Database Admin

Class 3 - More SQL

By Ian Robert Blair, M.S.

Agenda

- MySQL stored procedures and triggers
- MySQL Views
- MySQL User Management and Security

Triggers

- A trigger is invoked automatically when an SQL statement changes rows on a specified table
- MySQL allows a maximum of six triggers per table: before or after and insert, update, or delete
- Create trigger example:

    -
      ```
      CREATE TRIGGER staff_update_date
      BEFORE INSERT ON staff
      FOR EACH ROW SET NEW.last_update = NOW();
      ```
- DROP TRIGGER staff_update_date;

Stored Routines 1

- A **stored routine** (either a stored procedure or a stored function) allows MySQL users to define a set of statements that they can later call in a query
- A stored procedure is invoked manually with the CALL statement, taking zero or more arguments, and can pass back values through output variables
- A stored function is invoked manually by directly using its name, taking zero or more arguments and outputting a scalar value
- Reasons for stored routines:
    - Code reuse
    - Black box queries
    - Security via API
    - Security via ACL constraints

Stored Routines 2

- Example, a stored procedure that takes a store ID as input and outputs the number of films offered by the store:
    - DELIMITER |
      CREATE PROCEDURE store_offerings (IN p_store_id TINYINT UNSIGNED, OUT p_count INT UNSIGNED)
      SELECT COUNT(*) INTO p_count
      FROM inventory WHERE store_id = p_store_id;
      DELIMITER ;
- Call the procedure:
    - CALL store_offerings(1,@store_1_offerings);
    - SELECT @store_1_offerings;
- Drop procedure:
    - DROP PROCEDURE store_offerings

Transactions 1

- A transaction is a set of SQL statements that are executed as if they were one statement
- For a transaction to be finished and save data changes permanently, all the statements in the transaction have to be completed
- A transaction is a transaction only if it is ACID-compliant. ACID is an acronym that stands for atomicity, consistency, isolation, and durability.

Transactions 2

- Lets say, we want to transfer money from one account to another account

- The following transactions are required:

```
UPDATE checking SET balance = balance - 1000 WHERE id = 145356;
UPDATE checking SET balance = balance + 1000 WHERE id = 118254;
```

- If there is a failure, money is deducted from one account, but never appears in second account. Instead:

```
START TRANSACTION;
UPDATE checking SET balance = balance - 1000 WHERE id = 145356;
UPDATE savings SET balance = balance + 1000 WHERE id = 118254;
COMMIT;
```

Views

- Views are tools that assist in denormalizing data, such as for analysis, without changing the underlying system for transaction processing
- Views also allow for simplicity, abstraction, added security, and easier schema migration
- Views can be used to:
  - Restrict access to a subset of rows
  - Simplify queries

Views

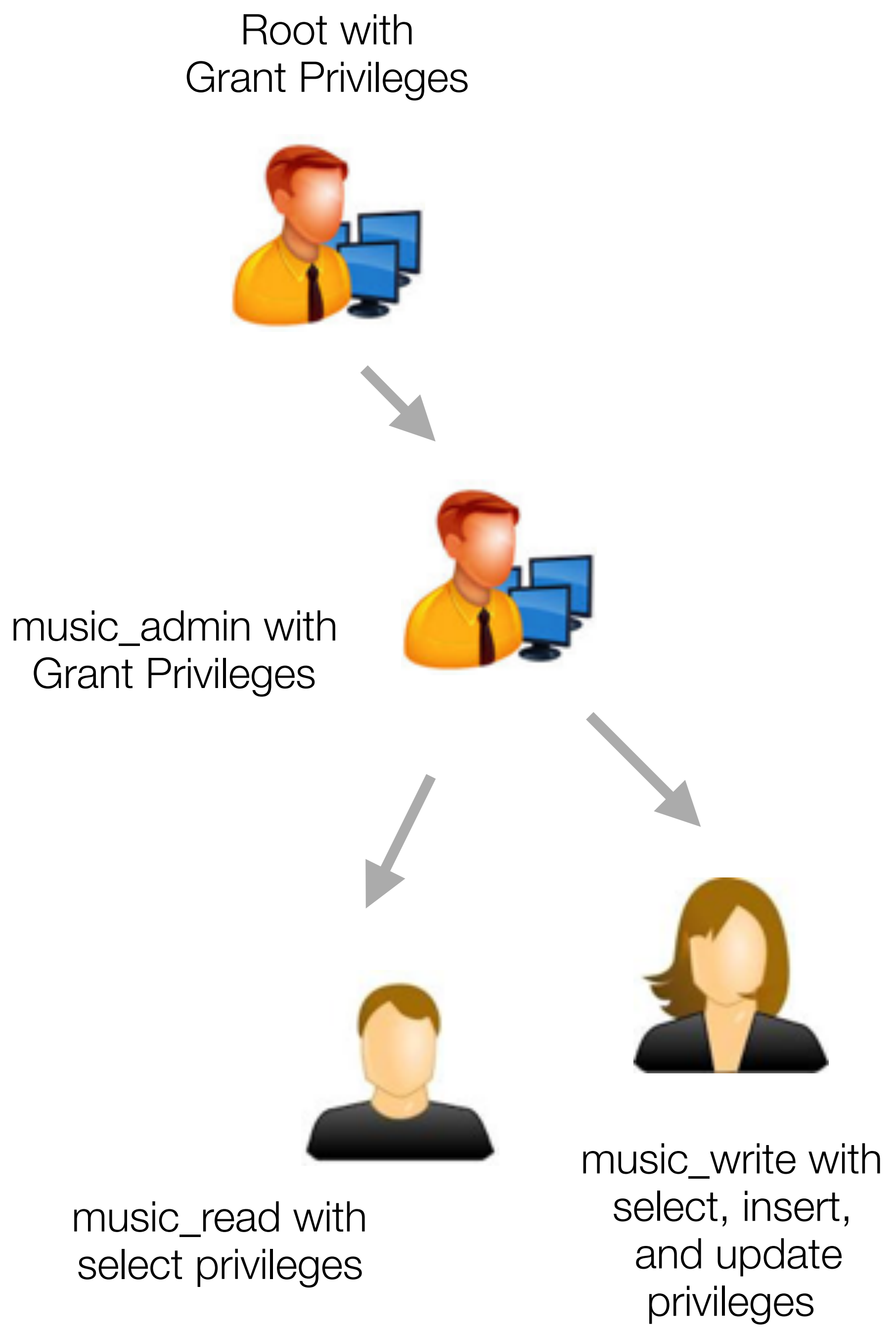
- A view is a virtual table that doesn’t store any data itself
- Instead, the data “in” the table is derived from a SQL query that MySQL runs when you access the view
- Create View statement:
  - CREATE VIEW Oceania AS
    SELECT * FROM Country WHERE Continent = 'Oceania';
- Views can be removed with
  - DROP VIEW Oceania

Updating Views

- A view is not updatable if it contains GROUP BY, UNION, an aggregate function, or any of a few other exceptions
- A query that changes data might contain a join, but the columns to be changed must all be in a single table
- Any view that uses the TEMPTABLE algorithm is not updatable
- MySQL doesn’t support materialized views
- A materialized view generally stores its results in an invisible table behind the scenes, with periodic updates to refresh the invisible table from the source data

Least Privilege

- The MySQL server comes with the user root, who can do everything on the MySQL server, including creating and deleting users, databases, tables, indexes, and data
- Most applications don't need superuser privileges for day-to-day activities
- You can define less powerful users who have only the privileges they need to get their jobs done
- For example, you could have a user music_admin who can perform any database operation on the music database, and the user music_read who can read data from the music database but can't change anything


Root with Grant Privileges
music_admin with Grant Privileges
music_read with select privileges
music_write with select, insert, and update privileges

User Accounts, pt. 1

- To create a new user, you need to have permission to do so (ie. root)
    - CREATE USER 'paola'@'localhost’ IDENTIFIED BY 'her_password',
- The GRANT statement can create and give privileges to users
- To grant with all rights to the music database
    - GRANT ALL ON music.* TO 'allmusic'@'localhost' IDENTIFIED BY 'the_password';
- ‘localhost’ specifies that the user can connect only from the localhost
- The “GRANT OPTION” privilege allows auser to pass on any privileges she has to other users
    - GRANT GRANT OPTION ON music.* TO 'hugh'@'localhost';

User Accounts, pt. 2

- Privileges can be granted at the global, database, table, or column level (see tables 9-1, 9-2 in Learning MySQL, pg. 304)
- To create a less privileged user:
  - GRANT ALL ON music.artist TO 'partmusic'@'localhost' IDENTIFIED BY 'the_password';
  - GRANT ALL ON music.album TO 'partmusic'@'localhost';
  - GRANT SELECT (track_id, time) ON music.track TO 'partmusic'@'localhost';
- To see a full list of all available privileges run:
  - SHOW GRANTS [for user@host]
- To see current user:
  - SELECT CURRENT_USER();

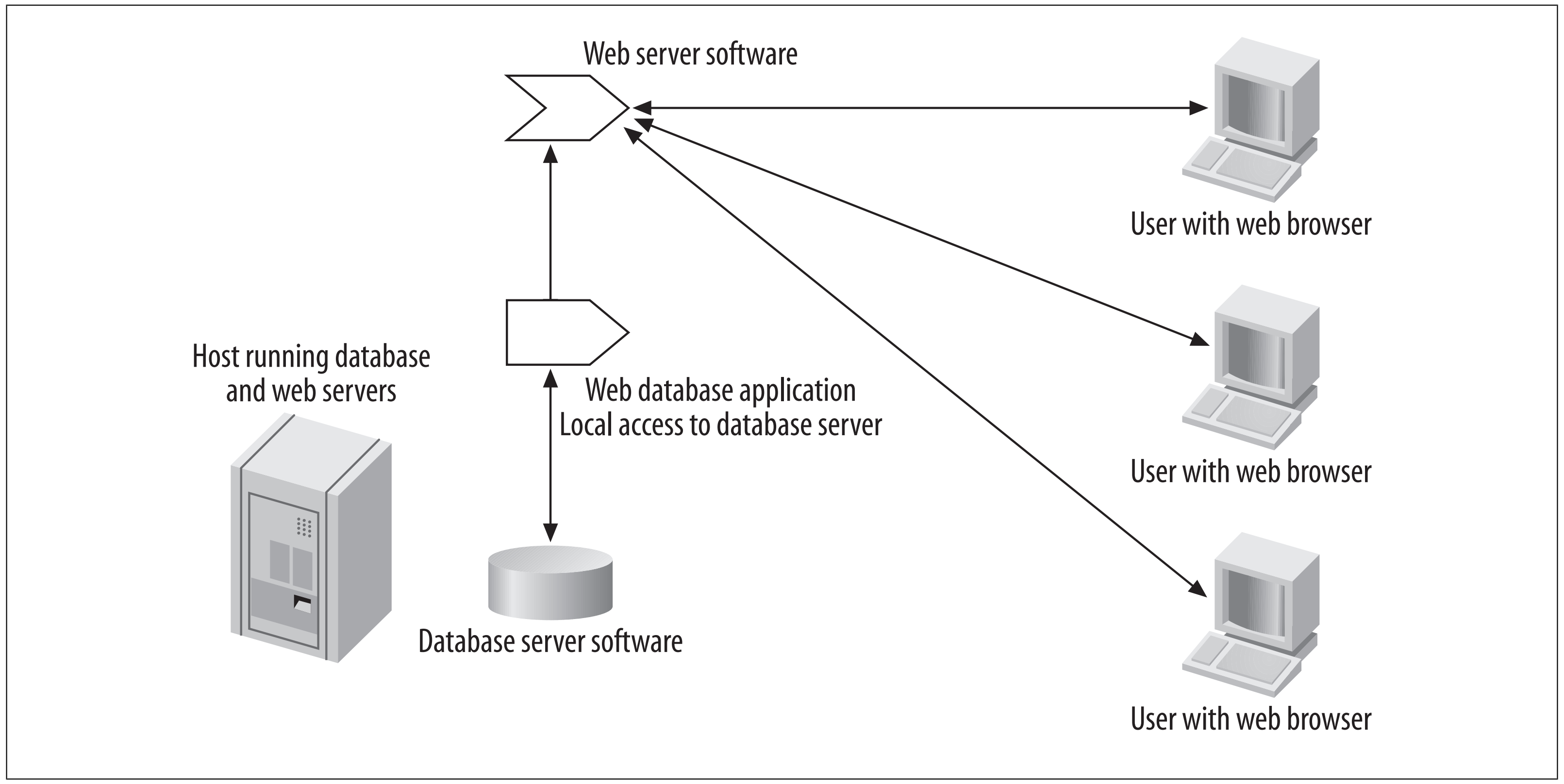
User Accounts

- MySQL comes with 2 default users:
    - root - super user who can do anything
    - anonymous - limited privilege user (no username specified)

Best Practices

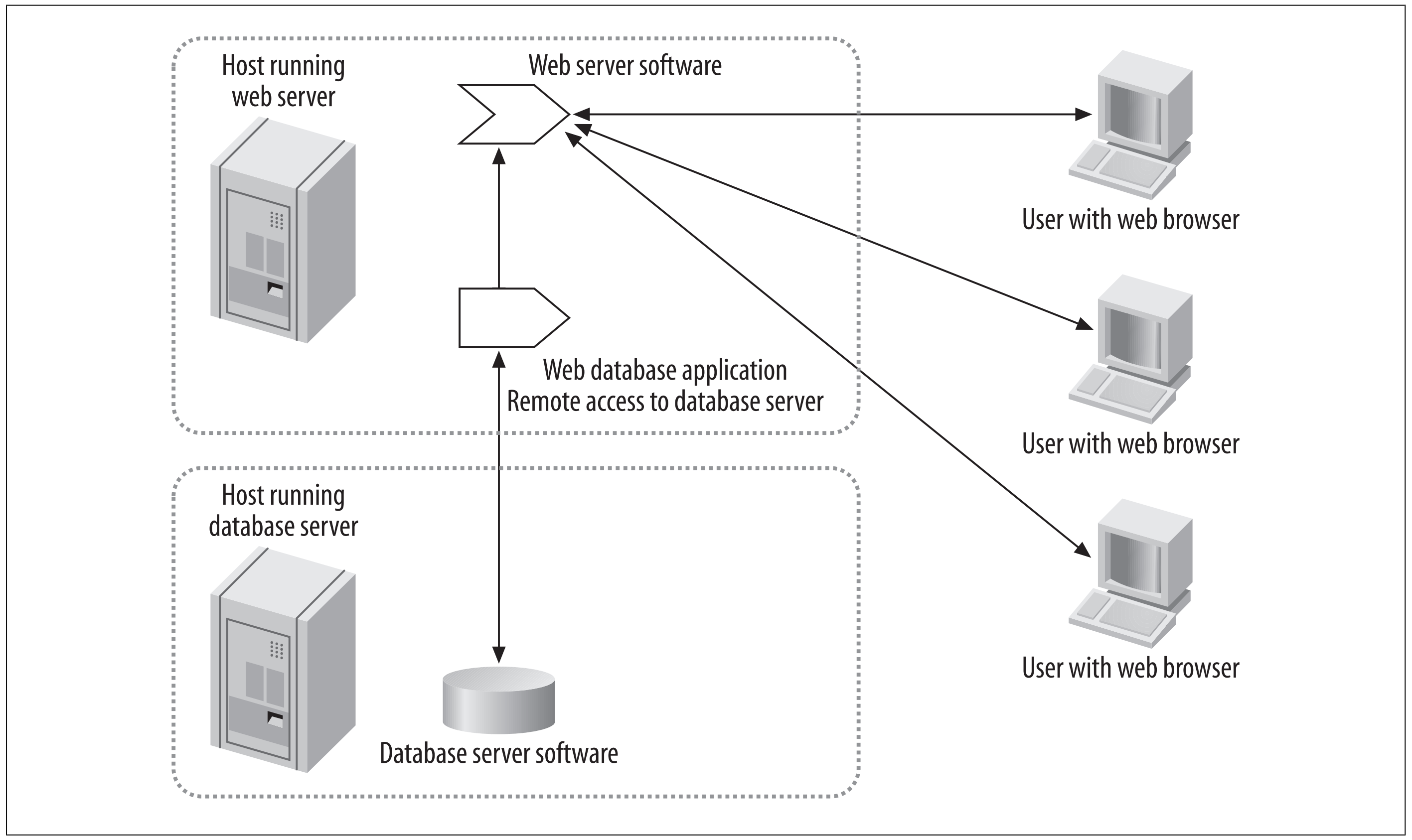
- Set a strong password for root
- Remove anonymous access
    - DROP USER ''@'localhost';
- Drop the test database
- Remove entirely or strongly restrict remote access
    - DELETE FROM mysql.user WHERE Host <> 'localhost';
- Implement a default deny security policy on databases, tables, and privileges
- Use the OS firewall to secure your server (Windows Firewall or Linux/Unix IP Tables)

Application Data Access, pt. 1



The web application users only require local access to the database server

Application Data Access, pt. 2



Access can be granted to users only from a remote web application servers

Remote Access

- If you want to allow a user to connect to the server from another computer, you must specify the host from which they can do so (the remote client)
    - GRANT ALL ON *.* TO 'hugh'@'192.168.1.2' IDENTIFIED BY 'the_password';
- The wildcard character % matches any string
- For example, this command allows jill to connect from any machine in the domain
    - GRANT ALL ON *.* TO 'jill'@'%.invyhome.com' IDENTIFIED BY ‘the_password';
- bind-address = * in my.cnf

Connect Remotely

- To connnect remotely
    - mysql -u username -p -h hostname
- To see status on connection
    - status

Revoking Rights and Dropping Users

- You can selectively revoke privileges with the REVOKE statement, which essentially has the same syntax as GRANT
  - REVOKE SELECT (track_id) ON music.track FROM 'partmusic'@'localhost';
  - REVOKE ALL PRIVILEGES ON music.artist FROM 'partmusic'@'localhost';
  - REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'partmusic'@'localhost';
- You can remove access to the MySQL server by removing a user
  - DROP USER 'partmusic'@'localhost';

Changing Passwords

- Passwords can be changed by re-issuing grant command
  - GRANT USAGE ON *.* TO 'selina'@'localhost' IDENTIFIED BY ‘*another_password*';
- Also issue a set password statement
  - SET PASSWORD=PASSWORD(‘the_password');
  - SET PASSWORD FOR 'selina'@'localhost' = PASSWORD('another_password');

View Users

- View all users
    - SELECT User,Host FROM mysql.user;

Resource Limits

- MySQL allows you to set
  - MAX_QUERIES_PER_HOUR
  - MAX_UPDATES_PER_HOUR
  - MAX_CONNECTIONS_PER_HOUR
- For example:

```
GRANT USAGE ON *.* to 'partmusic'@'localhost' WITH
MAX_QUERIES_PER_HOUR 100
MAX_UPDATES_PER_HOUR 10
MAX_CONNECTIONS_PER_HOUR 5;
```

MySQL Database Tables

- The user table holds user account
- The db, tables_priv, column_priv, hold privileges for all databases, tables, and columns

Reset Root Password

- mysqld_safe --skip-grant-tables
- FLUSH PRIVILEGES;
- SET PASSWORD for 'root'@'localhost'=PASSWORD('the_new_mysql_root_password');
- FLUSH PRIVILEGES;

Encrypted Client/Server Connection 1

- Obtain a valid certification (open SSL)
- Add the following to the options file on your server under [mysqld]
  - ssl-ca=/path/to/cacert.pem
  - ssl-cert=/path/to/mysql-server.crt
  - ssl-key=/path/to/mysql-server.key
- When the client connects use:
  - mysql -u username -p --ssl-ca=/path/to/cacert.pem
- Check session:
  - SHOW SESSION STATUS LIKE `ssl_cipher’

Encrypted Client/Server Connection 2

- Require SSL encryption for access to a table:
  - CREATE USER read_secure@localhost IDENTIFIED BY `secure_pass’ ;
  - GRANT SELECT ON sakila.* TO read_secure@localhost REQUIRE SSL;

Linux/Unix Firewall

- **netfilter** component is a set of tables that hold rules the kernel uses to control network packet filtering
- **iptables** utility sets up, maintains, and displays the rules stored by netfilter
- A **rule** comprises one or more criteria (matches or classifiers) and a single action (**a target**)
- Rules are stored in **chains**
- Each rule in a chain is applied, in order, to a packet until a match is found
- If there is no match, the chain's **policy,** or default action, is applied to the packet

Linux/Unix Firewall, pt. 2

- Chains are collected in three tables: **Filter, NAT,** and **Mangle**
- The Filter table
  - Used to DROP or ACCEPT packets based on their content; it does not alter packets
  - Builtin chains are **INPUT, FORWARD,** and **OUTPUT**
  - All user-defined chains go in this table
- The NAT table:
  - Used exclusively to translate the source or destination fields of packets
  - Builtin chains are **PREROUTING, OUTPUT,** and **POSTROUTING**
  - **DNAT (destination NAT)** alters the destination IP address of the first inbound packet in a connection so it is rerouted to another host

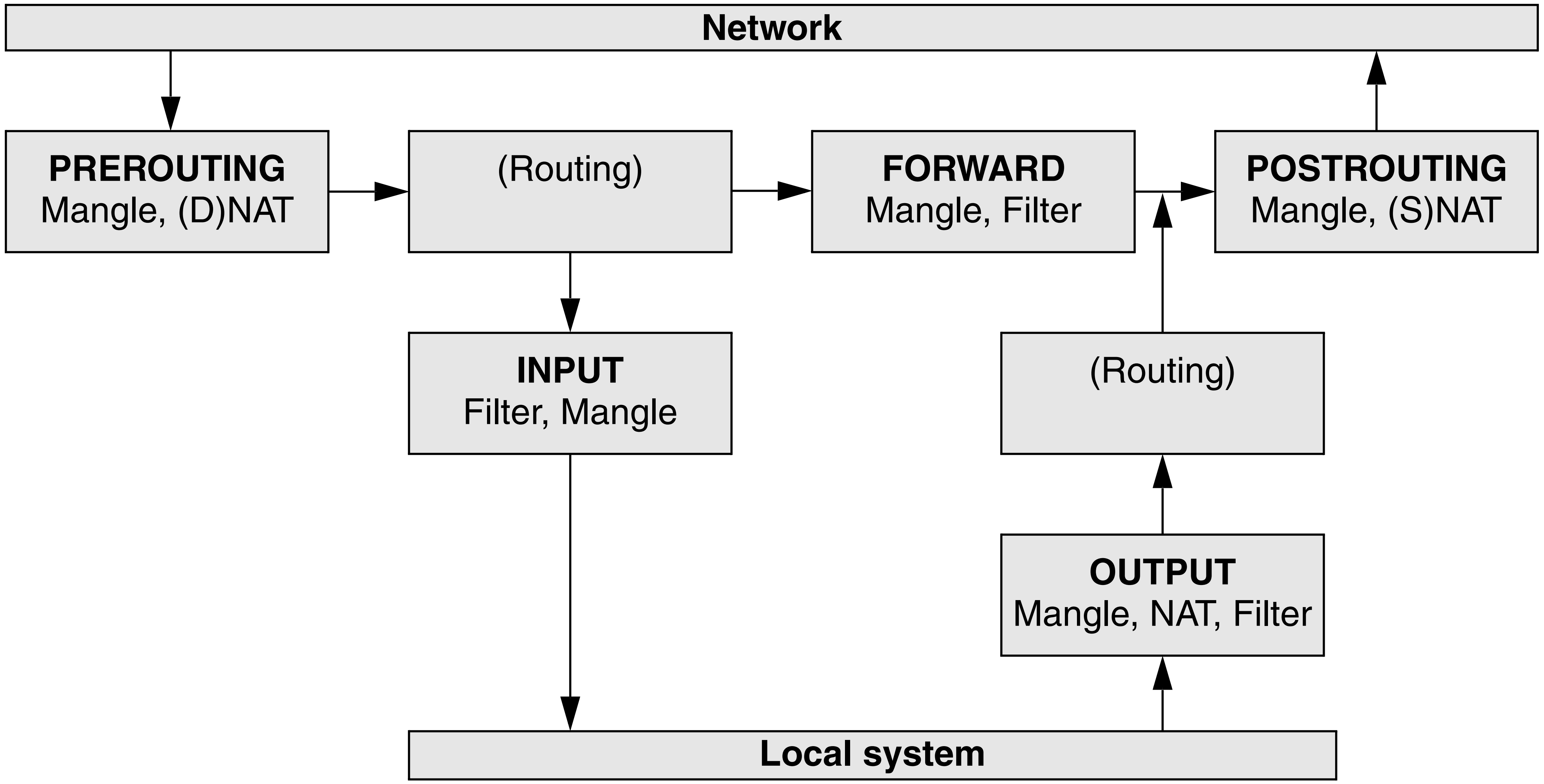
Linux/Unix Firewall, pt. 3

- The NAT Table, cont.:
  - **SNAT (source NAT)** alters the source IP address of packets so it appears to come from a different fixed IP address—(for example, a firewall or router)
  - **MASQUERADE** differs from SNAT only in that it checks for an IP address to apply to each outbound packet, making it suitable for use with dynamic IP addresses such as those provided by DHCP
- The Mangle Table:
  - Used exclusively to alter the TOS (type of service), TTL (time to live), and MARK fields in a packet. Builtin chains are PREROUTING and OUTPUT

Linux/Unix Firewall, pt. 4

- State
  - Connection tracking is handled by the **conntrack** module
  - Rules can match criteria based on the state of the connection the packet is part of
  - Connections classified as NEW, RELATED, INVALID or ESTABLISHED
- Jump or Target
  - Specifies the action the kernel takes if a network packet matches all the match criteria

Packet Filtering Diagram



IP Tables Utility

- The **iptables** utility is a tool that manipulates rules in the kernel
- Iptables commands can be put in a script and run script each time the system boots (called from **/etc/rc.d/rc.local**) or you can put the arguments in **/etc/sysconfig/iptables**
- The **iptables-save** utility copies packet filtering rules from the kernel to standard output
- The **iptables-restore** utility copies rules from standard input, as written by iptables-save, to the kernel

IP Tables Commands

- --append or –A, Adds rule(s) specified by rule-specifications to the end of chain
- --delete or -D, Removes one or more rules from chain
- --insert or –I, Adds rule(s) specified by rule-specifications and target to the location in chain specified by rule-number
- --list or –L, Displays the rules in chain
- --flush or -F, Deletes all rules from chain
- --policy or –P, Sets the default target or policy builtin-target for the builtin chain
- --help or -h, Displays help

IP Tables Match Criteria

- --protocol or -P [!], Matches if the packet uses the protocol
- --source or -S, Matches if the packet came from address
- --destination or -D, Matches if the packet is going to address
- --in-interface or -i, Matches if iface is the name of the interface the packet was received from
- --out-interface or -o, Matches if iface is the interface the packet will be sent to
- --destination-port or --dport, Matches a destination port number, service name, or range of ports (1025:, 80:88)
- --source-port or --sport, Matches a source port number, service name, or range of ports
- --state, ESTABLISHED, INVALID, NEW, and RELATED

IP Tables Targets

- ACCEPT, Causes the packet to leave the chain
- DNAT, Rewrites the destination address of the packet
  - --to-destination ip[-ip][:port-port]
- DROP
- LOG
- MASQUERADE, Similar to SNAT (below) with —to-source, except that it grabs the IP information from the interface on the specified port
- REJECT, Similar to DROP, except it notifies the sending system that the packet was blocked
- SNAT, Rewrites the source address of the packet (--to-ports port[-port])

IP Tables Examples 1

Flush all Rules

```
iptables -F
```

Set Default Policies

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

SSH Rules

```
iptables -A INPUT -i eth0 -p tcp -s 192.168.10.0/24 --dport ssh -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp -d 192.168.10.0/24 --sport ssh -m state --state ESTABLISHED -j
ACCEPT
```

SMTP Rules

```
iptables -A INPUT -i eth1 -p tcp --dport smtp -m state --state NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth1 -p tcp --sport smtp -m state --state NEW,ESTABLISHED -j ACCEPT
```

IP Tables Examples 2

```
iptables -A INPUT -i lo -p tcp --dport 3306 -j ACCEPT

iptables -A OUTPUT -o lo -p tcp --sport 3306 -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT

iptables -A OUTPUT -p udp -o eth0 --dport 53 -j ACCEPT

iptables -A INPUT -p udp -i eth0 --sport 53 -j ACCEPT

iptables -A INPUT -i eth0 -p tcp -s 192.168.100.0/24 --dport 3306 -m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp --sport 3306 -m state --state ESTABLISHED -j ACCEPT
```

Connection Options

Option	Usage	Description
bind-address	bind-address=192.168.1.1	Binds the mysql server to the specified IP address
port	port=3307	Specifies the listening port
skip-name-resolve	skip-name-resolve	Don’t do DNS lookups
skip-networking	skip-networking	mysql will not listen on TCP/IP Port

Additional Security

- A Root kit kit is a package of software that allows a hacker to “root” access to a system
- A root kits can clean logs, hide directories, users, and network connects by replacing system binaries and in some cases
- To check for root kits run:
    - rkhunter
- Additonally you may want to install a host based intrusion detection systems like the following:
    - AIDE, Tripwire, OSSEC

Lab 1

- Write the SQL statements to create the database from the previous homework.
- Create 3 users for the database you created in the previous class, and grant appropriate rights using the “least privilege” philosophy
- Add 6 more query statements (must draw from at least 2 tables)

Lab 2

- Add databases from the book using the “source” or “<“
- Backup a database using mysqldump, delete the database or some tables, and restore it
- Schedule a daily backup of your databases

HW

- Read ‘Learning MySql” Chapters 7-9
- Do page 275, Ex. 1-2
- Do page 349, Ex. 1-5
- Do page 369, Ex. 1-5