**Unreleased Resource: Sockets Development Mitigation SOP**

Unreleased resource vulnerabilities can occur from sockets failing to be released. Resource leaks are commonly caused by confusion over which part of the program is responsible for releasing the resource or error conditions and other exception circumstances. Most of these issues can result in general software reliability problems. If an attacker can intentionally trigger a resource leak, they may be able to launch a denial of service attack by depleting the resource pool.

**Defense Against Unreleased Resource: Sockets**

Sockets should be released in a finally block.

**Example**

private void echoSocket(String host, int port) throws UnknownHostException, SocketException, IOException {

Socket sock = new Socket(host, port);

BufferedReader reader = new BufferedReader(new

InputStreamReader(sock.getInputStream()));

while((String socketData = reader.readLine()) != null) {

System.out.println(socketData);

}

}

**Explanation**

The code above never closes the socket it opened, which can result in the JVM using up all of its sockets.

**Recommendation**

The code below rewrites the example from above with the socket released in the finally block (bolded below). This code also uses a helper function to log any exceptions that might occur when attempting to close the socket. Lastly, there is a check in place to ensure *sock* is not *null* before attempting to close. This is important because without it, the Java compiler may report that *sock* might not be initialized. This check avoids the detection by the compiler.

private void echoSocket(String host, int port) throws UnknownHostException, SocketException, IOException {

Socket sock;

BufferedReader reader;

try {

sock = new Socket(host, port);

reader = new BufferedReader(newInputStreamReader(

sock.getInputStream()));

while((String socketData = reader.readLine()) != null)

{

System.out.println(socketData);

}

} finally {

**safeClose(sock);**

}

}

public static void safeClose(Socket s) {

if(s != null && !s.isClosed()) {

try {

s.close();

} catch(IOException e) {

log(e);

}

}

}

**References**

1. [HP Enterprise Security – Unreleased Resource: Sockets](http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/unreleased_resource_sockets.html)