Chapter 31
FPVA Step 2, Resource Identification

Revision 4.0, March 2025.

Objectives

- Mastering the next step in learning to think like an analyst.
- Understanding the resources that a system accesses and evaluate the criticality of these resources.
- Learn how to construct a Resource Diagram and the components used in a Resource Diagram.
- Learn about the relationship between the Architecture Diagram and the Resource Diagram.
- Learn about tools and techniques you can use to gather information for constructing these diagrams.

31.1 Motivation

As we have discussed in the previous two chapters, the goal of FPVA is to focus the analyst's attention on the vulnerabilities affecting the highest value assets. After identifying the main components in the system and how they interact, we are now ready to start the Resource Identification step in which we identify the key resources used by the system.

A key resource is any object controlled by the system that is useful to a user of the system. Examples of data resources include files, databases, and data in memory. Examples of physical resources include disk space, CPU cycles, network bandwidth, and any attached devices such as sensors or physical devices being controlled by the system (such as is used in a cyber-physical system).

Remember that the impact surface describes the parts of the code where a resources are accessed as a consequence of an attack. The attack vector follows the path from the attack surface to the impact surface. When we add resource information to an Architecture Diagram, we get a picture that can help the analyst understand what resources are most at risk. The goal of this step is to document those resources.

For each resource, we will show which process or thread controls it, and on what host that resource is located. And separate from the diagram, as it will not fit in the diagram, you need to document which operations are allowed on each resource, and the consequences of an attacker gaining access to that resource. As you can imagine, later in the FPVA methodology, we will focus our search for vulnerabilities affecting the high value assets. High value assets are the resources that, if exploited by an attacker, would cause great harm. A clear example of such harm would be an attacker modifying an improperly protected password file to gain root access.

The key product from the Resource Identification step is the Resource Diagram. Though we use the term Resource Diagram, more properly it is an Architecture Diagram *annotated* with the resources in the system.

In this chapter, we will use Resource Diagrams from two real FPVA engagements as our running examples. In **Error! Reference source not found.**, we see a diagram from a system that manages and monitors computing resources. Note that this is a diagram from the same system that we discussed for the architectural analysis in Chapter 30. In Figure *2*, we see a diagram from a system that schedules resources in a high throughput computing environment.

Resource identification is crucial. If an attacker can modify a resource of control which resource is being accessed, then they might be able to change what programs run, under what user ID they run, what authentication credentials are used, and what permissions are used to control access to the system and its resources.

31.2 Key Resource: Files

Files are resources to which we obviously need to pay attention. Files are represented by a path name, and programs access open files though file descriptors or handles.

A path name might be a constant string in the program or based all or in part on user input. We saw from our discussion of the attack surface (Chapters 2 and 3) and from the discussion in FPVA Step 1 about interactions and communication (Chapter 30) that user input can arrive in a system through a wide variety of paths. If user input is used to control access to files, then an attacker can try to manipulate which files are read or written by the program, potentially causing a great deal of harm. This type of an attack, called a *path name traversal attack*, was discussed in Chapter 12.

You can identify the places in the code where files are opened by locating the kernel or library calls that open and create a file.

A program gets a file descriptor or file handle when creating a new file or opening an existing file. As above, you can identify the places in the code where this happens by locating the open and create calls. Also, in UNIX systems like Linux, FreeBSD or the POSIX subsystem on Windows, files descriptors can be inherited from the parent process. On these systems, you need to pay attention to calls such `fork` and `clone`.

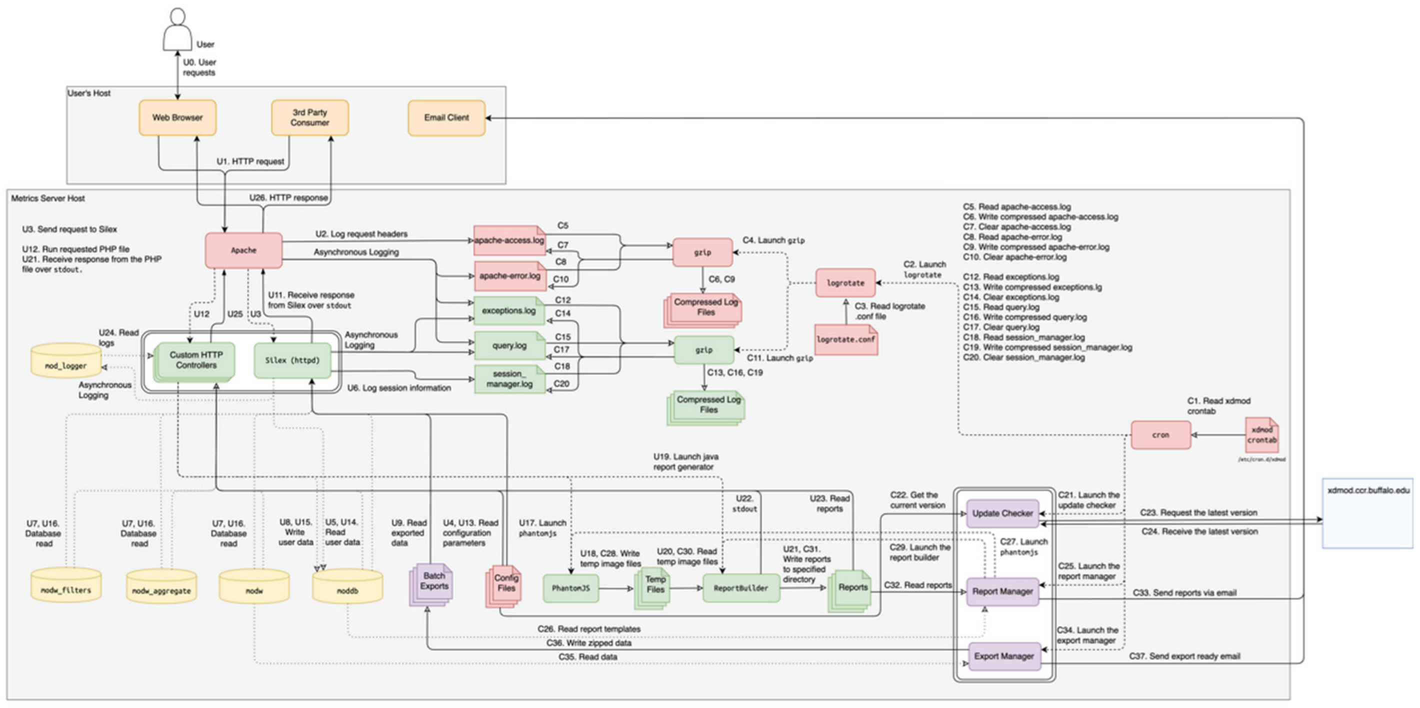

Figure 1: Example of Resource Diagram

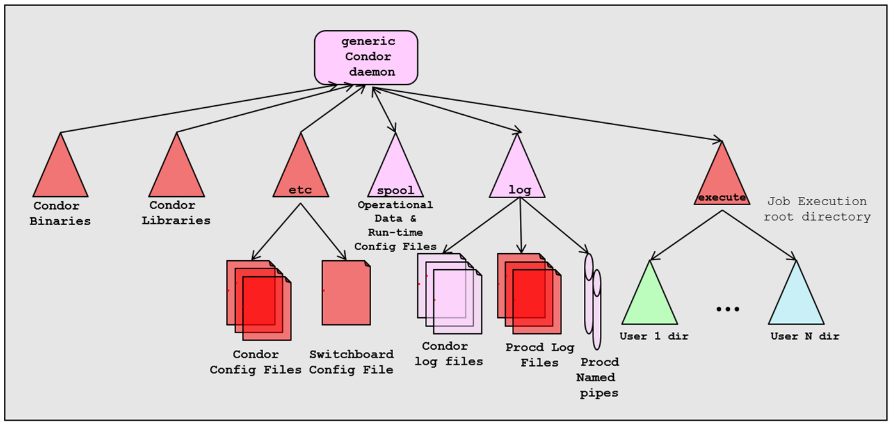

Figure 2: A Second Example Resource Diagram

On UNIX systems, you should pay special attention to file descriptors 0, 1, and 2, representing standard input, output and error respectively. These files are typically opened by the parent (creating) process and passed to the child (created) process.

Files can contain data to be processed by the system you are assessing, configuration information, or code. The latter is the case of executable code, libraries, or scripts. In addition, log files are an important type of file that is commonly written by a program.

Files that contain executable code are important because any change to them could change the future behavior of the software system. Configuration files are important to a security assessment because they contain values that determine how the system operates, including things like the names of other programs to run or file protection settings. As a result, an attacker might change which code gets run or which users are allowed to perform critical task. An attacker gaining access to a log file could perform an attack that creates a denial of service by filling up a disk or overwriting important log data in the case of a circular buffer. Overwriting log data can be used to hide activity associated with the attack.

In both of our example diagrams (Figure *1* and Figure *2*), we can see the files for our example systems. In these diagrams, we find configuration files and log files, which are common file types likely to be found on almost every system. Both configuration and log files are often targets of an exploit. Take a moment to locate these files in the diagrams.

In Figure 2, we see that this system used *named pipes* for some communication between processes on the same host. Named pipes can be thought of as features of the Architecture Diagram, but their names appear in the file system, so we include them here.

31.3 Key Resource: Directories

Directories are closely related to files. Directories contain named file system objects such as files, directories, communication channels (such as named pipes on UNIX), devices, and even processes (such as in the `/proc` file system on UNIX). Directory entries contain metadata, like file size, permissions, owner, and time of creation, last access, and last modification. Operations related to directories include creating, removing, and renaming directories; creating, removing, and renaming files; listing its content; and updating metadata such as file permissions or owner.

In Figure *2*, we see several examples of directories. They are drawn as triangles in an FPVA diagram. Take a moment to identify and name them.

On UNIX systems, some common directory system calls are:

`opendir readdir closedir creat open` (with `O_CREAT`) `fdopen mkdir mkfifo mknod symlink link unlink remove rename rmdir`

On Windows, some common directory system and library calls are:

`CreateDirectory CreateSymbolicLink RemoveDirectory Delete Exists GetDirectories Move`

31.4 Key Resource: Symbolic Links

A symbolic link is a file system object that points to another file, similar in concept to a desktop shortcut, but implemented in the file system. A symbolic link contains a path name that points to another entry in the file system, called the referent.

File system links are important because they create *aliases* for a file. Having an alias means that there are two completely different file names that point to the same file system object, and these two names could have different file permissions. Having multiple file system names for the same file system object means that there are multiple paths to attack that object. And if the permissions on the link itself are not set correctly, an attacker might be able to change the file that the system is accessing.

31.5 Key Resource: Databases

Databases are another important class of program resources, used for a wide variety of purpose, often containing sensitive information. Such information can include user authentication information such as user names and passwords, personal identifying information (PII), financial information, logging information, and scientific data. Exploits against a database can have serious consequences.

Our example system in Figure *1* uses a variety of databases and are key resources in this system. Two databases were of particular concern in this assessment. The `moddb` database contains user information, session keys and some configuration data, making it an attractive target for an attacker. In addition, the `mod_logger` database could be a target of a denial-of-service attack.

31.6 Closing Thoughts

Now, our architecture diagram is decorated with the resources used in the system. We note that our Architecture and Resource Diagrams also include coloring for the elements. In the next chapter on Trust and Privilege Analysis, we will see how we determine those colors and what they mean.

31.7 Summary

In this chapter, we:

- Understood how to find the resources that a system accesses and evaluate the criticality of these resources.
- Learned how to construct a Resource Diagram and the components used in a Resource Diagram.
- Learned about the relationship between the Architecture Diagram and the Resource Diagram.
- Learned about tools and techniques you can use to gather information for constructing these diagrams.

31.8 Exercises

1. What is the relationship between an Architecture Diagram and Resource Diagram?
2. Figure *1* and Figure *2* are examples of Resource Diagrams taken from FPVA assessments of real systems. Look carefully at each of these diagrams identifying the:
    a. Configuration files accessed
    b. Log files
    c. Databases
    d. Directories
3. For each of the four element in the previous question, explain why they are important to include in a Resource Diagram.
4. For a system with which you are familiar, follow the steps in the chapter to construct a Resource Diagram for that system. You can start

with the Architecture Diagram that you produced for the exercise in the previous chapter.

5. If you are familiar with Microsoft Threat Modeling (Chapter 5), compare how resources are illustrated in a threat modeling diagram compared with how they are illustrated in an FPVA Resource Diagram.