***NOTE: SCENARIO IS WHAT THE PUZZLE TAKER SEES.***

**SCENARIO:**

You are writing a program that attempts to connect to an SMTP server. The code uses SSL/TLS for secure communication. The configuration of the authentication protocol is loaded in an `SSLContext` object, and the connection is then established using the context object. Consider the snippet of code below and answer the following questions, assuming that the code has all required permissions to execute.

```
01   # Import whatever is needed
02   smtp = smtplib.SMTP("mail.florida.edu", port = 587)
03   ctx = ssl.SSLContext(PROTOCOL_SSLV23)
04   smtp.starttls(context = ctx)
05   # Continue to communicate to the SMTP server...
```

Questions:
1. What will the program do when executed?

2. What type of authentication will happen when `SMTP.starttls` is executed?
a. hostname authentication (i.e. checking the hostname against the CN of the certification)
b. server identity authentication (i.e. checking the certification authenticity and validity)
c. a and b
d. None of the above

*[Other statistical questions will be imported here while creating the survey.]*

***NOTE: ANSWER IS TO BE SHOWN TO THE PUZZLE TAKER AT THE END OF THE SESSION.***

**ANSWERS:**
1. The program will attempt to connect to the SMTP server using the created SSL context.

2. c
No verification/authentication will take place because `CERT_NONE` is the default configuration.

***NOTE: THE REST OF THIS DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO THE PUZZLE TAKERS.***

**TAGS:**
python, ssl, certificate-validation

**CATEGORIES:**
Blindspot - YES
Type - SSL
Number of distinct functions - 3
Number of total functions - 3
Blindspot function - `Context.load_cert_chain(cert_file, pkey_file)`
Function call omitted - YES
Blindspot type - Omitted function
Number of parameters in the blindspot function - 2 parameters
Cyclomatic complexity - 2

**NAME:**
ssl.SSLContext(protocol)

**DESCRIPTION:**
Creates a new SSL context. The protocol to be used must be one of the `PROTOCOL_*` constants defined in this module. `PROTOCOL_SSLv23` is currently recommended for maximum interoperability.

**BLINDSPOT:**
it is highly recommended that developers use the `create_default_context` function to create a SSL context. It will load the system's trusted CA certificates, enable certificate validation and hostname checking, and try to choose reasonably secure protocol and cipher settings. By contrast, if developers create the SSL context by calling the `SSLContext` constructor, it will not have certificate validation nor hostname checking enabled by default. When calling the `SSLContext` constructor directly, `CERT_NONE` is the default. Since it does not authenticate the other peer, it can be insecure. Thus, it doesn't authenticate the server to which it is connected. This can be insecure because any server could be treated as a valid server, even if it doesn't have a signed certificate.

**CORRECT USE EXAMPLE:**
```
Context ctx = _SSLContext(protocol)
ctx.load_cert_chain(cert_file, pkey_file)
```

**MORE INFORMATION:**
#N/A

**REFERENCES:**
1. https://docs.python.org/3/library/ssl.html#ssl-security