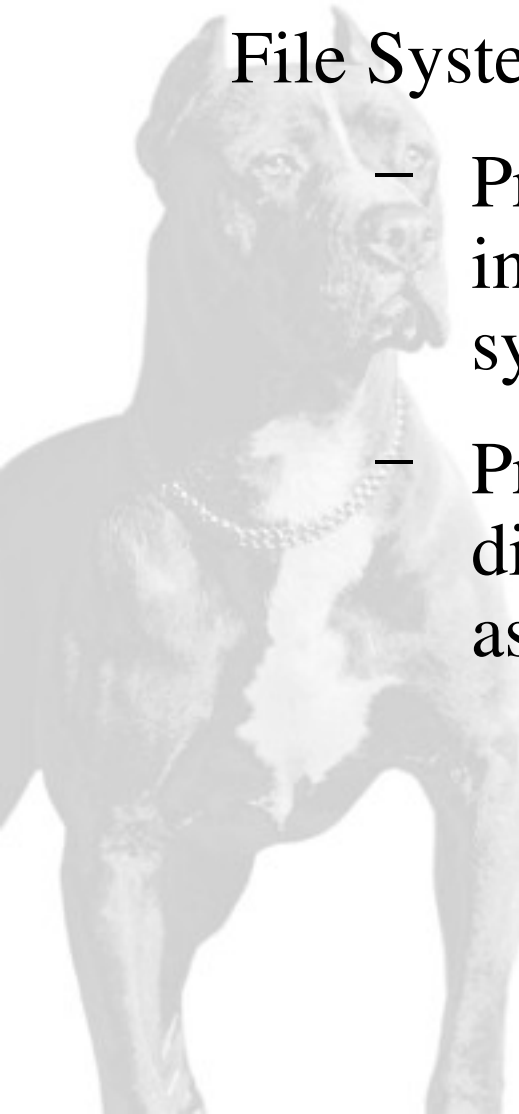
  PV_MAC_W
  PV_PV_PROC

Process Privileges

Three privilege sets are associated with each process:

Limiting	the maximum possible privilege set a process can have during its lifetime
Maximum	the set of privileges over which a process has control
Effective	the set of privileges used to override system restrictions

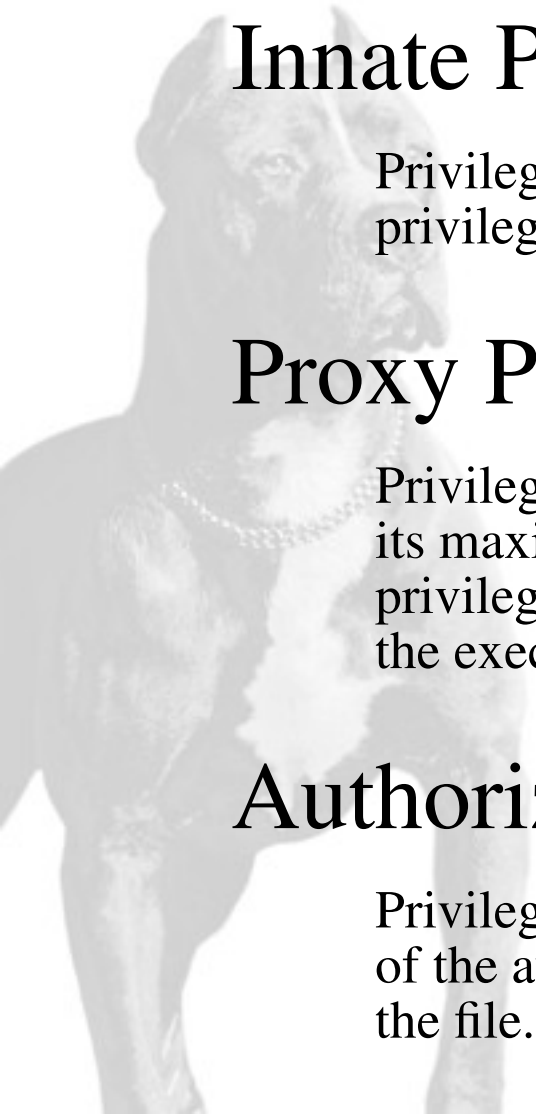
File System Privileges



File System Privilege Information:

- Privilege information is stored in the inode (index node) of each file on the system..
- Privilege information is ignored for directories Three privilege sets are associated with each file:
  - innate
  - proxy
  - authorized

File System Privileges (cont'd)



Innate Privileges

Privileges a process is guaranteed to have in its maximum privilege vector upon startup.

Proxy Privileges

Privileges that will be granted only if the process has them in its maximum privilege set prior to executing the file (*i.e.* privileges that the process will be allowed to "keep" across the exec).

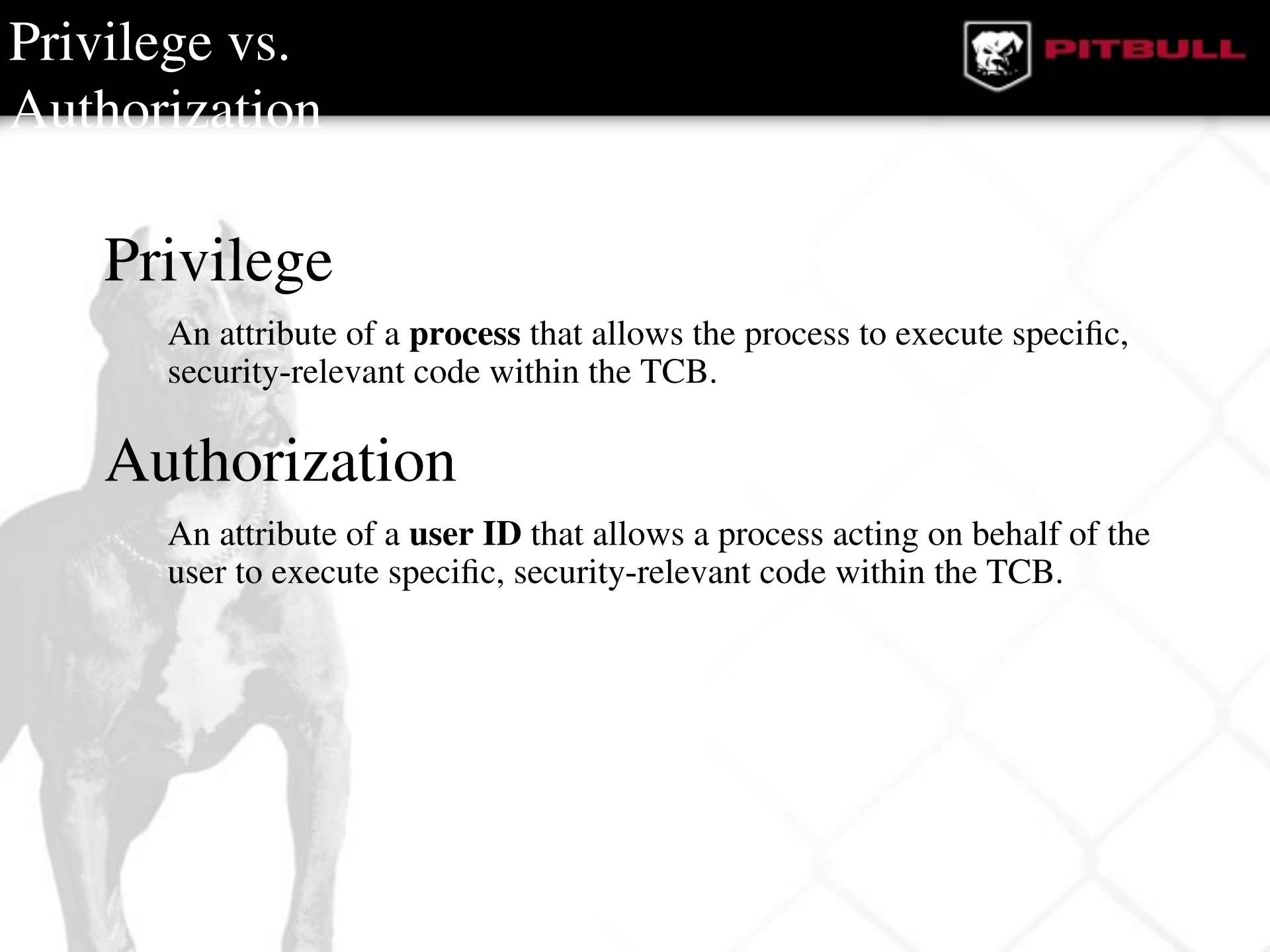
Authorized Privileges

Privileges that will be granted only if the user has at least one of the authorizations in the Privileged Authorization Set of the file.

Privilege Inheritance

- New processes are created via the `fork()` system call.
- `fork()` copies all privileges from the parent to the new child.
- Executable files (programs) are executed via the `exec()` system call.
- `exec()` calculates the new process's privileges according to several rules based on the privileges in the old process and privileges on the executable file.

Privilege vs. Authorization



Privilege

An attribute of a **process** that allows the process to execute specific, security-relevant code within the TCB.

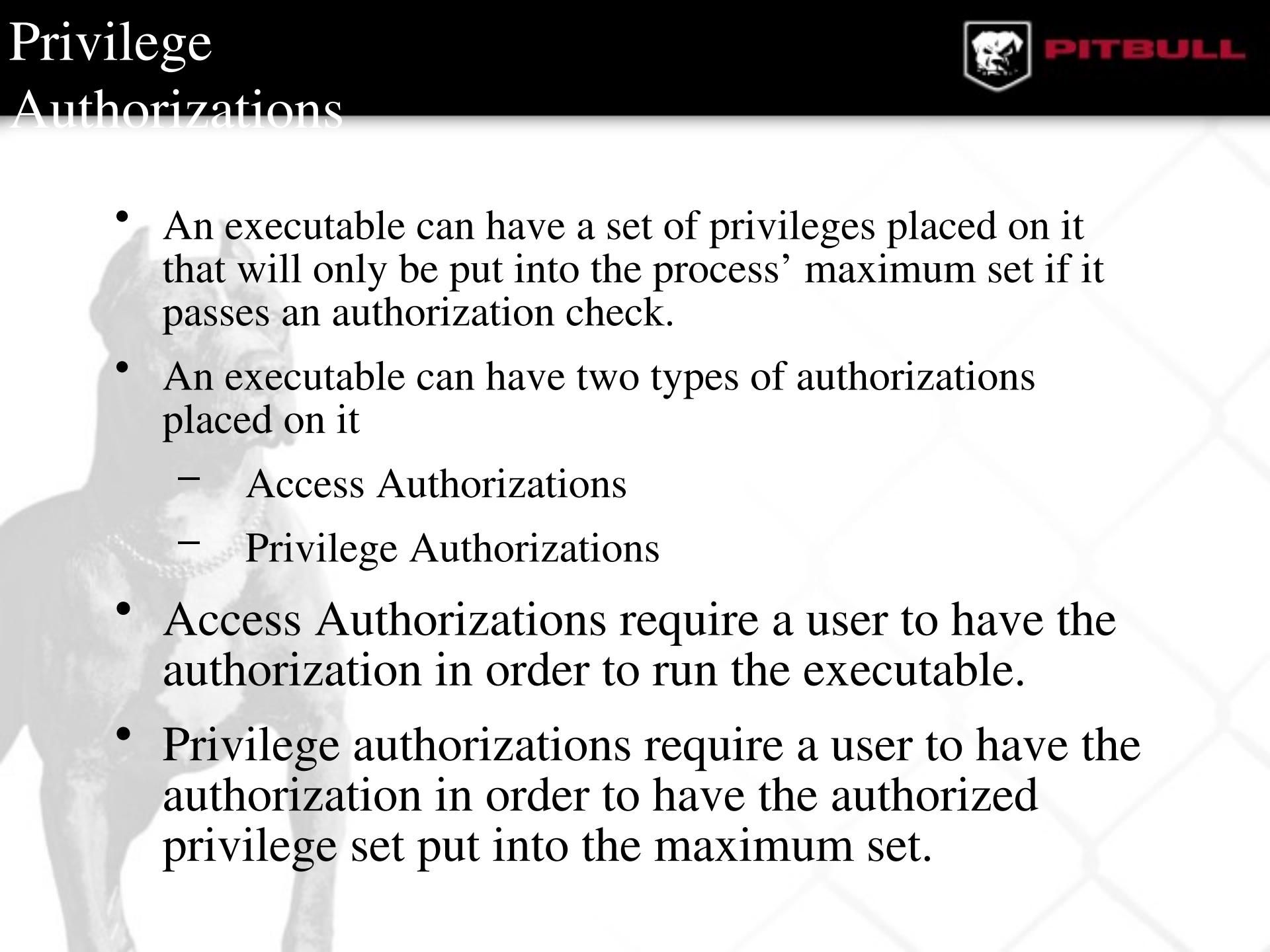
Authorization

An attribute of a **user ID** that allows a process acting on behalf of the user to execute specific, security-relevant code within the TCB.

Unauthorized Use of Programs

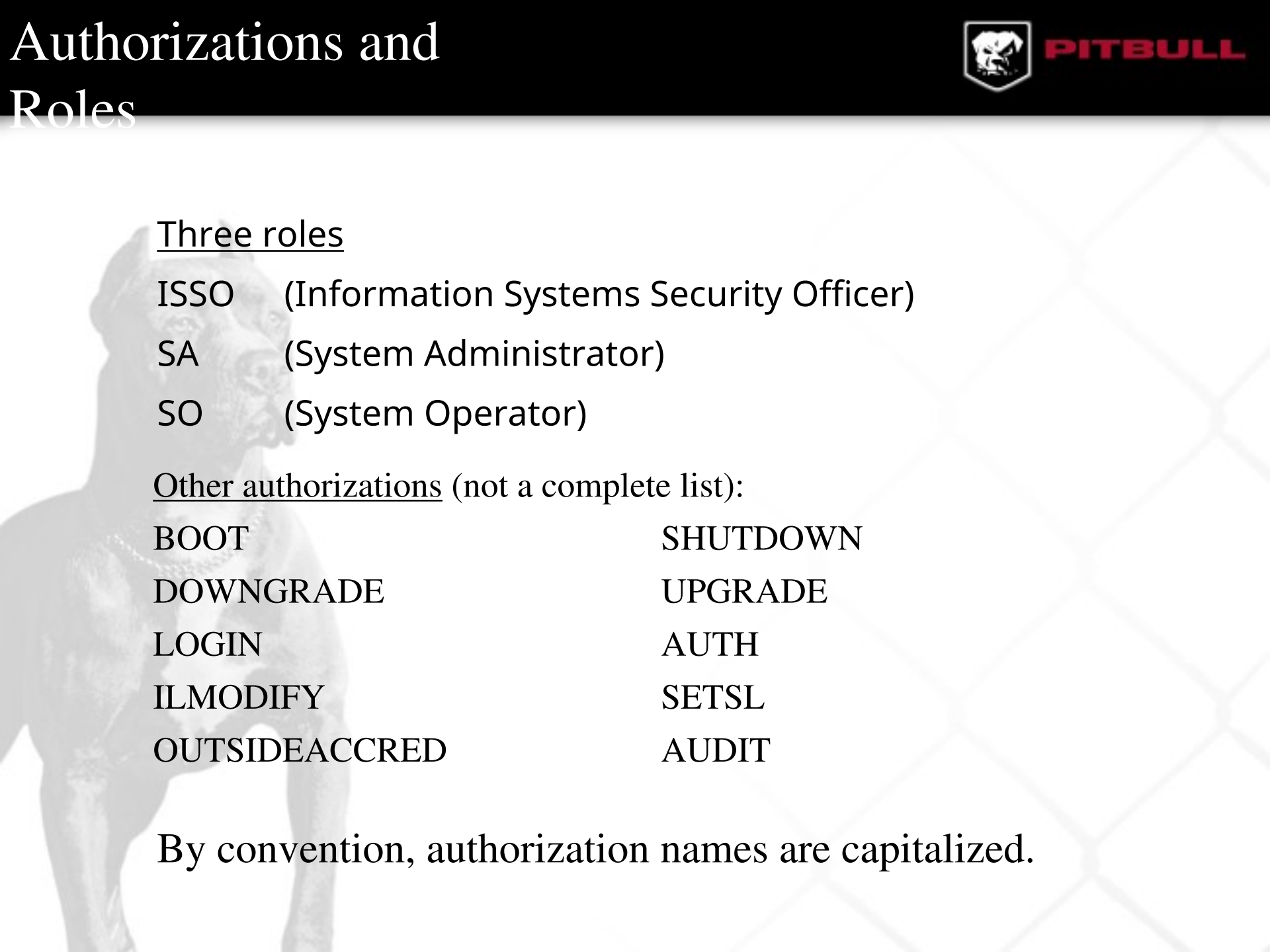
- When a user runs a program (creates a new process and runs an executable file) that is privileged, that program can check to see if the user is authorized to use the program or the privileges.
- If the person running the program doesn't have the appropriate authorization, the program can:
  1) exit with an error message
  2) run with a reduced feature set (*e.g.*, **`ls`** shows all files if run by an ISSO, otherwise it shows only files dominated by the **`ls`**ing process)
  3) disallow some functionality (*e.g.*, the **`-c`** option on **`cpio`** can only be used by the ISSO)

Privilege Authorizations



- An executable can have a set of privileges placed on it that will only be put into the process' maximum set if it passes an authorization check.
- An executable can have two types of authorizations placed on it
  - Access Authorizations
  - Privilege Authorizations
- Access Authorizations require a user to have the authorization in order to run the executable.
- Privilege authorizations require a user to have the authorization in order to have the authorized privilege set put into the maximum set.

Authorizations and Roles



Three roles

ISSO (Information Systems Security Officer)

SA (System Administrator)

SO (System Operator)

Other authorizations (not a complete list):

BOOT	SHUTDOWN
DOWNGRADE	UPGRADE
LOGIN	AUTH
ILMODIFY	SETSL
OUTSIDEACCRED	AUDIT

By convention, authorization names are capitalized.

MAC Definition

Mandatory Access Control:

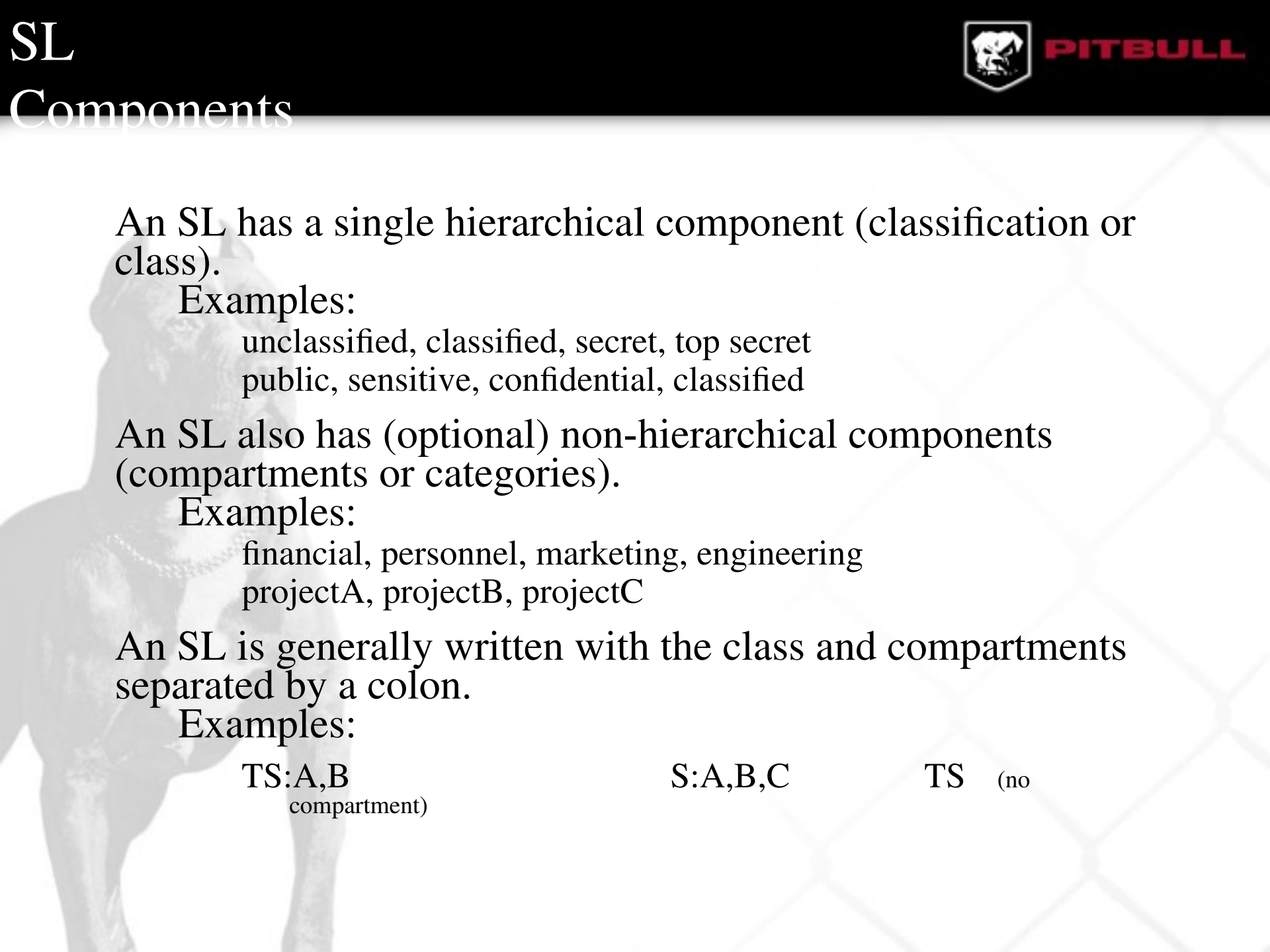
> *"A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity."*

TCSEC, Glossary

MAC Explanation

1. The owner of a file cannot change the MAC settings of a file unless he is authorized to do so.
2. The owner of a file cannot give another user access to the file unless the user is already authorized to have access to that class of data.
3. Copies of the file will automatically be protected at least as well as the original file, no matter how the copy is created and no matter what program or utility is used.

SL Components



An SL has a single hierarchical component (classification or class).

Examples:

unclassified, classified, secret, top secret

public, sensitive, confidential, classified

An SL also has (optional) non-hierarchical components (compartments or categories).

Examples:

financial, personnel, marketing, engineering

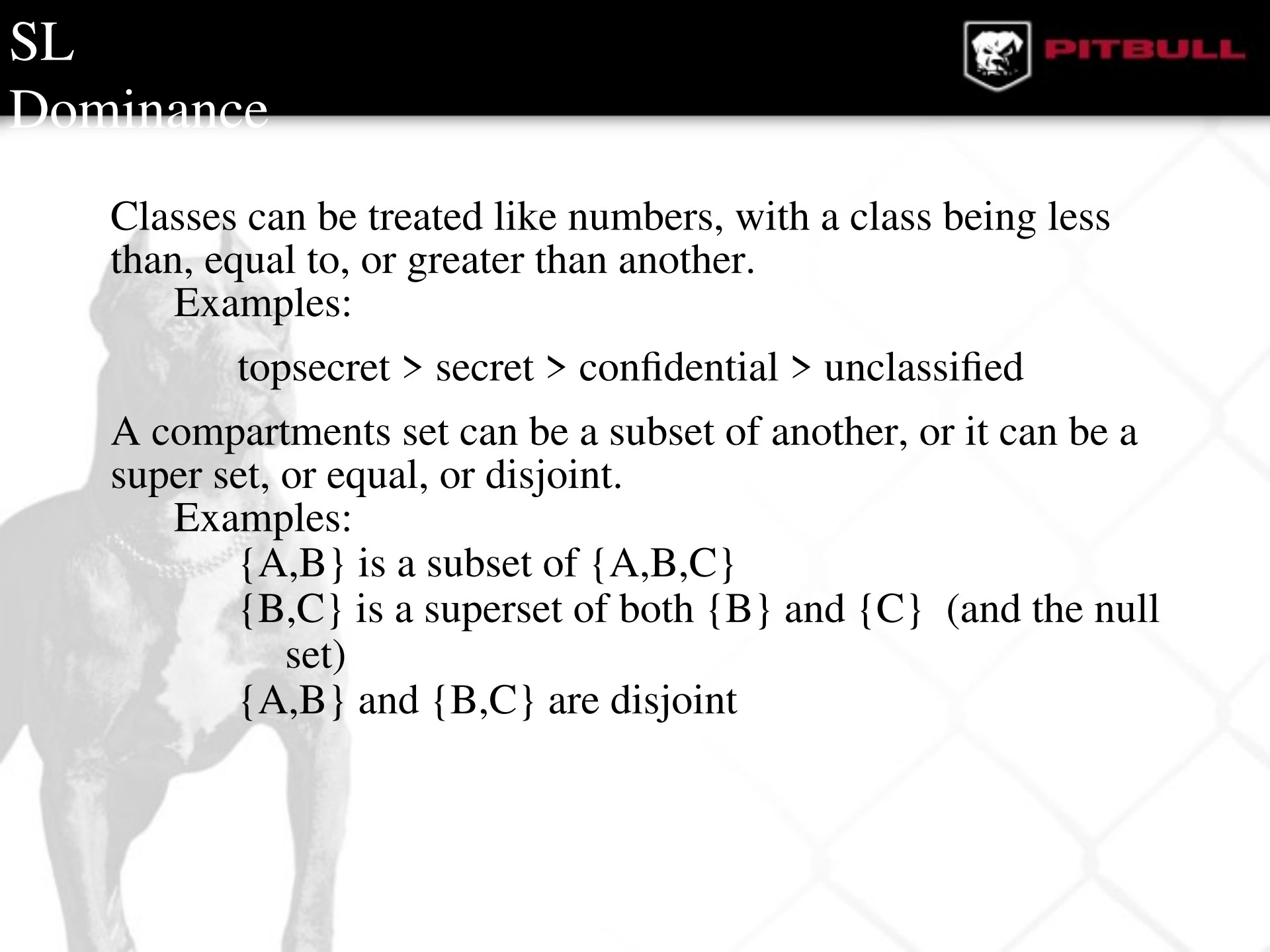
projectA, projectB, projectC

An SL is generally written with the class and compartments separated by a colon.

Examples:

TS:A,B S:A,B,C TS (no compartment)

SL Dominance



Classes can be treated like numbers, with a class being less than, equal to, or greater than another.

Examples:

topsecret > secret > confidential > unclassified

A compartments set can be a subset of another, or it can be a super set, or equal, or disjoint.

Examples:

{A,B} is a subset of {A,B,C}

{B,C} is a superset of both {B} and {C} (and the null set)

{A,B} and {B,C} are disjoint

SL Dominance
(cont'd)

PITBULL

A label (SL1) is said to dominate another label (SL2) if and only if both of the following are true:

1) the class of SL1 is greater than or equal to the class of SL2, and

2) the compartment set of SL1 is a superset of, or equals, the compartment set of SL2

Examples: (TS > S > C)

TS:A,B dominates TS:B and S:A,B and S

S:A,B,C dominates S:A and S and C:A,B,C

Equality is a special case of dominance, that is, if two labels dominate each other, they are equal.

For some labels, such as TS:A,B and S:C, neither label dominates the other.

SL Inheritance

- Every process and file on the system has an SL.
- When a process creates a file, the file is created with the SL of the process.
- When a process creates a new process (with the fork system call), the new process inherits the same SL.
- The system boots with a specific SL, which is inherited by all other processes. Some of these processes, such as the login process, can change their SLs.
- Each user account is assigned a default SL by the security officer as the account is created. This default is what a user will have as his session SL when he logs on.

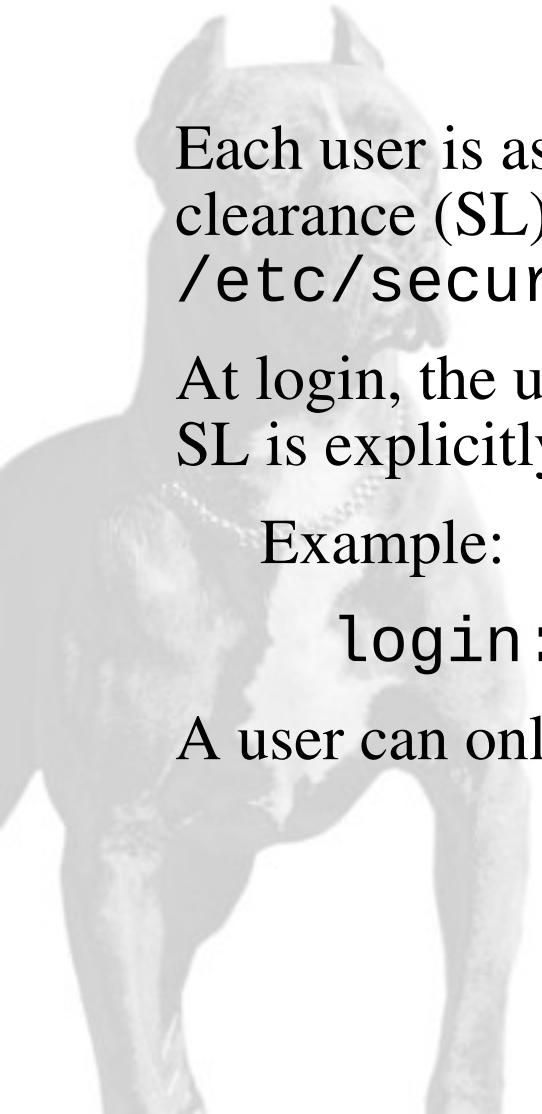
SL Access Controls

- A process cannot open a file for read unless the SL of the process dominates the SL of the file.
- A process cannot open a file for write unless the SL of the process equals the SL of the file.
- Unless a process has the privilege needed to change an SL, the process cannot change its own SL or the SL of any process or file on the system.
- This form of access control is called Mandatory Access Control (MAC).
- This means that if a process makes a copy of a file, the new file will be at the SL of the process, which is equal to or higher than the original file's SL, so the security of the copy is equal to or greater than the security of the original file.

Clearances and Max SLs

- Every process on the system has two extra SLs:
  - Minimum Clearance
  - Maximum Clearance
- These SLs are used only when the process attempts to change its own SL or the SL or another object.
- Directories and devices can have a second SL:
  - Maximum SL
- This SL is used for access control. A process can write to a device or into a directory (create or delete files) if the process SL is within the range of the device or directory's SLs instead of being restricted to being equal to the device or directory's minimum SL.

Login Label



Each user is assigned a default login SL, a minimum clearance (SL), and a maximum clearance (SL). (all stored in `/etc/security/clear`).

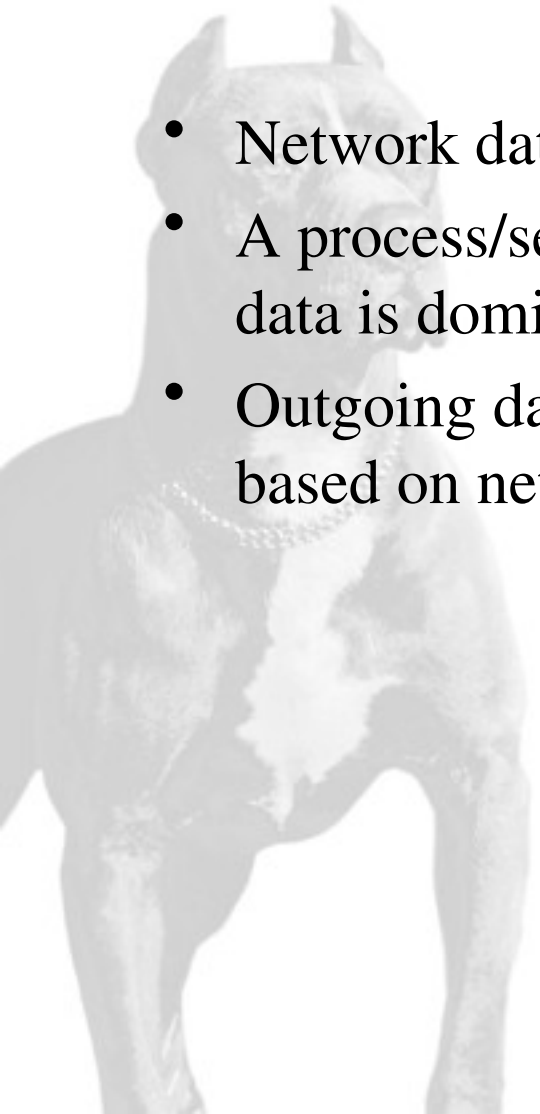
At login, the user is assigned his default login SL unless an SL is explicitly requested.

Example:

```
login: smith -e "othersl a b"
```

A user can only log in with an SL within his clearance range.

Network Labeling



- Network data is labeled based on interface, host, and port
- A process/service cannot talk to external hosts unless the data is dominated by the process
- Outgoing data is checked against a range of allowable date based on network rules

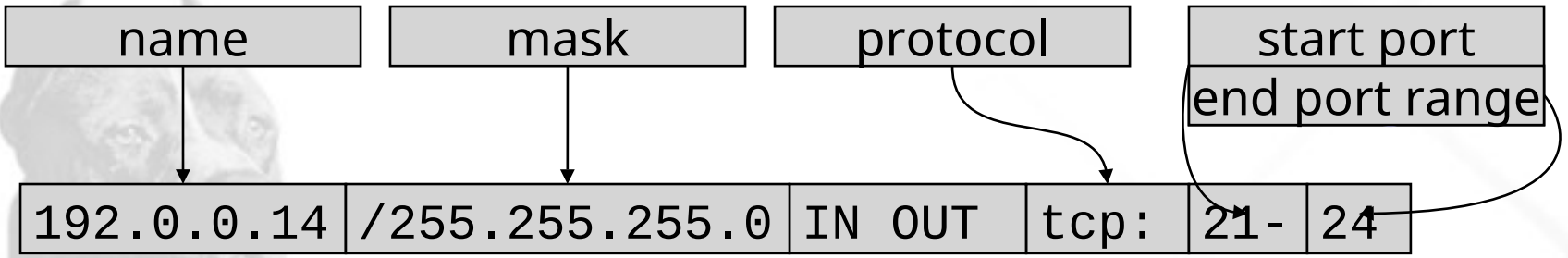
Network Security Rule

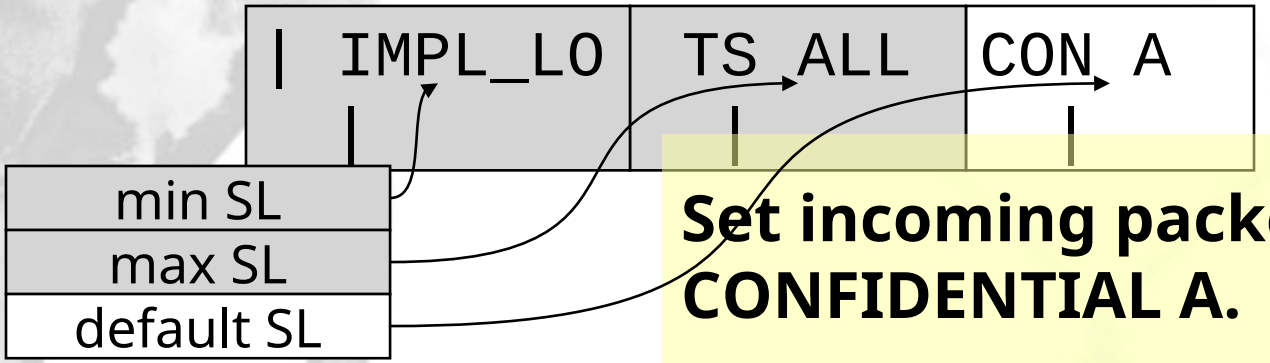
A table of network security rules is loaded into the networking stack. The rules look like this:

192.0.0.14	/255.255.255.0	IN OUT	tcp:	21-	24

IMPL_LO	TS ALL	CON A

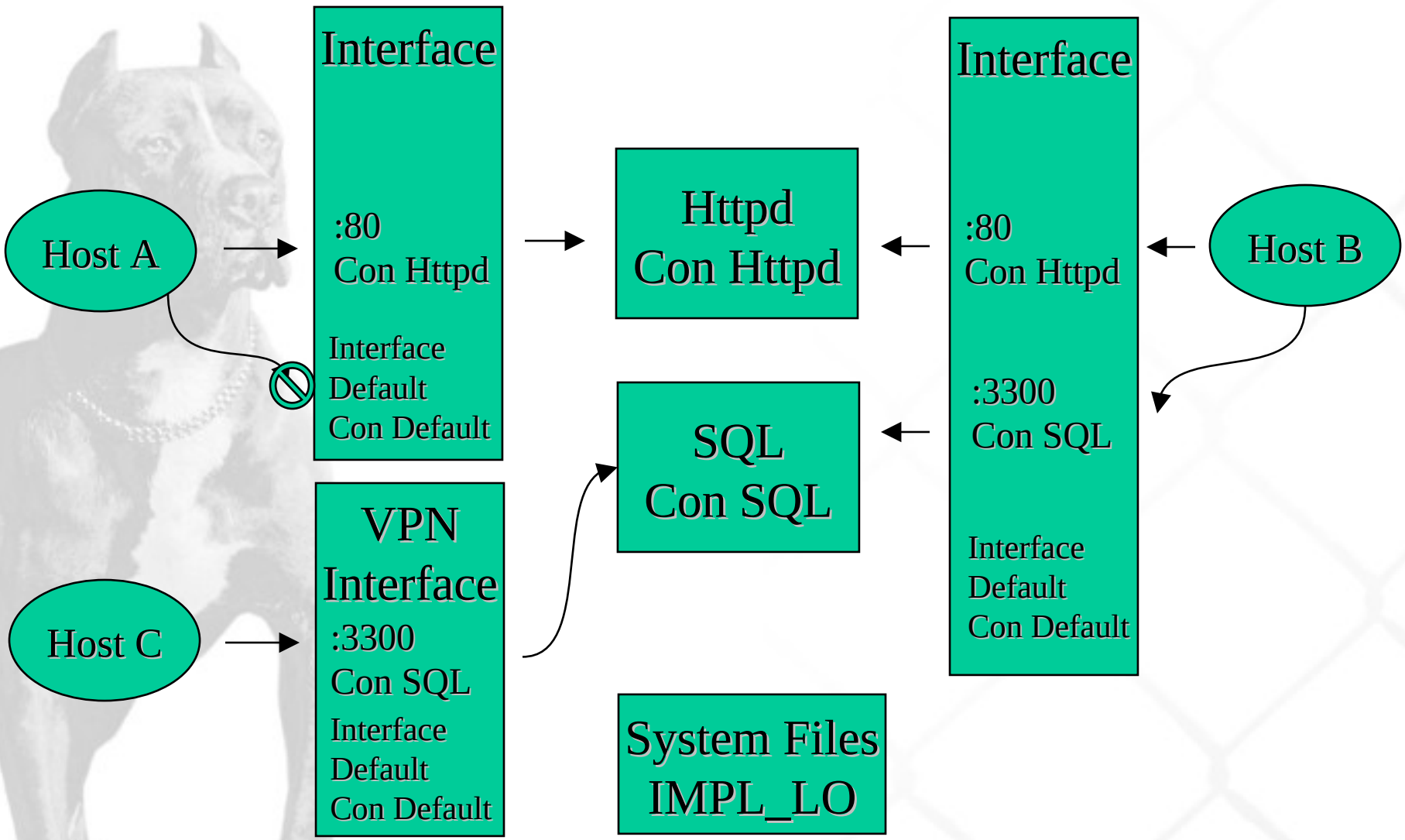
Network Security Rule

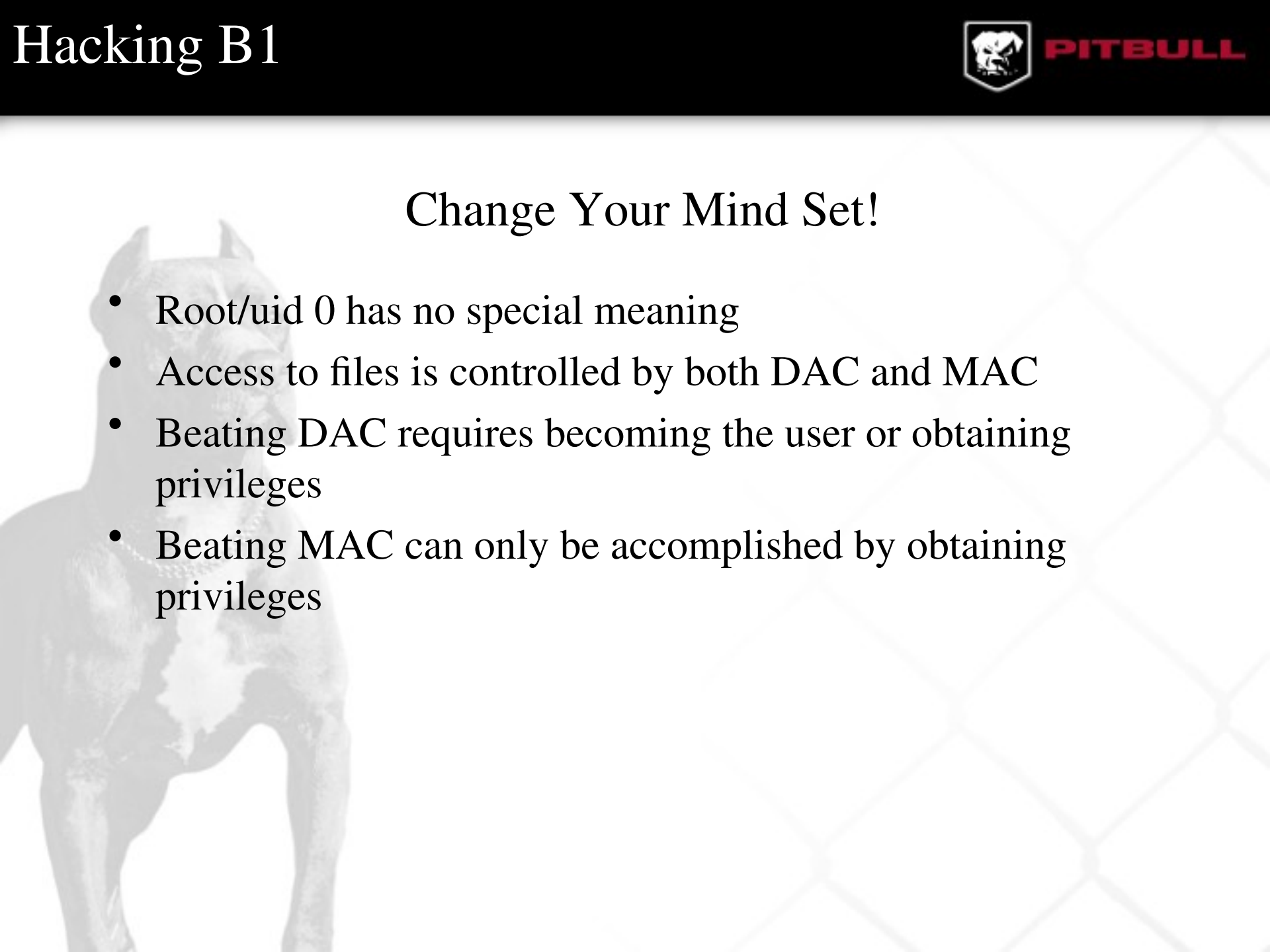




Set incoming packet SL to CONFIDENTIAL A.

Securing Services

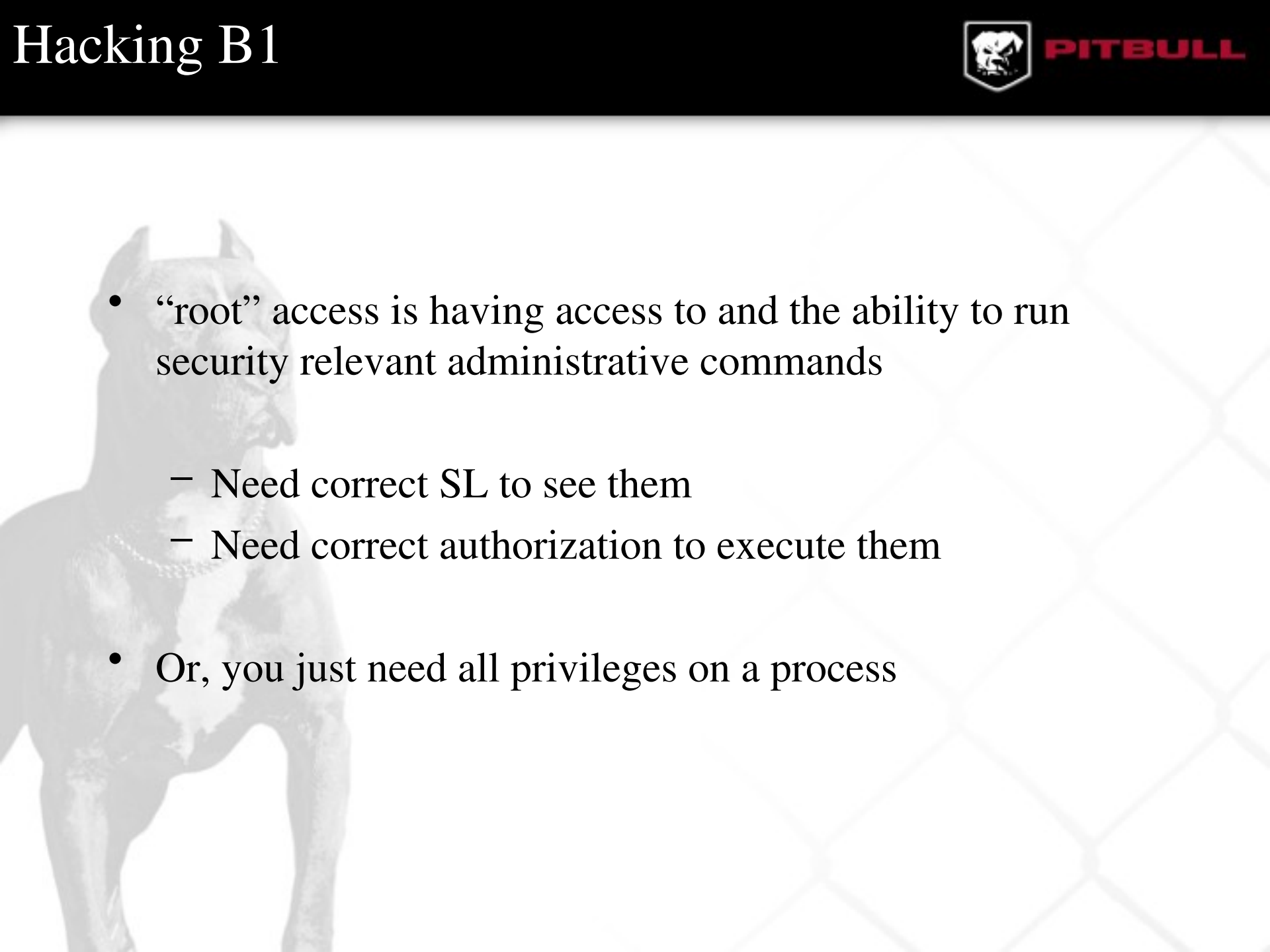




Change Your Mind Set!

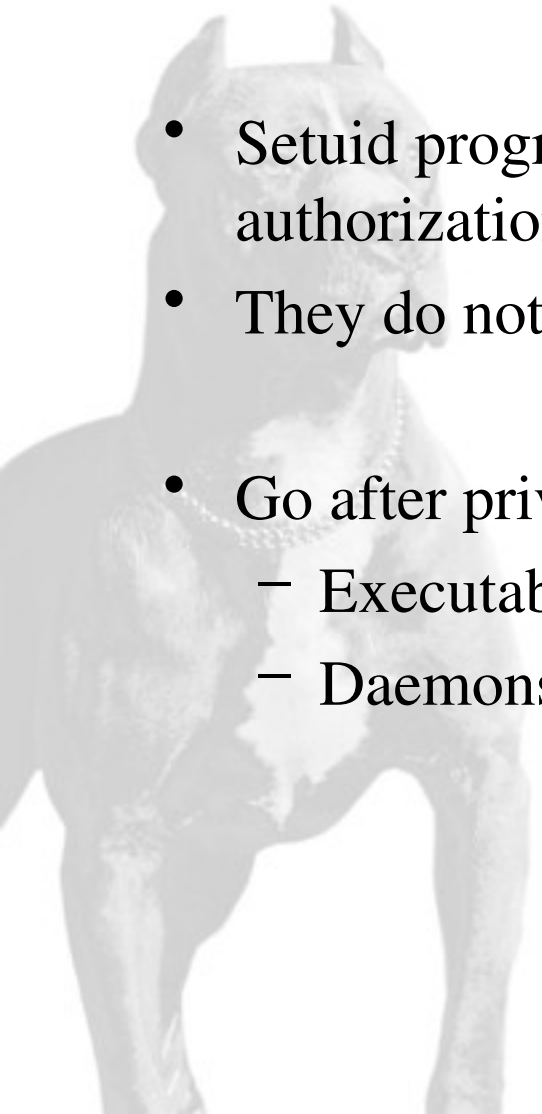
- Root/uid 0 has no special meaning
- Access to files is controlled by both DAC and MAC
- Beating DAC requires becoming the user or obtaining privileges
- Beating MAC can only be accomplished by obtaining privileges

Hacking B1



- "root" access is having access to and the ability to run security relevant administrative commands
  - Need correct SL to see them
  - Need correct authorization to execute them
- Or, you just need all privileges on a process

Going after setuid programs



- Setuid programs only change your user and possibly yield authorizations.
- They do not move you out of your effective SL.

- Go after privileged programs instead
  - Executables
  - Daemons

Attacking Privileged Programs

- Buffer Overflows
- Library Attacks

Buffer Overflows

- A BO that creates a shell will not yield privileges
- Privileges are lost across an ‘exec’.
- You’ll need new BO code that does more creative things. (Update authorizations database, clearance database, or network labeling rules)

Creating Libraries

- First thing to check is if the B1 system will use libraries not in “trusted” paths. If so, execute a program with the library path environment variable set.

- If not, then find the locations where libraries can be placed and get one in there!

Other Privileged Programs

- Programs that run with least privilege will typically yield only a limited set of privileges.
- Go after programs that yield DAC and MAC override privileges.
- Go after programs that have privileges that allow you to set privileges on processes (the key to getting all privileges!)

Getting files into system directories

- Check what the SL range is on system directories.
  - On PitBull use /tbin/secls –s <dir>
- Multilevel directories are open to attack by su'ing to a user that can write in them
- Single level directories require you to change your effective SL.

Multilevel directories

- If your effective SL is in the range of a multilevel directory and you own the directory then you can also delete the directory.
- Allows you to replace the directory with a new one with files of your own creation. (Try a whole new /lib directory)

- This allows you to circumvent MAC protection on files you do not have access to in the directory and delete them.

Getting a new SL

- Your effective SL limits what you have access to.
- Ways to change your SL
  - Setting it for the session
  - su to a new user
  - Network connections

Session SL

- Many TOSs allow you to select a session SL.
  - Must be in range of your clearance
- Under PitBull this can be done:
  - On console through login –e option
    - Login: isso –e “TS ALL”
    - Trusted ssh
      - Same syntax as login

su and your effective SL

- Check the TOS to see if ‘su - <user>’ changes effective clearance
  - This works under PitBull
- New effective SL must be in range of clearance

su and Authorizations

- Authorizations are tied to uid
- Becoming a new user such as ‘isso’ may gain you additional authorizations

- Beware the Limiting Authorization Set!

rc scripts

- rc scripts typically run with lots of privileges and thus have plenty of access to the system
- rc scripts also typically run with lots of authorizations and thus have access to all of the security relevant administrative commands

Network Labeling

- Different services may run at different SLs.
- Find one that is running at the SL you want to have access to (say for example one that allows you to execute administrative commands) and exploit it.

Kernel Bugs

- TOS security relies on the integrity of the kernel
- It goes without saying that if the kernel has a hole, then all TOS security mechanisms can be circumvented

Kernel Bugs (where to look)

- Regular old kernel bugs
- What SL is /dev/kmem at? Be the SL, get the uid and b00m it's all over.
  - Look for your process' cred structure and
    - give it all privileges
    - pick a uid you like
    - get to an SL that leads to administrative commands access.
- Access to any raw storage device can allow you to change SLs on files.
- Ioctl interfaces to device drivers tend to be less heavily scrutinized
- Are any security relevant system calls not being checked for privilege? Not likely, but it is worth looking.
- Is IPC checked for MAC? Can you cause problems with other programs using IPC?

Kernel Bugs (where to look)

- Anything not protected by MAC is only protected by DAC, and all you need now is a password.
- If you can circumvent MAC, you are back to the simpler problem of becoming a different user.

Basic Things to Always Check

- Can you log in as a highly authorized user?
  - Get the password for ‘isso’ or a security admin
  - If you are lucky, the administrative commands will be at an SL that lets you execute them.
- Are the security databases protected by MAC and DAC so that you can’t access them?

Commercial Trusted OS Platforms

- Argus PitBull .comPack
  - Solaris 7 (Sparc & x86)
  - Porting to:
    - Solaris 8 (Sparc & x86)
    - IBM AIX
    - Linux (32bit and 64bit kernels)
- Hewlett Packard
  - Virtual Vault (HP hardware)
- TrustedBSD (www.trustedbsd.com)

The Argus Revolution

The best way to learn how to do this is as always:
Get a B1 system and start securing and hacking it!

- Free PitBull Foundation Licenses for Individual Non-Commercial Use
- http://www.argusrevolution.com/

More Information

- More detailed talk on using TOS was given at BlackHat. Talk should be available on web site.
- White papers, documentation, and open discussions are available on the Argus Revolution web site as well.
- Please feel free to drop me any questions at:
  - thompson@argus-systems.com


ARGUS
SYSTEMS GROUP, INCORPORATED
www.argus-systems.com