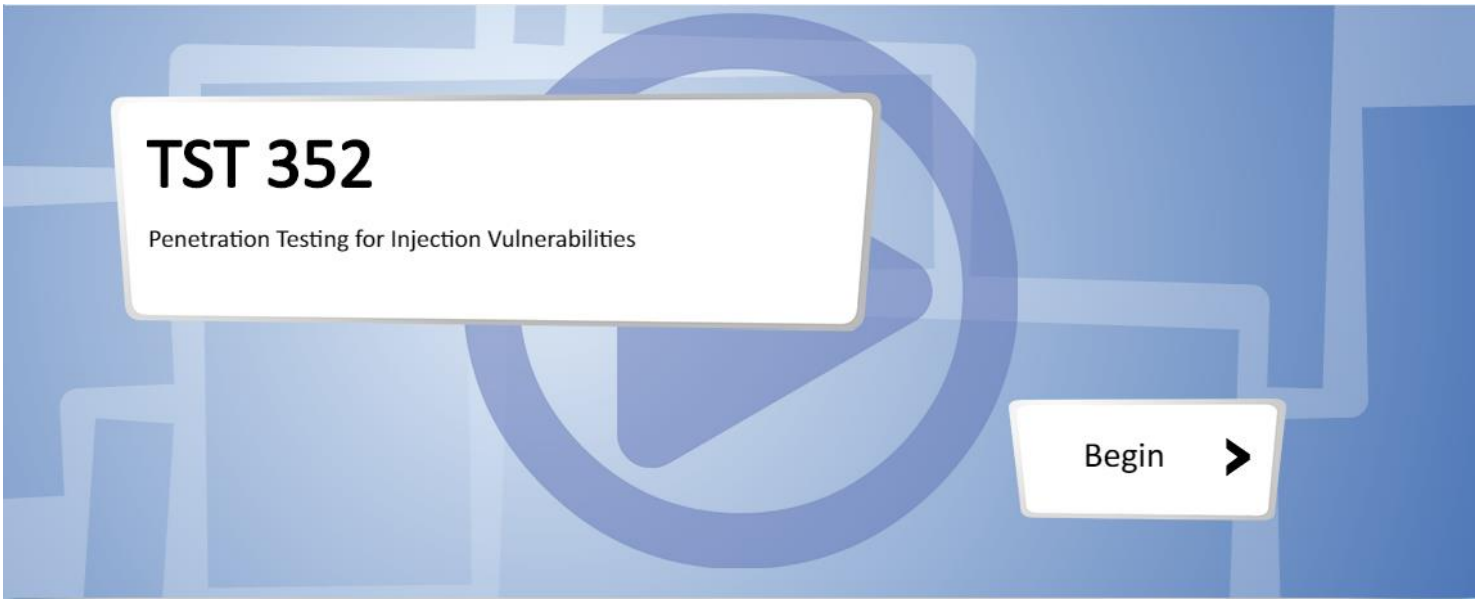
Table of Contents



Narration

On Screen Text

TST 352

Penetration Testing for Injection Vulnerabilities

Course Overview and Objectives

Narration

Injection is a broad category of vulnerabilities that stem from the ability to inject crafted data as input that will then manipulate the behavior of the application.

After completing this course, you will be able to identify common injection vulnerabilities and exploit them. Click the NIST button to see how this course aligns to industry standards.

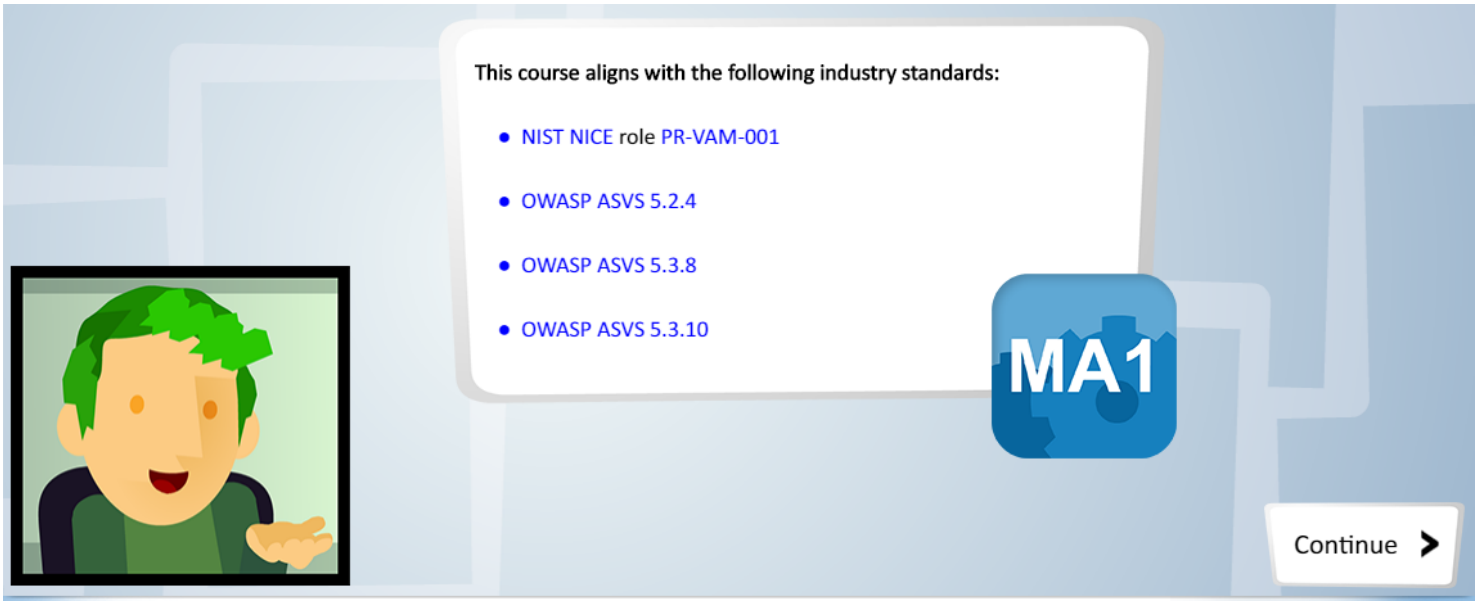
On Screen Text

Course Overview and Objectives

After completing this course, you will be able to:

- Test for common injection vulnerabilities
- Exploit common injection vulnerabilities

Alignment with Industry Standards



Narration

This course aligns with portions of the following industry standards: the National Initiative for Cybersecurity Education (NICE) framework produced by the National Institute of Standards and Technology (NIST), and the Application Security Verification Standard (ASVS) maintained by the Open Web Application Security Project (OWASP).

This course is aimed at the NIST NICE role PR-VAM-001, Vulnerability Analyst.

It aligns with the following OWASP ASVS requirements:

5.2.4 — Verify that the application avoids the use of eval() or other dynamic execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.

5.3.8 — Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding.

5.3.10 — Verify that the application protects against XPath injection or XML injection attacks.

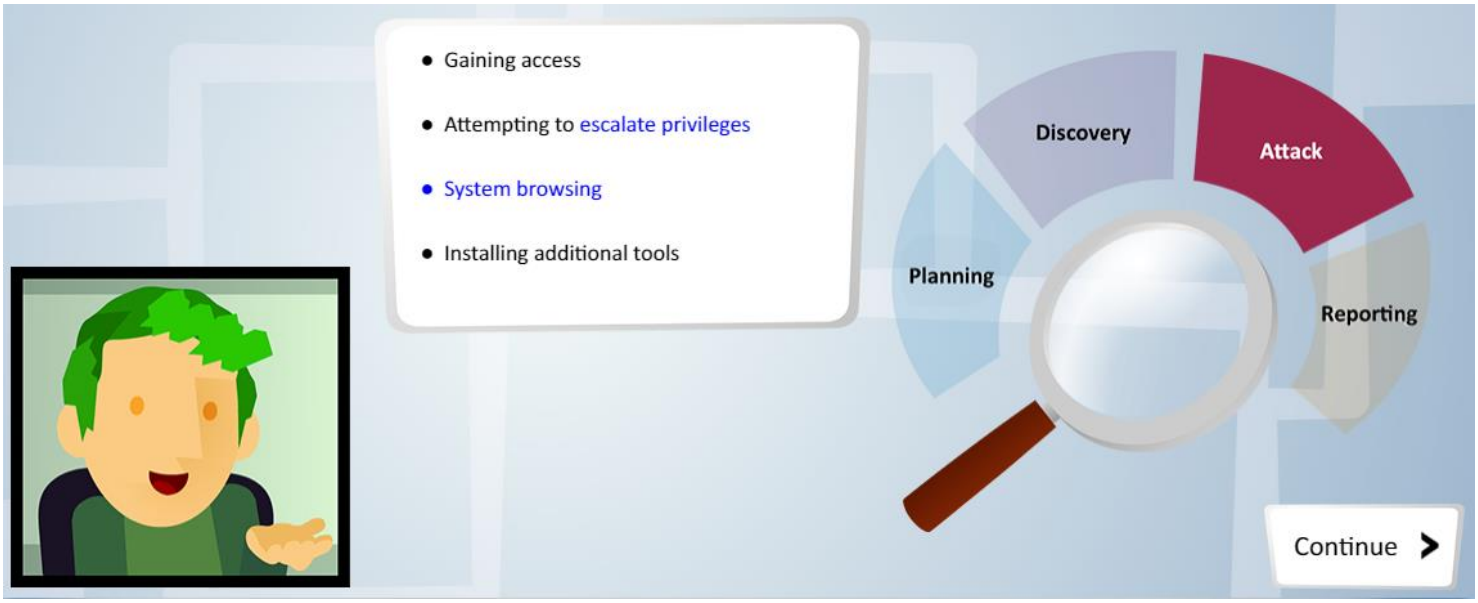
On Screen Text

Alignment with Industry Standards

This course aligns with the following industry standards:

- NIST NICE role PR-VAM-001
- OWASP ASVS 5.2.4
- OWASP ASVS 5.3.8
- OWASP ASVS 5.3.10

Penetration Testing for Injection Vulnerabilities



Narration

Generally, the penetration testing process has four main phases: planning, discovery, attack, and reporting.

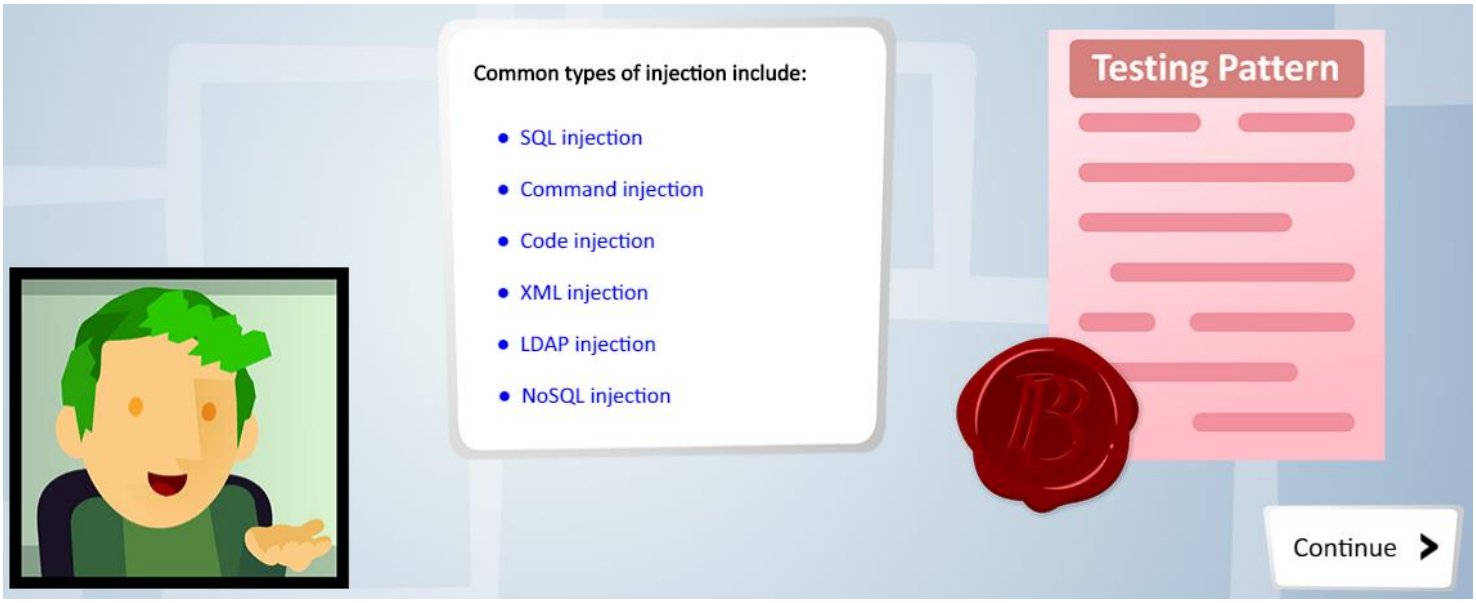
This course focuses on the attack phase, which itself typically includes steps of gaining access, attempting to escalate privileges, system browsing, and installing additional tools.
In this course, we describe a variety of injection vulnerabilities and provide an interaction to illustrate the typical testing process.

On Screen Text

Penetration Testing for Injection Vulnerabilities

- Gaining access
- Attempting to escalate privileges
- System browsing
- Installing additional tools

Injection Overview



Narration

Injection is a broad category of vulnerabilities that stem from the ability to inject crafted data as input that will then manipulate the behavior of the application.

Injection vulnerabilities are easy to detect and exploit. At the same time, injection vulnerabilities often have very severe impact, making injection vulnerabilities dangerous.

The types of injection vulnerabilities are defined by the contexts in which they appear.

For example, an injection vulnerability in the code that executes SQL queries is an SQL Injection vulnerability and an injection vulnerability in the code that executes external commands is called a Command Injection vulnerability.

The exact impact of injection vulnerabilities also depends on the context.

For example, vulnerabilities in database access code will often allow attackers to access arbitrary data in the database, while vulnerabilities in code that executes commands will allow attackers to execute arbitrary commands.

The most common types of injection include: SQL Injection, Command Injection, Code Injection, XML Injection, LDAP Injection, NoSQL Injection, etc.

While actions to exploit the different types of injection vulnerabilities are different, testing for the presence of injection vulnerabilities follows a general pattern.

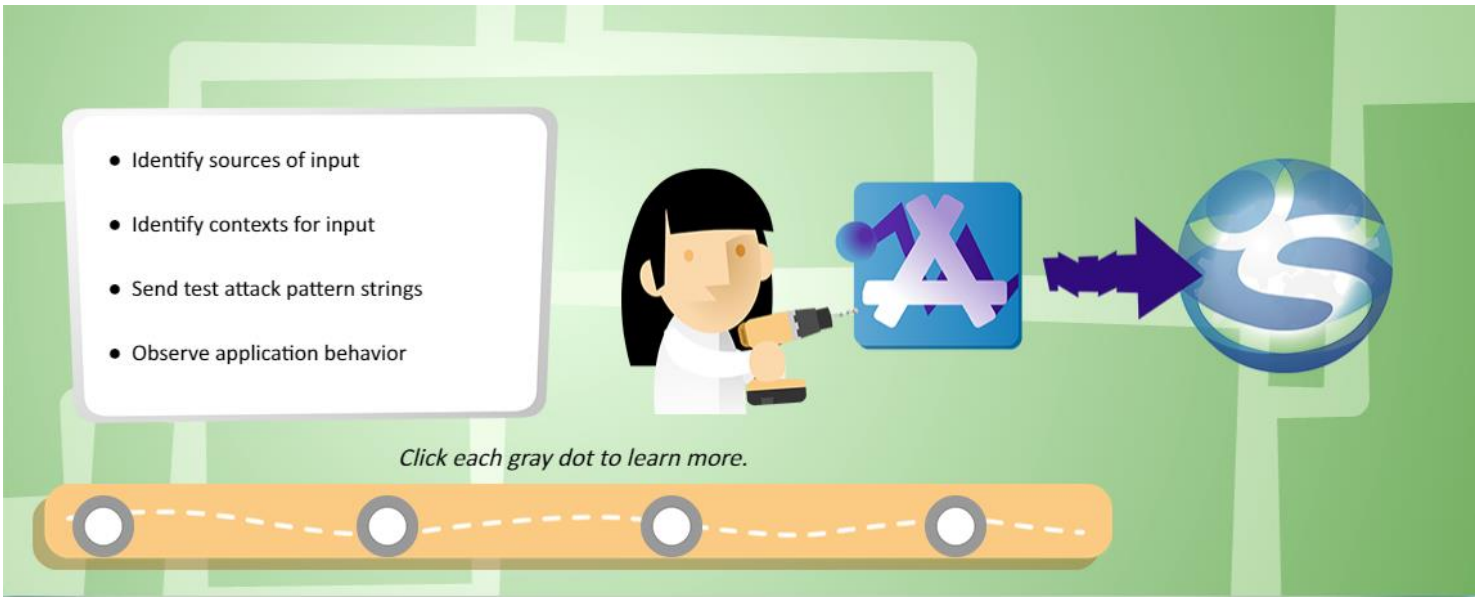
On Screen Text

Injection Overview

Common types of injection include:

- SQL injection
- Command injection
- Code injection
- XML injection
- LDAP injection
- NoSQL injection

Injection Testing Process Overview



Narration

The general process to test an application for injection vulnerabilities is as follows:

First, identify sources of input to the application.
Next, identify contexts for input.
Then, send test attack pattern strings as input.
Finally, observe application behavior when processing test strings.

Click each gray dot to learn more.

On Screen Text

Injection Testing Process Overview

- Identify sources of input
- Identify contexts for input
- Send test attack pattern strings
- Observe application behavior

Click each gray dot to learn more.

Identify Sources of Input

Narration

The first step in penetration testing for injection vulnerabilities is to identify sources of input to the application.

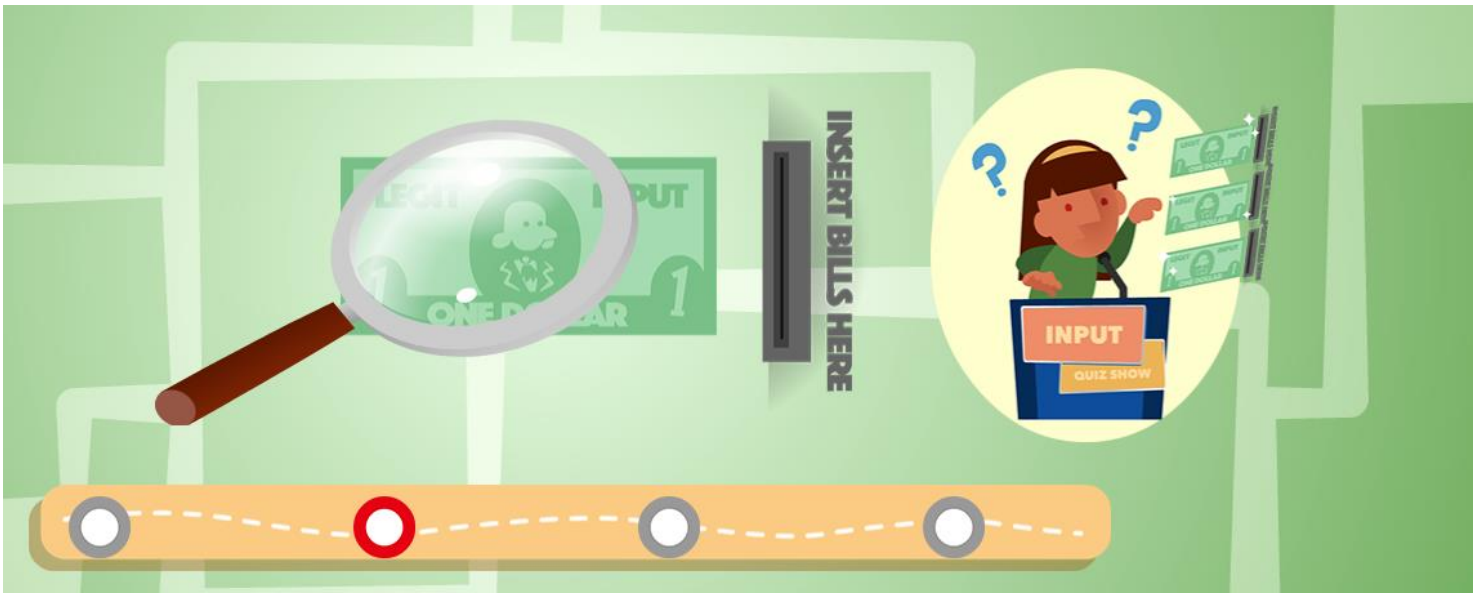
Generally speaking, we are only interested in text-based sources of input to the application. Input that cannot be interpreted as plain text, such as clicking on GUI elements, is not relevant to testing for injection.

When testing, priority should be given to inputs that are known to be processed in contexts that are susceptible to injection, such as input being stored in SQL databases, used to form LDAP queries, or used as arguments when executing external applications.

On Screen Text

Identify Sources of Input

Identify Contexts for Input



Narration

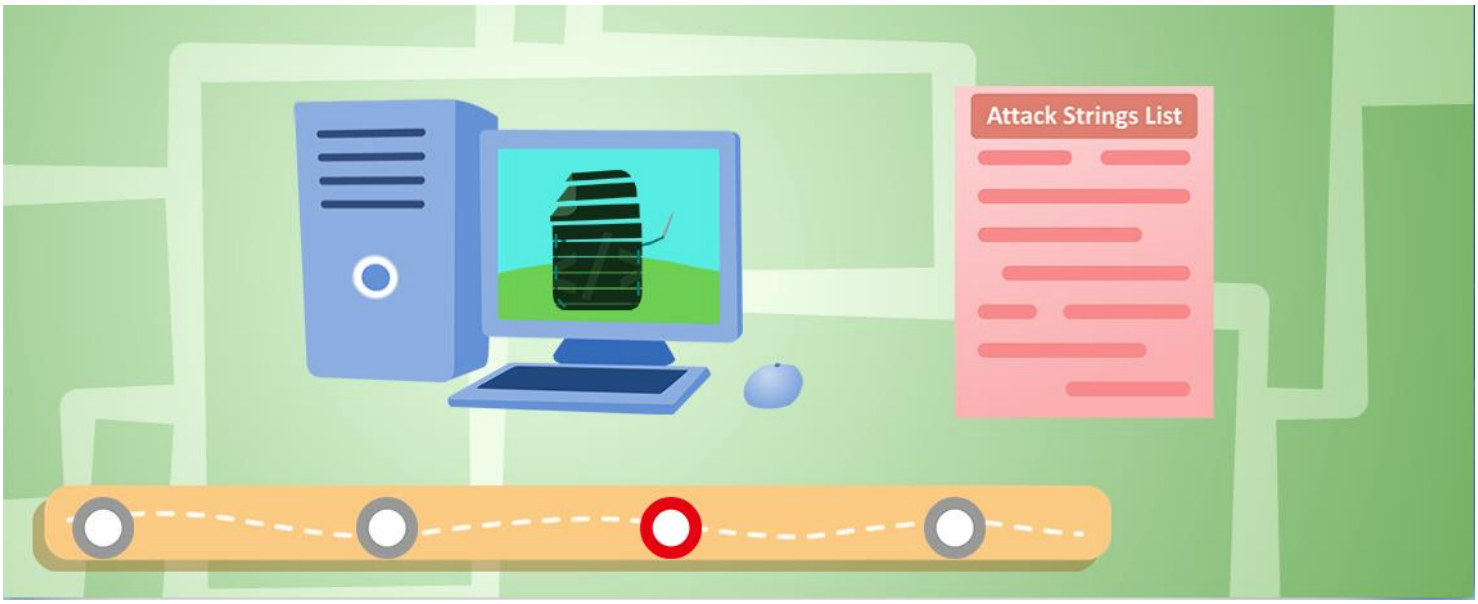
Once relevant sources of input have been identified, identify contexts for input. That is, for each source of input to the application, attempt to identify whether the input data is going to be processed in a context that is susceptible to injection, and what context that might be.

If you cannot positively identify the type of injection that might be triggered by a specific type of input, you can still test for injection by passing input that should trigger different types of injection that might be present.

On Screen Text

Identify Contexts for Input

Send Test Attack Pattern Strings



Narration

The next step is to send test attack pattern strings as input. Testing for different types of injection vulnerabilities requires using the correct test strings for the context.

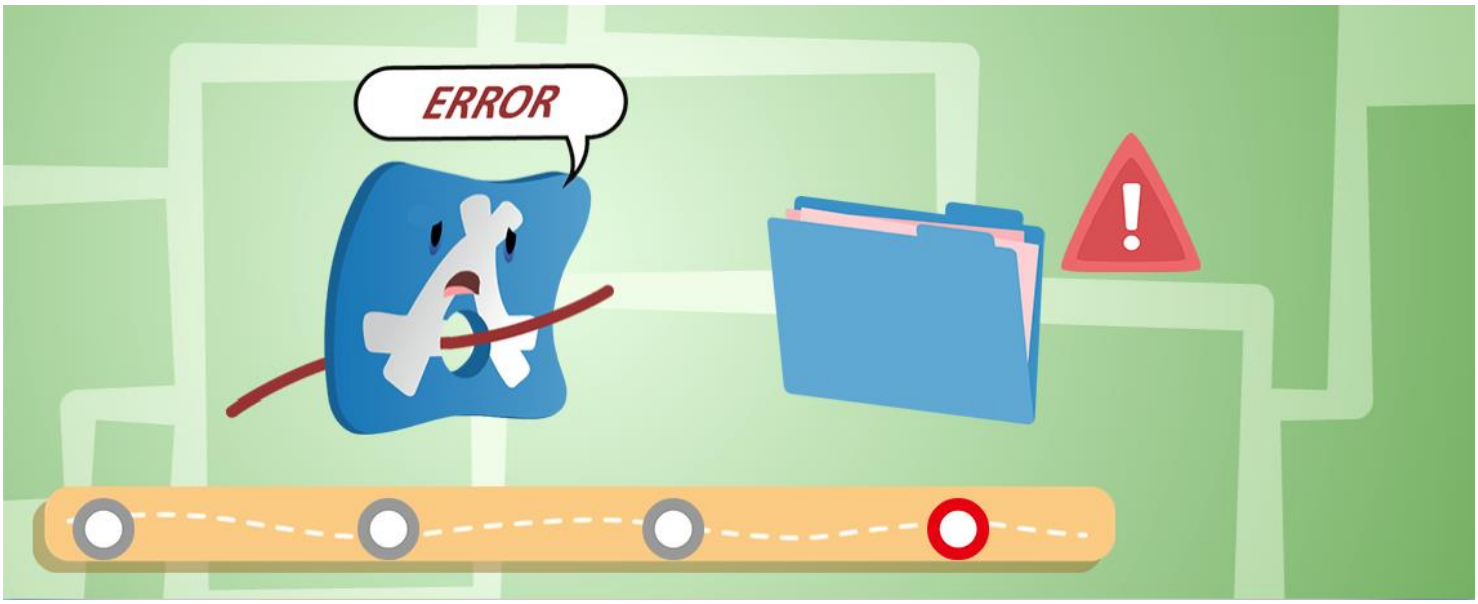
If you have identified the type of injection that is likely to be present, then you should send the test patterns that match that type of injection. Otherwise, you can send test strings for all the types of injection that might be present as input.

While collections of test strings for most common types of injection are readily available on the Internet, it is best if you understand the nature of the vulnerabilities and are able to customize the lists of attack strings for your purposes rather than simply reuse lists prepared by someone else.

On Screen Text

Send Test Attack Pattern Strings

Observe Application Behavior



Narration

Finally, observe application behavior when it processes test strings. If your test strings trigger a vulnerability, the application will usually enter an error state and will often, but not always, display an error message.

In some cases, however, the application will appear to function correctly even though a vulnerability is present.

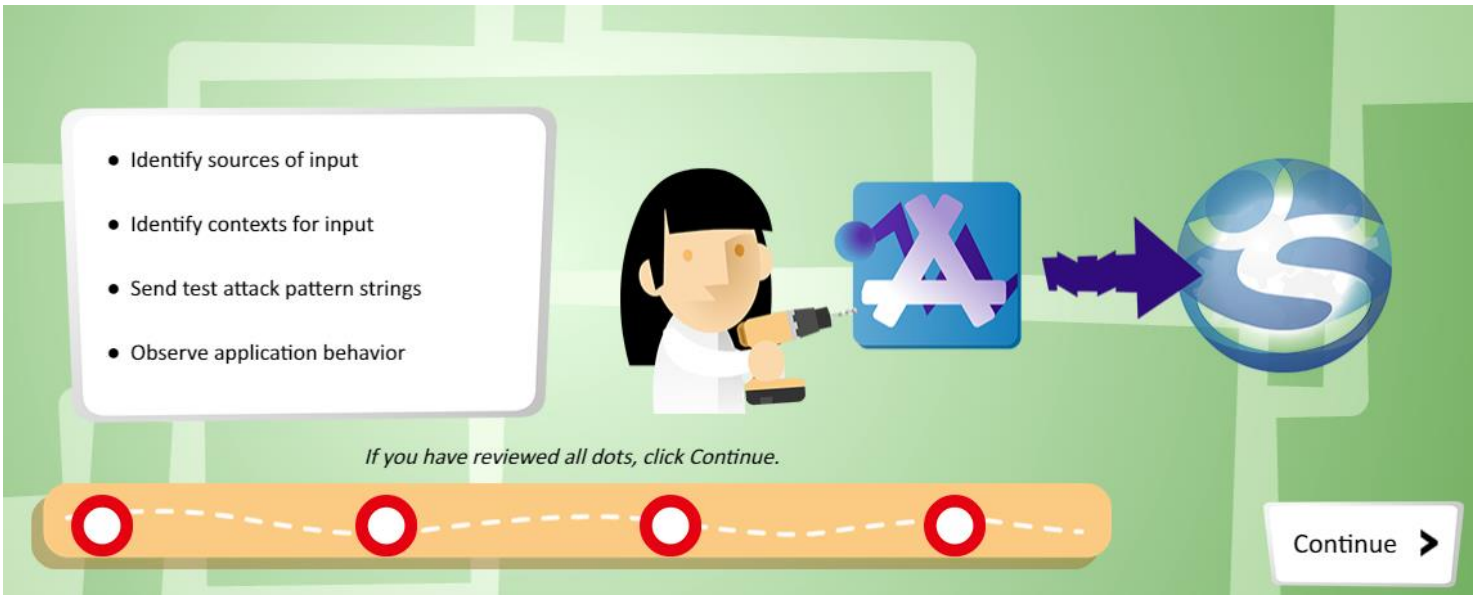
You can still discover vulnerabilities that do not cause obvious changes in application behavior by passing test strings that perform actions, the results of which will reveal if a vulnerability is present.

For example, if you're testing for a command injection vulnerability, you might try to pass a test string that creates a new file on the filesystem used by the application.
If the file appears after you send your test string to the application, you know that a vulnerability is present.

On Screen Text

Observe Application Behavior

Injection Exploitation Process Overview



Narration

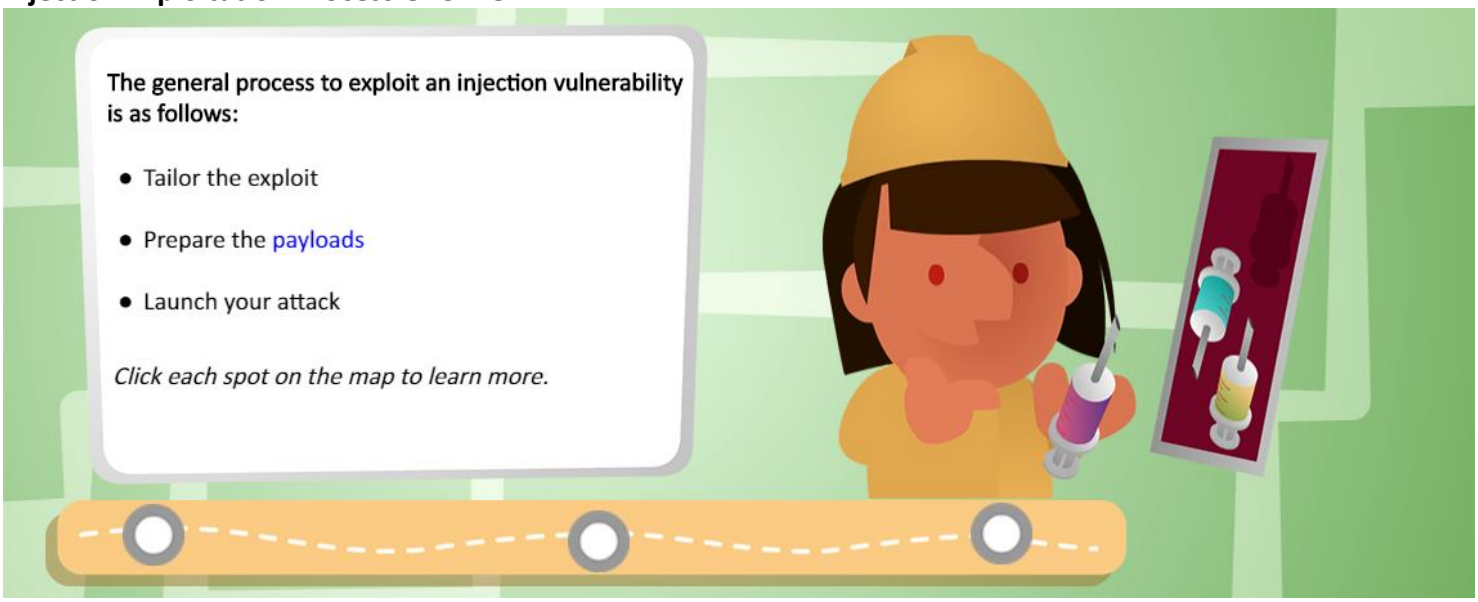
On Screen Text

Injection Exploitation Process Overview

- Identify sources of input
- Identify contexts for input
- Send test attack pattern strings
- Observe application behavior

If you have reviewed all dots, click Continue.

Injection Exploitation Process Overview



Narration

Once you have identified that a specific type of injection vulnerability is present, then you may wish to exploit it further to determine how much additional leverage you can gain.

The general process to exploit an injection vulnerability is as follows:

First, tailor the exploit.

Next, prepare the payloads.

Finally, launch your attack.

Click each step to learn more.

On Screen Text

Injection Exploitation Process Overview

The general process to exploit an injection vulnerability is as follows:

- Tailor the exploit
- Prepare the payloads
- Launch your attack

Click each spot on the map to learn more.

Tailor the Exploit

Narration

First, tailor the exploit. An injection exploit is usually a text string that contains special characters or keywords that will trigger the vulnerability and cause a portion of the string to be executed as instructions rather than processed as data.

Tailoring the exploit means making your exploit work as intended. Usually, you will start with the attack pattern string that is known to trigger the vulnerability and then adjust it to develop a fully weaponized exploit.

The exploit might be just the crafted data you will pass to the target application, or it might be a utility that will automate some of the exploitation process in addition to passing the crafted data to the application.

Many exploits have configuration options that help them work more reliably for certain systems or change the behavior of the exploit.

It is always easier to tailor an exploit if you have access to the source code of the vulnerable application or if you have a functional copy of the vulnerable application in a lab environment that you control.

Ideally, your lab resembles the target environment as much as possible, although injection exploits tend to be so simple that it is often possible to tailor the exploit using just the live target.
However, the longer you take to tune your exploit in a live environment, the more likely your activities are to be discovered, which is undesirable for the kind of covert penetration testing where weaponizing injection exploits may be needed.

On Screen Text

Tailor the Exploit

Prepare the Payloads

Narration

Next, prepare the payloads. If the objective of your test is to determine how much additional leverage you can gain from the discovered vulnerabilities, then you will need a payload, or multiple payloads, to take advantage of the discovered injection vulnerability.

For example, for a command injection exploit, the payload will be the set of commands that you wish to execute with the privileges of the compromised application.

For an SQL injection exploit, the payload will be the set of queries or statements that you wish to execute on the database used by the vulnerable application.

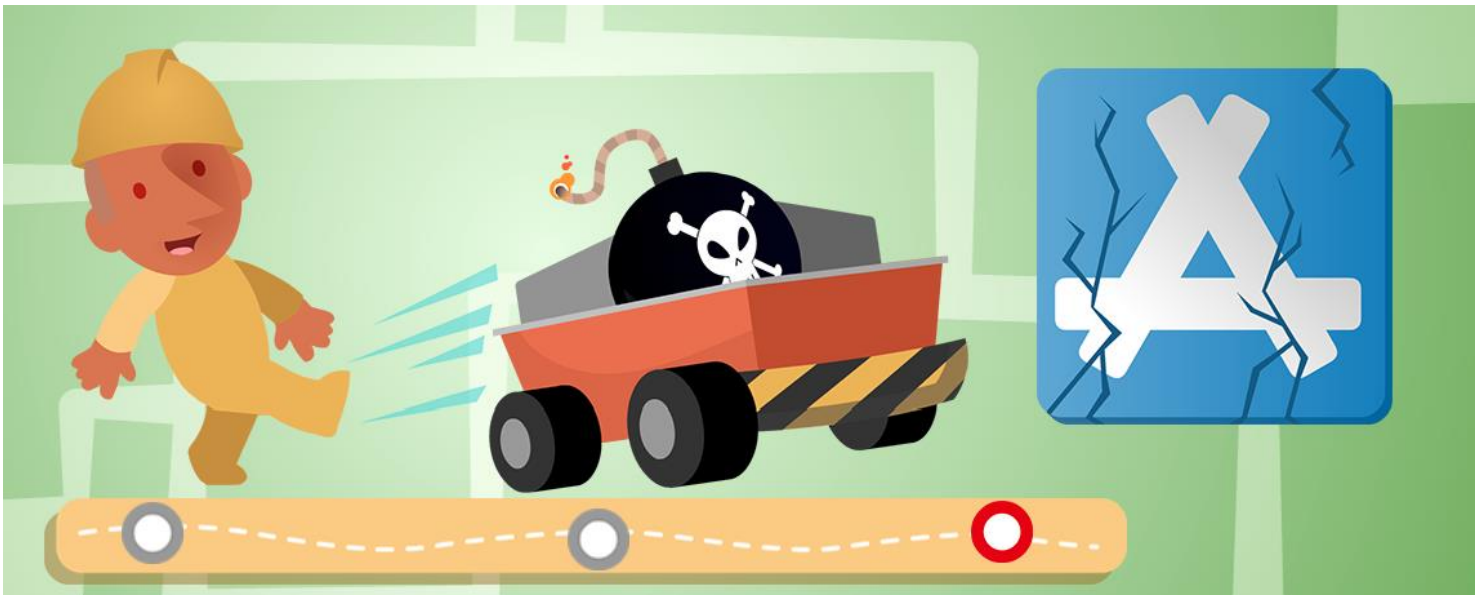
It is likely that you will need to adjust your injection payloads during exploitation, but you should have some idea about what you wish to accomplish, and how, before you start.

The payload is combined with the exploit when executing an attack—the exploit triggers the vulnerability and passes control to the payload, which in turn performs the actions that the attacker desires.

On Screen Text

Prepare the Payloads

Launch Your Attack



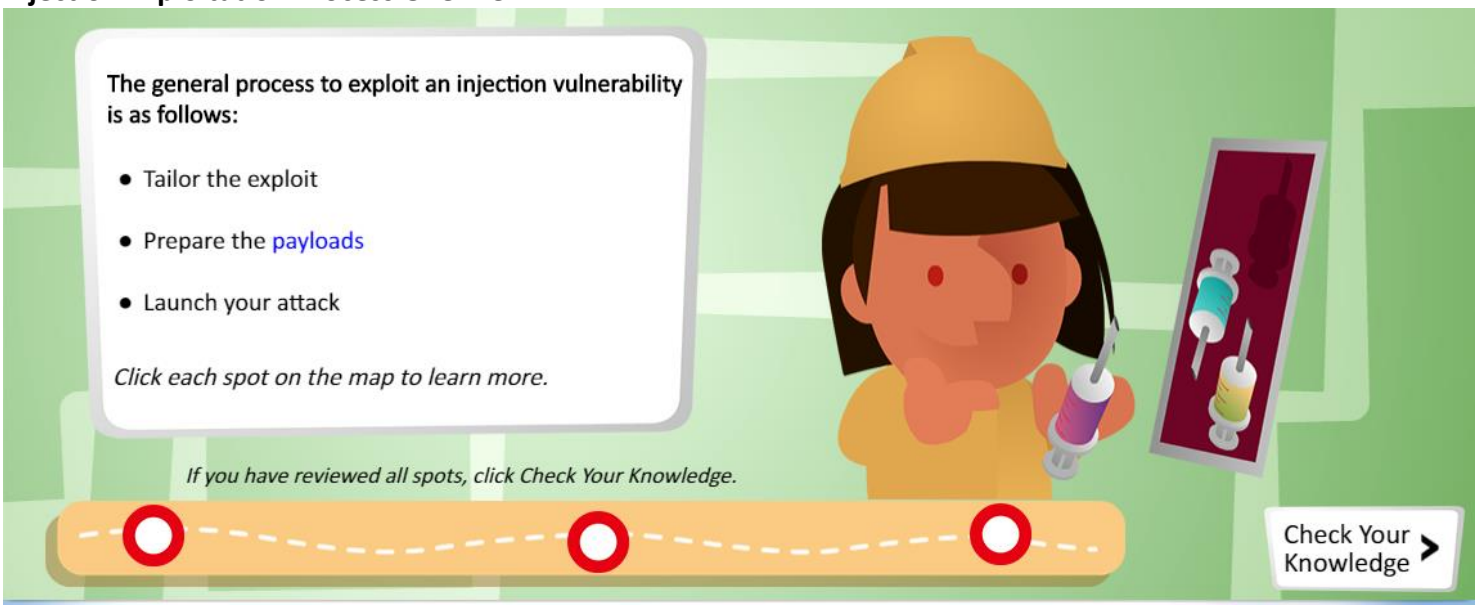
Narration

Finally, once you have your exploit and payloads ready, you can launch your attack. Pass the exploit data to the vulnerable application and observe the results. If your exploit is a utility, then launch the exploit with the correct configuration options to launch your attack.

On Screen Text

Launch Your Attack

Injection Exploitation Process Overview



Narration

On Screen Text

Injection Exploitation Process Overview

The general process to exploit an injection vulnerability is as follows:

- Tailor the exploit
- Prepare the payloads
- Launch your attack

Click each spot on the map to learn more.

If you have reviewed all spots, click Check Your Knowledge.

Interaction

Narration

You are preparing to test an application that resolves Internet hostnames for injection. There is only one input field—the hostname to resolve.

You infer that the application passes the input data to an external command to resolve the hostname, which means that a command injection vulnerability might be present.

What is your next action?

On Screen Text

Interaction

What is your next action?

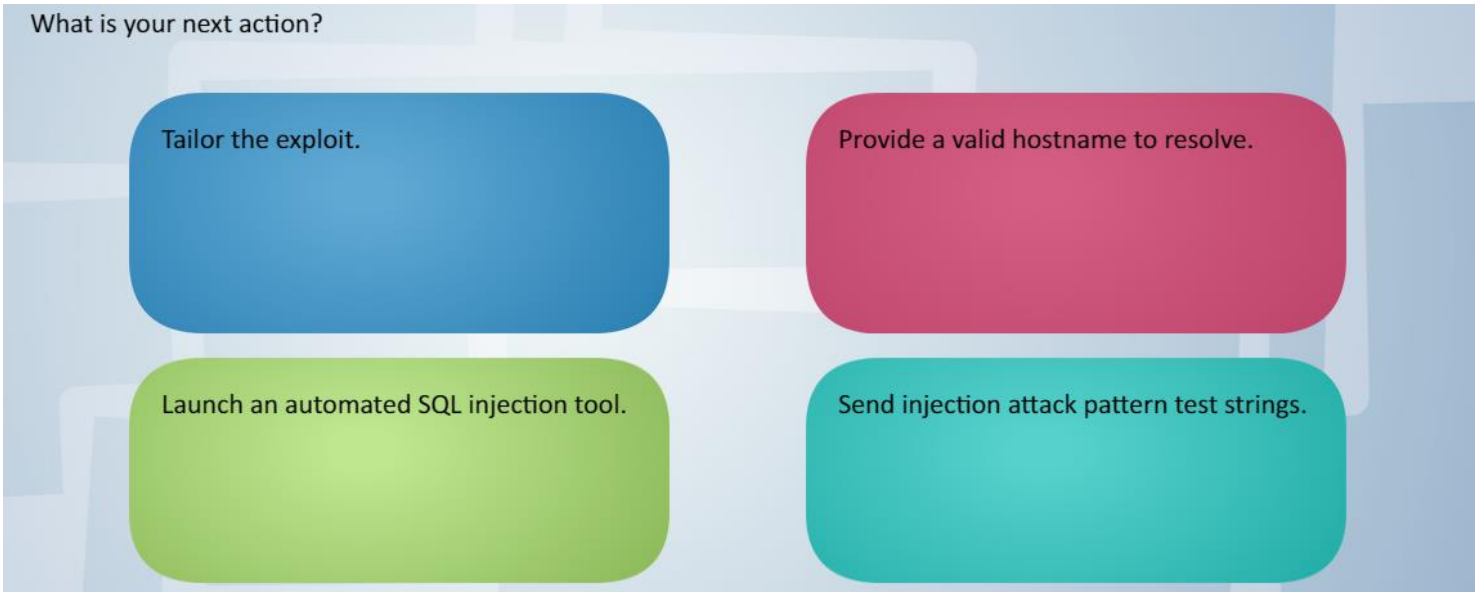
Send command injection attack pattern test strings. - **Correct**

Resolve localhost.

Launch an SQL injection tool.

Tailor the exploit.

Interaction (Cont.)



Narration

You have sent a command injection test string.

The application displays an output that indicates the presence of a command injection vulnerability.

You wish to gain additional leverage using this discovered vulnerability.

What is your next action?

On Screen Text

Interaction (Cont.)

add-to-your-blog.php

ajax

arbitrary-file-inclusion.php

authorization-required.php

back-button-discussion.php

browser-info.php

capture-data.php

captured-data.php

captured-data.txt

classes

client-side-control-challenge.php

credits.php

data

database-offline.php

directory-browsing.php

dns-lookup.php

document-viewer.php

documentation

framer.html

framing.php

hackers-for-charity.php

home.php

html5-storage.php

images

includes

index.php

installation.php

javascript

level-1-hints-page-wrapper.php

login.php

owasp-esapi-php

page-not-found.php

password-generator.php

passwords

pen-test-tool-lookup-ajax.php

pen-test-tool-lookup.php

php-errors.php

phpinfo.php

phpmyadmin

phpmyadmin.php

privilege-escalation.php

process-commands.php

redirectandlog.php

register.php

rene-magritte.php

repeater.php

robots-txt.php

robots.txt

secret-administrative-pages.php

set-background-color.php

set-up-database.php

show-log.php

What is your next action?

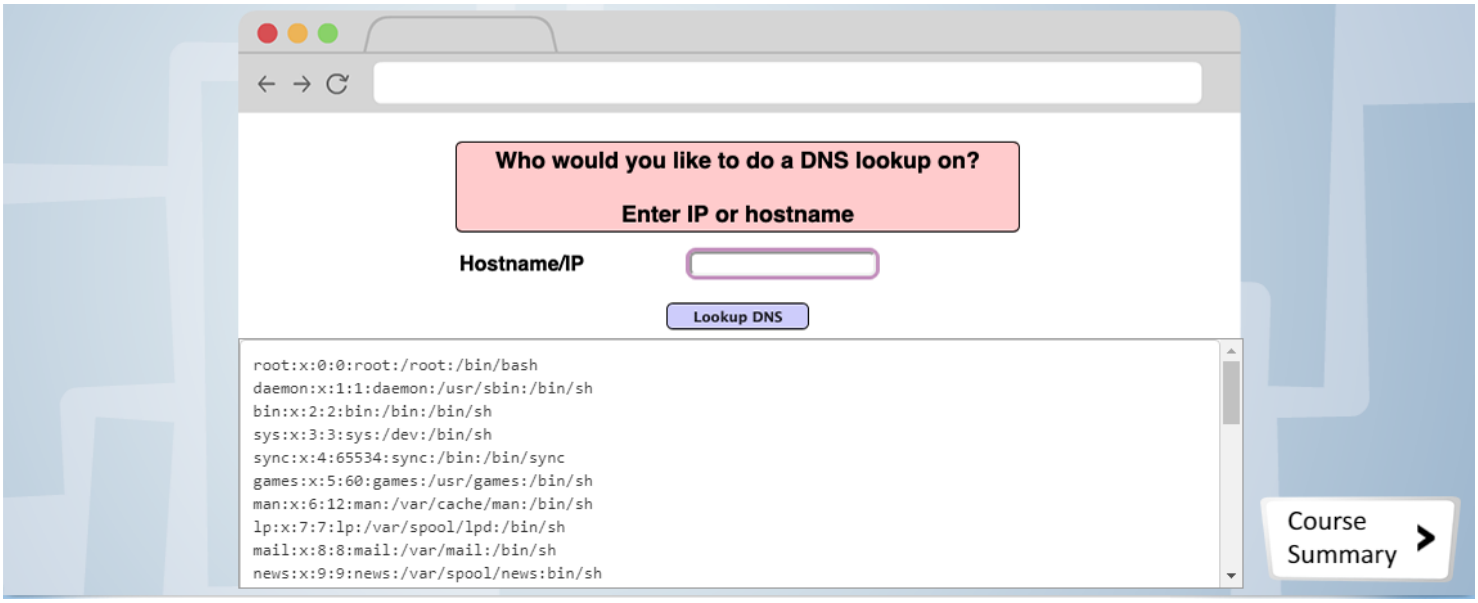
Tailor the exploit. - **Correct**

Launch an automated SQL injection tool.

Provide a valid hostname to resolve.

Send injection attack pattern test strings.

Interaction (Cont.)



Narration

You have successfully identified a command injection vulnerability and tailored it to extract sensitive data from the vulnerable system.
You can use additional payloads to perform additional actions on the target system.

On Screen Text

Interaction (Cont.)

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/bin/sh

man:x:6:12:man:/var/cache/man:/bin/sh

lp:x:7:7:lp:/var/spool/lpd:/bin/sh

mail:x:8:8:mail:/var/mail:/bin/sh

news:x:9:9:news:/var/spool/news:bin/sh

uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh

proxy:x:13:13:proxy:/bin:/bin/sh

www-data:x:33:33:www-data:/var/www:/bin/sh

backup:x:34:34:backup:/var/backups:/bin/sh

list:x:38:38:Mailing List Manager:/var/list:/bin/sh

irc:x:39:39:ircd:/var/run/ircd:/bin/sh

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh

nobody:x:65534:65534:nobody:/nonexistent:/bin/sh

libuuid:x:100:101::/var/lib/libuuid:/bin/sh

syslog:x:101:102::/home/syslog:/bin/false

klog:x:102:103::/home/klog:/bin/false

mysql:x:103:105:MySQL Server,,,:/var/lib/mysql:/bin/false

landscape:x:104:122::/var/lib/landscape:/bin/false

sshd:x:105:65534::/var/run/sshd:/uuser/sbin/nologin

postgres:x:106:109:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash

messagebus:x:107:114::/var/run/dbus:/bin/false

tomcat6:x:108:115::/usr/share/tomcat6:/bin/false

user:x:1000:1000:user,,,:/home/user:/bin/bash

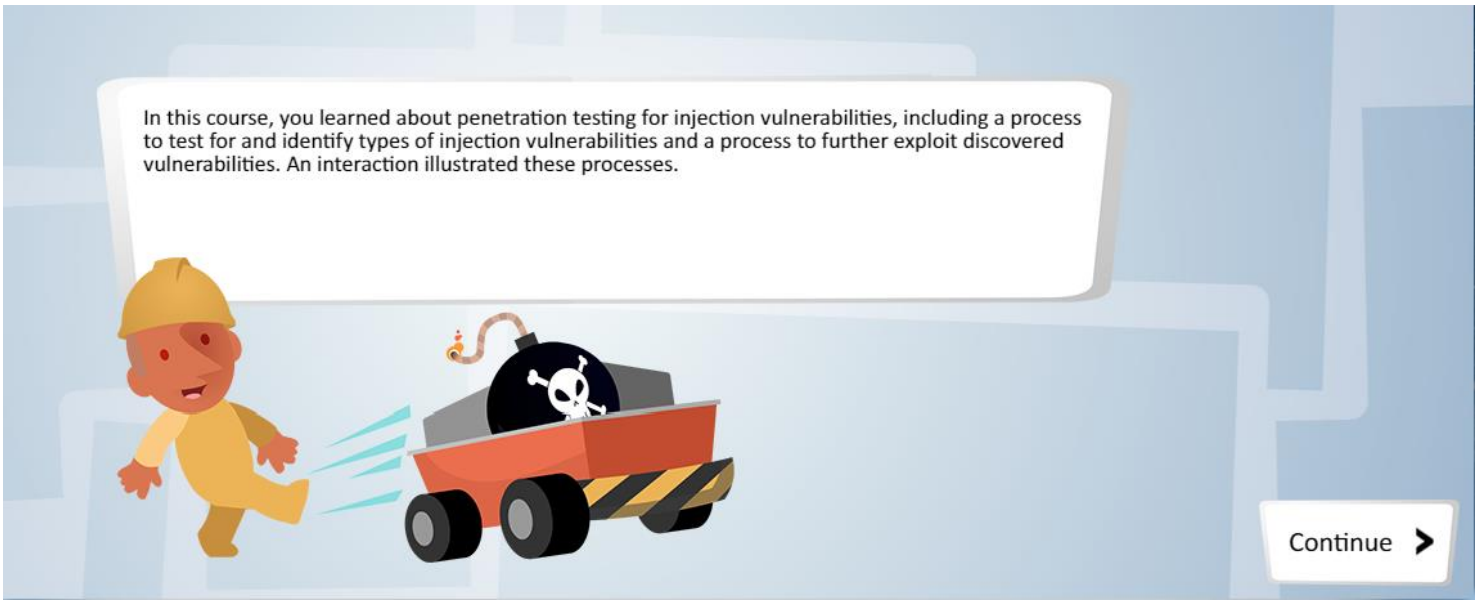
polkituser:x:109:118:PolicyKit,,,:/var/run/PolicyKit:/bin/false

haldaemon:x:110:119:Hardware abstraction layer,,,:/var/run/hald:/bin/false

pulse:x:111:120:PulseAudio daemon,,,:/var/run/pulse:/bin/false

postfix:x:112:123::/var/spool/postfix:/bin/false

Course Summary



Narration

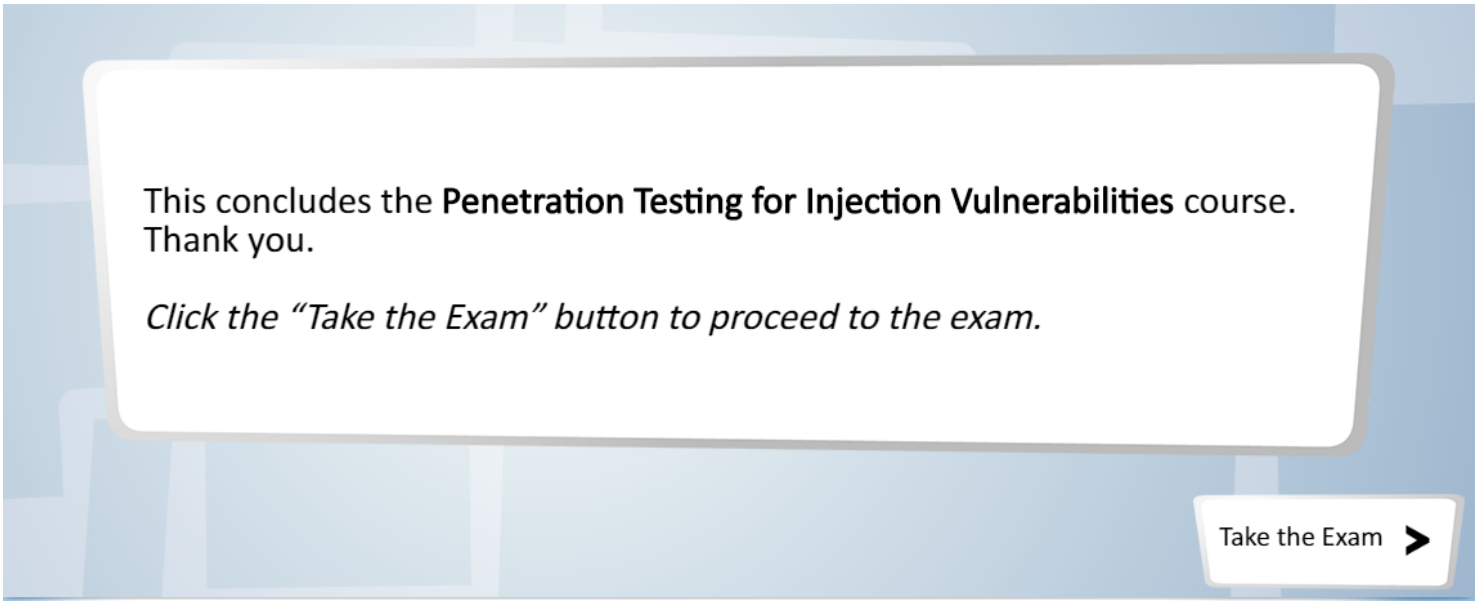
In this course, you learned about penetration testing for injection vulnerabilities, including a process to test for and identify types of injection vulnerabilities and a process to further exploit discovered vulnerabilities. An interaction illustrated these processes.

On Screen Text

Course Summary

In this course, you learned about penetration testing for injection vulnerabilities, including a process to test for and identify types of injection vulnerabilities and a process to further exploit discovered vulnerabilities. An interaction illustrated these processes.

Thank You



Narration

On Screen Text

This concludes the **Penetration Testing for Injection Vulnerabilities c**ourse. Thank you.

Click the "Take the Exam" button to proceed to the exam.

Thank You