Here Be Dragons

yonmo

2022-12-28

Contents

1 Introduction

One summer day, while enjoying a movie, a friend of mine who will remain nameless–you know who you are–recommened I take a look at the comic viewer javascript code for the well known adult japanese comic publisher FAKKU. Originally that suggestion sounded ridiculous; however, across my time, making and investigating wacky code I have found that some of the most interesting and unknown techniques come from materials marketed to the most unhinged of internet dwellers and those who screw around with them (games, pornography, gambling, etc). As such, on this absurd dare I decided to build my castle. I spent several months analyzing and looking for bypass methods for FAKKU's comic viewer client-side application code. In so doing, I learned way way too much about the restrictions of browser technology and created some ridiculous techniques I figured may be useful to web hackers everywhere.

2 Research and Development

I began my search by attempting to access the browser toolbox utility, inspect element, which would enable me to view large portions of the media loaded and set breakpoints in their script to step through the script's functionality. Immediately it became obvious that this had been mitigated by the program's runtime, wherein it would call the "debugger" keyword repeatedly, its call can be identified within the keyword list in two seperated locatations split into two strings "debu" and "gger", which later in the script are concatenated togethor in order to evaluate.

2.1 Cloning the Web Application Locally

In order for me to identify the functionality of the script, I attempted to clone the html, css, and js onto a local webserver I spun up in order to manually step through the code. Once I modified the js code to not include the debugger keyword, I realized due to the nature of how the objects in the DOM are created from the loaded media elements I would need to have a logged in user on the domain to even get the content necessary for the functions to be called. So, I decided to try a different method for obtaining information about the script and the media protections.

2.2 Inspecting Inconsistencies in the DOM and Prototype Tree

fkelfmel

2.3 Prototype "Curing" the DOM through Object Inheritence

In order to bypass the prototype poisoning the developers loaded inherently into the page, I just needed to obtain a clean prototype and "cure" the objects.

Listing 1: To Create Elements

```
0 // Prototype Poison The DOM to Allow Me To Create Elements
1 Object.setPrototypeOf(document, Object.getPrototypeOf(document.implementation.createDocument('http
      ://www.w3.org/1999/xhtml', 'html', null)));
```

2.4 Proxy Poisoning For Intel

fmekmfle

3 Conclusions and Ideas For New Bypasses

vemkvemvk

4 Appendix

Listing 1 shows the code utilized for Object Chain Poisoning.