Setting Up Database Access

To obtain read-only access to the database, you must first configure the appliance to allow access. Then, you must configure your client system to connect to the appliance by downloading the JDBC driver and accepting the SSL certificate from the appliance you want to access. Finally, you must configure your reporting application to connect to the appliance.

Note Before you set up database access, you should make sure you have fulfilled the prerequisites described in Prerequisites, page 1-13.

For more information, see the following sections:

Creating a Database User Account

License: Any

To configure access to the Firepower System database, you must first create a user account and assign it permission to access the Firepower System database. You can grant this permission by assigning the account either the system-provided user role of External Database User or a custom user role created by your organization that includes the External Database User permission. See the *Firepower Management Center Configuration Guide* for information on creating the user account and viewing the permissions in a given user role.

Warning **External Database Access is a Global privilege. A user with External Database Access will be able to query information for all domains.**

Tip Users assigned the system-provided Administrator role have the External Database User permission by default.

In a multidomain deployment, you can create user accounts in any domain in which you have been assigned Admin access. However, the External Database User role is only available at the Global domain level. External Database Users can access all events regardless of domain.

Enabling Database Access on the Firepower Management Center

License: Any

After you create an External Database user, you must configure the Firepower Management Center to allow access to the database on the appliance. You must also configure a database access list on the appliance and add all host IP addresses that will query the external database.

To enable database access:

Access: Admin

Step 1 On the Firepower Management Center, select **System > Configuration**.

Step 2 Click **External Database Access** on the left.

The Database Settings menu appears.

Step 3 Select the **Allow External Database Access** check box.

The **Access List** field appears.

Step 4 Type the fully qualified domain name (FQDN), or IPv4 address, of the Firepower Management Center in the **Server Hostname** field, depending on your third-party application requirements. You cannot use an IPv6 address as you cannot use an IPv6 address to install a certificate.

If you type an FQDN, you must make sure that the client can resolve the FQDN of the Firepower Management Center. If you type an IP address, you must make sure that the client can connect to the Firepower Management Center using the IP address.

Step 5 To add database access for one or more IP addresses, click **Add Hosts**.

An **IP Address** field appears in the **Access List** field.

Step 6 In the **IP Address** field, you have the following options, depending on the IP addresses you want to add:

- an exact IPv4 address (for example, `192.168.1.101`)
- an exact IPv6 address (for example, `2001:DB8::4`)
- an IP address range.
  - For information on using IP address ranges in the Firepower System, see IP Address Conventions in the *Firepower Management Center Configuration Guide*.
- **any**, to designate any IP address

Step 7 Click **Add**.

The IP address is added to the database access list.

Step 8 Optionally, to remove an entry in the database access list, click the delete icon ( ).

Step 9 Click **Save**.

The database access settings are saved.

Step 10 Continue with the procedure in the next section, Downloading the JDBC Driver.

Downloading the JDBC Driver

License: Any

After you create an External Database user and configure the Firepower Management Center to allow database access, download the JDBC driver to the client system. You must use this JDBC driver to connect to the database.

To download the JDBC driver:

Access: Admin

Step 1 On the Firepower Management Center, select **System > Configuration**.

Step 2 Click **External Database Access** on the left.

The Database Settings menu appears.

Step 3 Next to **Client JDBC Driver**, click **Download** and follow your browser's prompts to download the **`client.zip`** package.

Step 4 Unpack the ZIP package. Note the location.
Make sure you preserve the file structure of the package.
The driver, along with other files, is packaged in a ZIP file (**`client.zip`**). The package contains the following directories:

- **`bin`**, which contains a sample client called RunQuery, as well as the executable files you use to install the certificate for encrypted communication between your client and the Firepower Management Center
- **`lib`**, which contains JDBC driver JAR files
- **`src`**, which contains source code for the executable files in the **`bin`** directory

Step 5 Continue with the procedure in the next section, Installing the Client SSL Certificate.

Installing the Client SSL Certificate

Once you have downloaded the JDBC driver, use the system-provided program named InstallCert to accept and install the SSL certificate from the Firepower Management Center. Your client system and the Firepower Management Center communicate securely with SSL certificate authentication. When you accept the certificate, your computer adds it to the keystore (**`jssecacerts`**) in the **`security`** directory of the currently running JRE:

```
$JAVA_HOME/jre[version]/lib/security
```

The following represent common locations of the keystore for computers running Microsoft Windows and UNIX, respectively:

- `C:\Program Files\Java\jre[version]\lib\security\jssecacerts`
- `/var/jre[version]/lib/security/jssecacerts`

Note If the Java query application you plan to use to access the database access function uses a different JRE, you must copy the keystore to the **`security`** directory of the other JRE.

To install the SSL certificate using InstallCert:

Step 1 On your computer, open a command line interface.

Step 2 At the command prompt, change to the **`bin`** directory created when you unpacked the ZIP package.

Step 3 To install the Firepower Management Center's SSL certificate, type the following and press Enter:

```
java InstallCert defense_center
```

where *`defense_center`* is either the FQDN or the IP address of the Firepower Management Center. InstallCert does not support IPv6 addresses. If you are on an IPv6 network, you must use a resolvable hostname.

Output similar to the following example from a computer running Microsoft Windows appears:

```
Loading KeyStore C:\Program Files\Java\jre6\lib\security...
Opening connection to defensecenter.example.com:2000...
Starting SSL handshake...
Subject GENERATION=server, T=vjdbc, O="Cisco, Inc.",
...
...
```

You are prompted to view the certificate.

Step 4 Optionally, view the certificate.

You are prompted to accept the certificate.

Step 5 Accept the certificate.

The certificate is accepted, and output similar to the following example from a computer running Microsoft Windows appears:

```
Added certificate to keystore 'C:\Program Files\Java\jre6\lib\security\jssecacerts'
using alias 'defensecenter.example.com-1'
```

If you plan to use Crystal Reports, note the location of the keystore (`jssecacerts`). You will need this information later.

Step 6 You have the following options:

- If you have a third-party application, continue with the procedure in the next section, Connecting to the Database Using a Third-Party Application, page 2-5.
- If you have a custom application, continue with the procedure in Connecting to the Database Using a Custom Program, page 2-6.

Connecting to the Database Using a Third-Party Application

After you install the certificate, you can query the database on a Firepower Management Center using any third-party client that supports JDBC SSL connections. The following table lists information you may need to configure a connection between your client and the Firepower Management Center.

Table 2-1 Connection Information for Database Access Clients

Information	Description
JDBC URL	The following JDBC URL identifies the Firepower System database so the JDBC driver on your client can establish a connection with it: `jdbc:vjdbc:rmi://defense_center:2000/VJdbc,eqe` where *defense_center* is either the FQDN or the IP address for the Firepower Management Center.
JDBC driver JAR files	You must use the following JAR files when you configure a connection to the Firepower System database: • `vjdbc.jar` • `commons-logging-1.1.jar` These files are located in the **`lib`** subdirectory where you unpacked the **`client.zip`** file you downloaded and unpacked, as described in Downloading the JDBC Driver, page 2-3.
JDBC driver class	You must use the following driver class when you configure a connection to the Firepower System database: `com.sourcefire.vjdbc.VirtualDriver`
user name and password	To connect to the database on an appliance, use a user account that has the External Database User permission. For more information, see Creating a Database User Account, page 2-1.

The following sections contain tips for connecting to the Firepower System database using three popular industry-standard reporting tools. Whether you use one of these tools or another Java-based application, you should refer to the documentation for your reporting tool for detailed instructions on how to create a JDBC SSL connection.

Crystal Reports

The following is valid for installing Crystal Reports 2011 on a 32-bit Windows environment. If you run a 64-bit Windows environment, the filepaths may be different.

To allow Crystal Reports 2011 to connect to the Firepower System database, you must:

– Add the JDBC driver JAR files that you downloaded from the Firepower Management Center to the Crystal Reports classpath. Assuming a default installation of Crystal Reports, you can edit the classpath section in the following file:

```
C:\Program Files\SAP BusinessObjects\SAP Business Objects
Enterprise XI 4.0\Java\CRConfig.xml
```

- Copy the keystore that was created when you installed the client SSL certificate to the appropriate Crystal Reports security directory. Assuming a default installation of Crystal Reports, that directory is:

  ```
  C:\Program Files\SAP BusinessObjects\SAP Business Objects
  Enterprise XI 4.0\win32_x86\jdk\jre\lib\security
  ```

- Create a new JDBC (JNDI) connection with the Database Expert, using `Cisco` as the database name.

JasperSoft iReport

To allow iReport to connect to the Firepower System database, you must:

- Add the JDBC driver JAR files that you downloaded from the Firepower Management Center to the iReport classpath.
- Add a new JDBC driver using the JDBC driver JAR files that you downloaded from the Firepower Management Center. After you add the driver files, iReport should find the correct driver class.
- Create a new database connection using the driver you just created.

Actuate BIRT

To allow BIRT to connect to the Firepower System database, you must:

- Add a driver definition using the **Generic JDBC Driver** template.
- Create a new database connection using the **Generic JDBC** profile type.
- Create a data source for reports using the **JDBC Data Source** data source type.

Tip If you cannot select the Cisco driver class when creating a new JDBC data source profile, add the driver using the JDBC driver JAR files you downloaded from the Firepower Management Center.

Connecting to the Database Using a Custom Program

Once you install the certificate, you can enable custom Java report tools to query the Firepower System database. Cisco provides a sample Java command line application named RunQuery that establishes the required SSL connection using the JDBC driver provided with your Firepower Management Center. RunQuery retrieves both table records and table metadata. The source code is included in the **`src`** directory of the ZIP package you downloaded from the Firepower Management Center. See Downloading the JDBC Driver, page 2-3.

Note RunQuery is a sample client only, **not** a fully featured reporting tool. Cisco **strongly** recommends against using it as your primary method of querying the database. For information on using RunQuery, refer to the README file included in the ZIP package.

See the following for more information on connecting to the database using a custom program:

- Sample Code for Custom Java Programs, page 2-7 describes the Java classes and methods that the RunQuery application uses to set up the database connection and submit queries.
- Running the Application, page 2-8 discusses environment requirements necessary for your Java application to execute.

Sample Code for Custom Java Programs

The RunQuery source code uses the functions discussed below. These code samples illustrate one of several possible implementation approaches.

Dynamically setting the SSL provider connection

After you install the SSL security certificate on your client (see Installing the Client SSL Certificate, page 2-3), you can dynamically register your JSSE provider with the following line in your program:

```
Security.addProvider(new com.sun.net.ssl.internal.ssl.(Provider());
```

Initializing the JDBC driver for your program

You can load the JDBC driver class in your Java application using the `Class.forName()` method, as follows:

```
Class.forName("com.sourcefire.vjdbc.VirtualDriver").newInstance();
```

If your program launches from the command line, the user supplies the JDBC class as follows:

```
java -Djdbc.drivers="com.sourcefire.vjdbc.VirtualDriver" program_name ...
```

where *program_name* is the name of your program.

Connecting the program to the database

Your program must obtain a JDBC connection object before it can submit queries. Use the `DriverManager.getConnection` method as follows to establish the connection and get the connection object:

```
Connection conn = DriverManager.getConnection("jdbc:vjdbc:rmi://my_dc:2000/VJdbc,eqe",
"user", "password");
```

where *my_dc* is either the FQDN or the IP address for the Firepower Management Center, `user` is the database access user account name, and *password* is the account password.

Querying the data in the Cisco tables

Create an SQL query object to submit the query and assign the retrieved records to a result set, as follows:

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("sql");
```

where *sql* is the SQL query. See Querying the Database, page 2-9 for supported SQL functions.

Producing the results of a table query

With the result set (`rs`) generated by the above query, you can output the fields as follows:

```
while(rs.next())
{
    for(int i=1; i<= md.getColumnCount(); i++)
    {
        System.out.print(rs.getString(i) + " ");
    }
    System.out.print("\n");
}
```

Getting schema information

Your program can list the tables in the database, as follows:

```
DatabaseMetaData metaData = conn.getMetaData();
      ResultSet tables = meta.getTables(null, null, null, null);
      while (tables.next())
      {
System.out.println(tables.getString("TABLE_NAME"));
      }
```

Your program can list a table's columns, as follows:

```
ResultSet columns = metaData.getColumns(null, null, "table_name", null);
```

where *table_name* is the name of the database table.

Running the Application

Before you run your application, you must set the CLASSPATH on the client computer to include the current directory and the locations of your application's JAR files.

If you downloaded and unpacked the ZIP package for Database Access as noted in Downloading the JDBC Driver, page 2-3, update the CLASSPATH as follows:

To run the application in a Unix environment:

Step 1 Use the following command:

```
export CLASSPATH=$CLASSPATH:.:path/lib/vjdbc.jar:path/lib/commons-logging-1.1.jar
```

where *path* is the directory path where you unpacked the ZIP package downloaded from the Firepower Management Center.

To run the application in a Windows 7 environment:

Step 1 Right-click the Computer icon and select **Properties**.

The **System** window appears.

Step 2 Click **Advanced System Settings**.

The **System Properties** window appears.

Step 3 Select the **Advanced** tab.

Step 4 Click **Environment Variables...**.

The **Environment Variables** window appears.

Step 5 Select the **CLASSPATH** system variable and click **Edit...**.

The **Edit System Variable** window appears.

Step 6 Add the following to the **Variable value:** field:

```
.;path\bin;.;path\lib\vjdbc.jar;.;path\lib\commons-logging-1.1.jar;.;path\lib
```

where *path* is the directory path where you unpacked the ZIP package downloaded from the Firepower Management Center.

Step 7 Click **OK** to save the value.

The Environment Variables window appears.

Step 8 Click **OK** to save the value. You can now run the application.

Querying the Database

External Database Access is a Global privilege. Unless constrained in the query, results will be for all domains in the database.

The following sections contain information on supported query syntax and other important query-related requirements and limitations:

- Supported SHOW Statement Syntax, page 2-9 describes the supported MySQL **SHOW** statement syntax for querying the Firepower System database.
- Supported DESCRIBE or DESC Statement Syntax, page 2-9 describes the supported MySQL **DESCRIBE** statement syntax for querying the Firepower System database.
- Supported SELECT Statement Syntax, page 2-10 describes the supported MySQL **SELECT** statement syntax for querying the Firepower System database.
- Join Constraints, page 2-12 describes the joins supported for querying the Firepower System database and explains how to get information on the specific allowed joins for any table.
- Querying Data Stored in Unfamiliar Formats, page 2-12 describes how to perform queries on data stored in formats that may be unfamiliar (including UNIX timestamps and IP addresses) so your queries are successful and your results appear as expected.
- Limiting Queries for Performance Reasons, page 2-14 contains recommendations on constraining your queries so as not to degrade the performance of the Firepower System.
- Query Tips, page 2-14 contains tips for querying intrusion events across several appliances.

For schema information and allowed joins, see the following chapters:

- Schema: System-Level Tables, page 3-1
- Schema: Intrusion Tables, page 4-1
- Schema: Statistics Tracking Tables, page 5-1
- Schema: Discovery Event and Network Map Tables, page 6-1
- Schema: Connection Log Tables, page 7-1
- Schema: User Activity Tables, page 8-1
- Schema: Correlation Tables, page 9-1

Supported SHOW Statement Syntax

The **SHOW** statement lists all tables in the Firepower System database. The following represents the supported MySQL **SHOW** statement syntax you can use when querying the Firepower System database:

```
SHOW TABLES;
```

Any **SHOW** statement syntax not listed above is not supported.

Supported DESCRIBE or DESC Statement Syntax

The Firepower System database offers limited use of the **DESCRIBE** statement. In the Firepower System database, the output of the DESCRIBE statement only lists the names of the columns and the type of data in each column. The following represents the supported MySQL **DESCRIBE** statement syntax you can use when querying the Firepower System database:

```
DESCRIBE table_name;
```

The Firepower System database also supports the identical command DESC:

```
DESC table_name;
```

Table 2-2 Supported DESCRIBE Statement Syntax

Where...	Is...
table_name	the name of a table you are querying

Any **DESCRIBE** statement syntax not listed above is not supported. In particular, the Firepower System database access feature does not support:

- the INDEX FOR clause
- the TABLE clause
- the PROCEDURE clause

Supported SELECT Statement Syntax

The following represents the supported MySQL SELECT statement syntax you can use when querying the Firepower System database:

```
SELECT
[ALL | DISTINCT]
[COUNT ( field ) | COUNT (*) ]

select_expr [, select_expr ...]
FROM table_references
[WHERE where_condition]
[GROUP BY { column_name | position } [ ASC | DESC ], …]
[HAVING where_condition]
[ORDER BY { column_name | position } [ ASC | DESC ], …]
[LIMIT { [offset,] row_count | row_count OFFSET offset}]
```

The following table details the required syntax for the clauses and arguments in the above SELECT statement.

Table 2-3 Supported SELECT Statement Syntax

Where...	Is...
select_expr	{*column_name* [[AS] *alias*] | *function*(...) [[AS] *alias*] | *aggregate_function*(...) [[AS] *alias*]}
column_name	the name of a field you are querying

Table 2-3 Supported SELECT Statement Syntax (continued)

Where...	Is...
function	{ABS | CAST | CEILING | CHAR_LENGTH | COALESCE | CONV | CHARACTER_LENGTH | CONCAT | CONVERT | COUNT | CURRENT_DATE | CURRENT_TIME | CURRENT_TIMESTAMP | EXTRACT | FLOOR | HEX | INET_ATON | INET_NTOA | INET6_ATON | INET6_NTOA | LEFT | LOWER | LPAD | MID | MOD | NULLIF | OCTET_LENGTH | POSITION | RIGHT | ROUND | SUBSTRING | SYSDATE | TIME | TIMESTAMP | TRIM | UPPER}
aggregate_function	{AVG | COUNT | COUNT(DISTINCT) | MAX | MIN | SUM}
field	the name of the field on which you are performing a function
table_references	one of: • *table_reference* INNER JOIN *table_reference join_condition* • *table_reference* LEFT [OUTER] JOIN *table_reference join_condition*
table_reference	*table_name* [[AS] alias]
table_name	the name of a table you are querying
join_condition	ON *conditional_expr*
conditional_expr	an equality comparison between fields that are join-compatible; see Join Constraints, page 2-12 for more information
where_condition	one of: • IS NULL or IS NOT NULL • NOT, ! • BETWEEN ... AND ... • LIKE • =, !=, <>, >, >=, <, <=

If you are not familiar with how supported MySQL syntax is expressed, see the following table for tips.

Table 2-4 MySQL Syntax Format

These symbols...	That is...	Represent...
brackets	[]	an optional clause or argument
curly brackets	{}	a required clause or argument
pipe	|	a choice between clauses or arguments

Any SELECT statement syntax not listed above is not supported. In particular, the Firepower System database access feature does not support:

- SELECT *, that is, you must explicitly specify fields
- unions
- subqueries
- the WITH ROLLUP modifier to the GROUP BY clause
- the INTO clause
- the FOR UPDATE clause

Join Constraints

The joins you can perform on Firepower System database tables are limited, for performance and other practical reasons. Cisco does not allow you to perform joins where the result is not likely to be useful for event analysis.

You can perform only inner joins and left (outer) joins. Nested joins, cross joins, natural joins, right (outer) joins, full (outer) joins, and joins with the `USING` clause are **not** supported.

The schema documentation indicates the supported joins for each table. Joins not listed are not supported. For example, you cannot join the **`compliance_event`** and **`intrusion_event`** tables on IP address fields, even though both tables contain IP address information. In addition, joins on deprecated tables and deprecated fields are not listed.

Querying Data Stored in Unfamiliar Formats

The Firepower System database stores some data in formats that may not be display-friendly. The following sections detail how to perform queries on various fields so your queries are successful and your results appear as expected:

- IPv6 Addresses, page 2-12
- IPv4 Addresses, page 2-12
- MAC Addresses, page 2-12
- Packet Data, page 2-13
- UNIX Timestamps, page 2-13

IPv6 Addresses

The Firepower System database stores IPv6 addresses in binary format. For results in hex notation, use the **`HEX()`** function. To query the database on a specific IPv6 address, use the **`UNHEX()`** function.

For example, the following statement queries the **`connection_log`** table, which contains information on monitored sessions, constraining the query by a specific IPv6 address:

```
SELECT HEX(initiator_ip), HEX(responder_ip), packets_sent, bytes_sent

FROM connection_log

WHERE initiator_ip = UNHEX('20010db80000000000000000000004321');
```

IPv4 Addresses

The Firepower System database stores IPv4 addresses in binary format within the same fields as IPv6 addresses. As with IPv6 addresses, use the `HEX()` function for hex notation. The database follows the RFC by filling in bits 80-95 with 1s, which yields an invalid IPv6 address. For example, the IPv4 address 10.5.15.1 would be stored as `00000000000000000000FFFF0A050F01`.

MAC Addresses

The Firepower System database stores MAC addresses in binary format. For results in hex notation, use the `HEX()` function.

For example. the following statement queries the **rna_host_mac_map** table, which contains information on hosts with MAC addresses that are not identified with an IP address, limiting the query to the first five hosts:

```
SELECT HEX(host_id), HEX(mac_address)
FROM rna_host_mac_map
LIMIT 5;
```

Packet Data

The Firepower System database stores packet data for intrusion events in binary format. For results in hex notation, use the `HEX()` function.

For example. the following statement queries the **intrusion_event_packet** table to obtain packet data for a particular event:

```
SELECT HEX(packet_data)
FROM intrusion_event_packet
WHERE event_id = 1234;
```

UNIX Timestamps

The Firepower System database stores most timestamps as UNIX timestamps, which represent the number of seconds elapsed since 00:00:00 January 1st, 1970 (UTC). For results in your local time, use the **FROM_UNIXTIME**`()` function.

For example, the following statement queries the **audit_log** table, which keeps a record of all user actions on the web interface of an appliance, and returns up to 25 results:

```
SELECT FROM_UNIXTIME(action_time_sec), user, message
FROM audit_log
LIMIT 0, 25;
```

Keep in mind that all times in the database are in UTC. Although the **CONVERT_TZ()** function is allowed, it only provides results in UTC.

Note that some events have microsecond resolution associated with them. Use the **CONCAT()** and **LPAD()** functions to concatenate the UNIX timestamp and the microsecond increment. For example, the following statement queries the **intrusion_event** table:

```
SELECT CONCAT(FROM_UNIXTIME(event_time_sec)), '.', LPAD (event_time_usec, 6, '0')),
HEX(host_id),
rule_message
FROM intrusion_event
LIMIT 0, 25;
```

To query the database for events with a particular UNIX timestamp, use the **UNIX_TIMESTAMP()** function.

Limiting Queries for Performance Reasons

Although the system limits the joins you can perform on Firepower System database tables, it does still allow some expensive queries - queries that may negatively impact the performance of your Firepower Management Center.

Therefore, you should try to limit the result set for large tables. Strategies include:

- constraining queries to a specific leaf domain
- constraining queries to a specific time range
- constraining queries by IP address
- using the **LIMIT** clause

Depending on your deployment, querying many tables may require a limited result set. In particular, the following tables can contain up to 100 million events on a DC3000:

- **fireamp_event**
- **intrusion_event**
- **intrusion_event_packet**
- **connection_log** (pre-Version 5.0 name: **rna_flow**)
- **connection_summary** (pre-Version 5.0 name: **rna_flow_summary**)

Queries on network map tables may also be expensive, depending on the number of hosts the system has detected on your monitored network.

Query Tips

The following sections provide tips on ensuring unique results when you build queries that include detection engines or intrusion events.

Device Names

Device names are not necessarily unique across multiple Firepower Management Centers. To ensure uniqueness, include a specific device UUID in your query.

Intrusion Events

To uniquely match an intrusion event across multiple managed devices, include the following fields in your query of the **intrusion_event** table:

- `intrusion_event.event_id`
- `intrusion_event.event_time_sec`
- `intrusion_event.sensor_uuid`

Troubleshooting Queries

You can configure multiple Firepower Management Centers to allow access to a a single client, but each system must be configured individually. The available information from each system depend on multiple factors. If there is no data to query, your queries will not return expected results.

The following list outlines some of the specific reasons why a query may not return results:

- Your query is too specific. For example, you may need to adjust the time range or IP address range of a query.
- Not all of the fields for an event may be populated, depending on the network traffic that caused an event to be generated. For example, not all connection events contain payload information.
- You have not configured logging for the event type you are querying.
- You have disabled event storage.
- you are attempting to query on a domain to which you do not have access.

For more information on how events are generated and logged, see the *Firepower Management Center Configuration Guide*.

Sample Queries

The following sections contain sample queries that illustrate how you can use the database access feature:

Caution Performing some of these sample queries may be expensive, depending on your deployment. See Limiting Queries for Performance Reasons, page 2-14 for more information.

Audit Records for a User

The following query returns all records in the audit log for a particular user, displaying all timestamps in UTC:

```
SELECT FROM_UNIXTIME(action_time_sec), user, message

FROM audit_log

WHERE user = 'eventanalyst';
```

Intrusion Events by Priority and Classification

The following query duplicates the Drilldown of Event, Priority, and Classification view in the Events By Priority and Classification workflow. If you have not changed the default Intrusion Events workflow in your user preferences, this is the first page you see when you select **Analysis > Intrusion Events** on the Firepower Management Center web interface:

```
SELECT rule_message, priority, rule_classification, count(*) as Count
FROM intrusion_event
WHERE reviewed="0" GROUP BY rule_message, priority, rule_classification
ORDER BY Count
DESCLIMIT 0, 25;
```

Intrusion Events and Their Associated Policies

The following query lists intrusion events from the specified week. For each event it shows the associated intrusion policy violation and rule classification.

```
SELECT FROM_UNIXTIME(event_time_sec) AS event_time, event_id AS intrusion_event,
intrusion_event_policy_name AS policy, rule_classification AS classification
FROM intrusion_event
WHERE event_time_sec BETWEEN UNIX_TIMESTAMP('2011-10-01 00:00:00') AND
UNIX_TIMESTAMP('2011-10-07 23:59:59')
ORDER BY policy ASC;
```

Lists of Detected Hosts

The following query returns the basic information in the hosts network map for all MAC hosts (hosts without an IP address) detected on your network, along with the hardware vendor for each NIC:

```
SELECT HEX(mac_address), mac_vendor, host_type, FROM_UNIXTIME(last_seen_sec)
FROM rna_mac_host;
```

The following query maps IP addresses to MAC addresses:

```
SELECT HEX(ipaddr), HEX(mac_address), HEX(host_id)
FROM rna_host_ip_map LEFT JOIN rna_host_mac_map on
rna_host_ip_map.host_id=rna_host_mac_map.host_id;
```

List of Detected Servers

The following query joins two related tables to give you a list of the servers detected on your network along with many of their attributes, similar to what you can see in a table view of servers on the Firepower Management Center's web interface:

```
SELECT FROM_UNIXTIME(s.last_used_sec), HEX(s.host_id), s.port, s.protocol, s.hits,
i.service_name, i.vendor, i.version, i.source_type, s.confidence
FROM AS s
LEFT JOIN rna_ip_host_service_info AS i ON (s.host_id = i.host_id AND s.port = i.port AND
s.protocol =
i.protocol);
```

Note that this query left joins the tables on the set of `host_id`, `port`, and `protocol`, as required by Database Access. See rna_host_service Joins, page 6-35 and rna_host_service_info Joins, page 6-40.

Server Vulnerabilities on Your Network

The following query joins two vulnerability-related tables to give you a list of valid server-related vulnerabilities detected for a particular host, along with whether each vulnerability is exploitable across a network:

```
SELECT h.rna_vuln_id, v.title, v.remote

FROM rna_host_service_vulns AS h

LEFT JOIN rna_vuln AS v ON (h.rna_vuln_id = v.rna_vuln_id)

WHERE h.ip_address = INET_ATON('10.10.10.4')

AND h.invalid = 0;
```

Note that this query left joins the tables on **rna_vuln_id**, as required by rna_host_service_vulns, page 6-47 and rna_vuln Joins, page 6-57.

Operating System Summary

The following query duplicates the Summary of OS Names page in the Operating System Summary workflow. If you have not changed the default workflow in your user preferences, this is the first page you see when you select **Analysis > Hosts** on the Firepower Management Center web interface, then select **Hosts**:

```
SELECT vendor, product, count(*) AS total

FROM rna_host_os

GROUP BY vendor, product

ORDER BY total DESC;
```

Operating System Vulnerabilities for a Host

The following query joins two vulnerability-related tables to give you a list of valid operating system-related vulnerabilities detected for a particular host, along with whether each vulnerability is exploitable across a network:

```
SELECT h.rna_vuln_id, v.title, v.remote

FROM rna_host_os_vulns AS h

LEFT JOIN rna_vuln AS v ON (h.rna_vuln_id = v.rna_vuln_id)

WHERE h.host_id = UNHEX('9610B6E6F1784DA4B39BEA7A210AAD68')

AND h.invalid = 0;
```

Note that this query left joins the tables on **rna_vuln_id**, as required by Database Access. See rna_host_os_vulns, page 6-30 and rna_vuln Joins, page 6-57.

Host Violation Count

The following query duplicates the Host Violation Count page in the Host Violation Count workflow. If you have not changed the default Compliance White List Violations workflow in your user preferences, this is the first page you see when you select **Analysis > Correlation > White List Violations** on the Firepower Management Center's web interface.

```
SELECT host_id, HEX(host_id), white_list_name, count(*) AS total
FROM white_list_violation
GROUP BY host_id, white_list_name
ORDER BY total DESC;
```