Lab 4: Web Security

50.020 Security

Hand-out: February 16
Hand-in: February 23, 9pm

1 Objectives

- Perform SQL Injection and Cross-Site Scripting on a vulnerable web site
- Perform Command Injection to execute arbitrary commands
- We provide you the source code. In case you get stuck, you can have a look to understand better what you are doing

2 Part 1: SQL Injection

- We distribute a `lab4server.py` server script, run it with python3 locally to test out your exploits
- After starting the server locally, go to `http://localhost:5000` to be greeted by a login page. Alice's email is alice@alice.com, but what is the password?
- Based on the example attacks from the class, find a way to log in as Alice!

3 Part 2: Cross-Site Scripting

- Note: restarting the webserver will refresh the SQL database

3.1 Second order (persistent) attack

- Log in as Alice again, and check the different options the website gives you. Can you use one of them to conduct a second order XSS attack?
  - Your goal is to inject some code on a page that the admin user will look at
  - Your injected code (HTML or .js) should make the admin disclose his cookie to you
  - In particular, assume that the admin sees a list of all user data when he logs in: email, name, and password hash
- Find a way to inject a simple XSS string into the database
- Exploit this to obtain the `session` cookie of the victim! By using that session ID, you could take over an existing session the victim has with the website

  - Update your XSS to trick the admin's browser to do an HTTP GET to a URL of the attacker's choice
  - Make the admin visit `http://localhost:5000/news?text=`<adminsession>
  - This will create a public news entry with the admin's session
- After injecting your XSS string, login as admin user to test that the injection works. Use the second password from the `secrets` file
  - Logout after displaying the main site, login as alice again (using the password, or the SQL injection)
  - You should now be able to see the news item!

3.2 First order attack

- Look for a way to temporarily inject code via a specifically crafted URL
- Exploit this to obtain the session of the victim (again)
  - Lets assume the admin is logged into the site when clicking on the link
  - Clicking the link should make the admin visit the localhost webserver with a specific URL, which then causes admin's browser to issue a HTTP GET to `http://localhost:5000/news?text=`<adminsession>
  - Craft the URL, login as admin user, then visit the XSS URL you designed
  - Admin's session should again be leaked as news item
  - Note: it might seem a bit redundant to do a first order XSS attack to make the attacker visit the `news` URL. Try putting the javascript exploit code into a simple html file, and visit it with your browser. Does it still work, without being injected into the page? why not?
- We had to manually enable XSS attacks in our templates, can you spot where? Can you fix all XSS vulnerabilities you found?

4 Command Injection

4.1 Listing secret file

- This time, we will not inject any JavaScript.
- Find a way to execute a shell command on the server by using one of the provided input fields
- Can you exploit this vulnerability to get the content of a local file, e.g. list the file "secret"?

4.2 Opening a reverse shell

- This time, we will try to get a local shell on the target server!
- A "reverse shell" is a program started on the victim machine, that connects to the attacker (to pass through firewalls and similar).

- We need two parts for this: a program waiting on your machine to accept incoming connections
  - Use something similar to `nc -l -p 8080 -vvv` on your machine to open a listener waiting on port 8080
  - When the victim connects, you should be able to type commands, which will be executed on the victim's machine
- We also need a suitable command to run on the webserver (in this case, also localhost) to connect a shell to your machine
  - Hint: have a look at `http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet`
- Inject that command similar to your previous command - Easy!?

5 Hand-In

- Write a short report covering the following topics:

5.1 SQL

- Explain the string you used for the SQL injection and explain why it works.
- How can this vulnerability be fixed in the provided script?

5.2 XSS

- Explain which content you inserted in the second order attack, and how it works
- Explain why the first order attack requires the attack code to be injected, and not just embedded on a third party website
- Explain if you were able to fix the XSS attacks by changing the templates

5.3 Injection

- Explain which command you used to obtain the content of "secret", and why it works
- Explain which command you used to obtain the reverse shell, and why it works