# The Honeypot Chronicles

Ryan Sheehan

December 2019

## 1 Abstract

In order to think like an attacker, it is important at times to play the role of one. This is common practice in the world of computer security. Students of the discipline are encouraged to take part in capture the flag games where they attempt to find vulnerabilities in a provided system, as this kind of learning creates more proactive analysts. Reading CVE, CWE, and OWASP lists can only provide an account of how a system was already compromised, and this is often too reactionary. A good systems analyst needs to be able to identify flaws that may have never been encountered before. This is how playing the role of the attacker can help us. For my final project, I wanted to go a step beyond the format of the CTF game. Instead of being the attacker on the outside trying to get in, I wanted to be the attacker on the inside who owns the system in question and is attempting to use it maliciously. Luckily, I found myself with a perfect opportunity to do so. A couple of my friends and I take part in a group here at Tufts that hosts pop-up restaurants and we decided that our next event was going to be a one night only on-campus food delivery service. Naturally, I figured this would be a perfect environment to test my newfound security knowledge and see what was possible to find out about any given user that visited the site hosted on my server. So that became a goal of mine.

## 2 Introduction

The night of October 19th at approximately 10PM, t86latenight.com went live. The site stayed up until 12AM, at which point we took it offline as we were no longer taking orders. I used an AWS S3 bucket for hosting and registered the domain for use with it. In the time that the site was live, we received 36 orders and had 73 visitors to our site. A picture of the landing page is included in figure 1. My goal with this website, simply put, was to see how much I could possibly learn about each user that visited my website. I had no intention with doing anything with this information (and in fact I plan on disposing of it after finishing this paper), but my main interest was learning if the amount of information I could collect about someone should be of particular concern.

From this, I wanted to learn how a user could then prevent my site and other sites like it from collecting this information in the future.

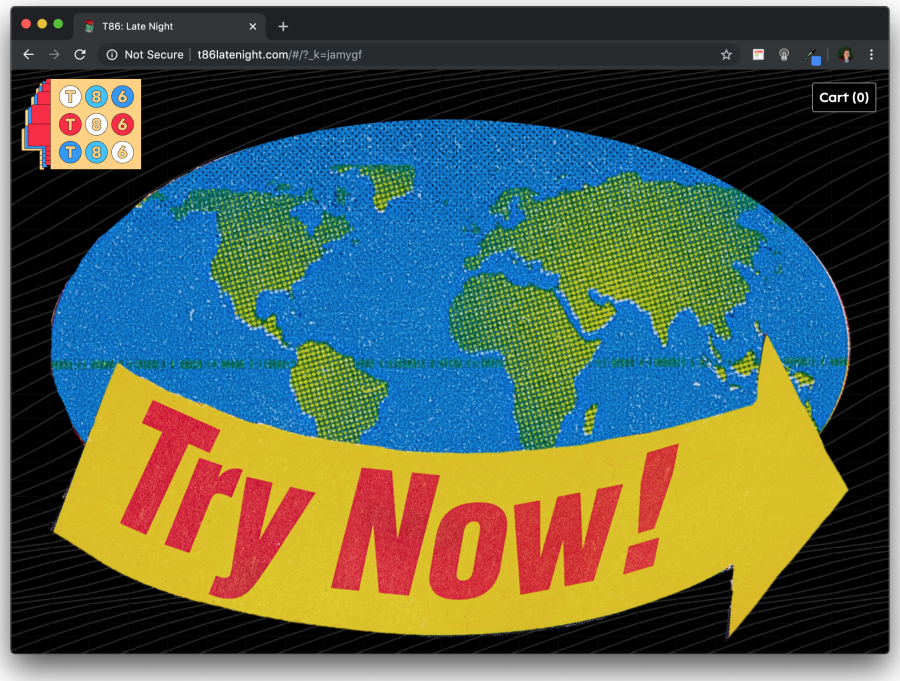


Figure 1: Landing Page

## 3 To the Community

I chose this topic because I have found time and time again that I always learn best by making something and I wanted to have an environment of my own creation to play around with. Additionally, I thought it would be interesting to see what I, a very amateur web programmer, was able to build as to benchmark that against what a more sophisticated attacker would thus likely be capable of.

## 4 Background Information

To start, I wanted to make use of the more obvious methods that most websites employ by default a lot of the time. This of course meant I had to go register for Google Analytics account. By doing so, I gained access to a lot of general information about the demographics of users on my website. Namely, Google

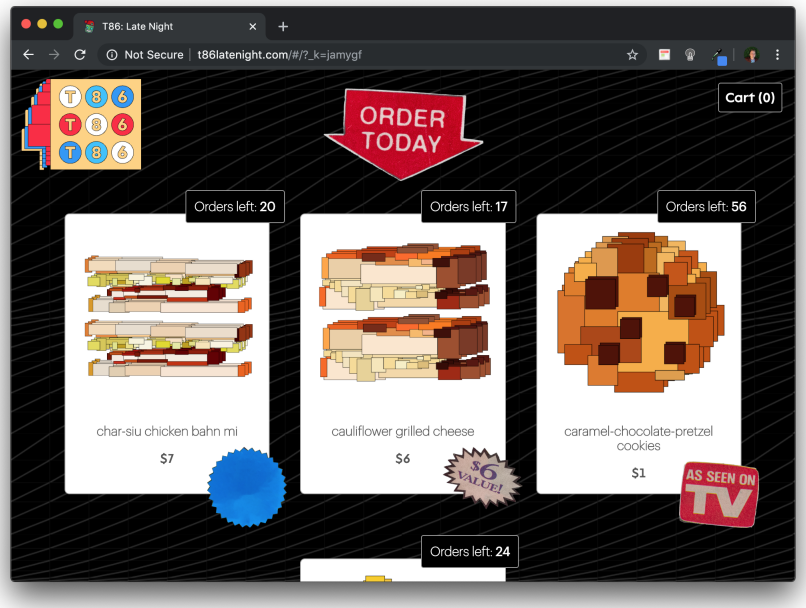


Figure 2: Menu Selections

Analytics provides tools for accessing user location and user system information. Included in these umbrella categories are:

- Whether or not this was a user's first time on the site
- How long their session lasted
- Their language
- Their country
- Their city
- Their browser
- Their operating system (pc)
- Their service provider (pc)
- Their operating system (mobile)
- Their service provider (mobile)
- Their resolution

Already we were off to a good start. Next I wanted to find out more about each of my user's browsers than Google Analytics would tell me. I wanted to see what kind of browser plugins each person was employing. For users coming to our site on Firefox all we had to do was ask 'showExtensions();' and we'd be presented with them. From my research, it appeared that there used to be a more circuitous way to check for Chrome plugins that had since been fixed. The method for Chrome involved checking for extensions by attempting to load images in the format 'chrome://adblockplus/skin/adblockplus.png' where "adblockplus" would be replaced by whatever extension you would want to check for. However, even though Chrome has in fact patched that particular vulnerability, we can use the idea in a similar way to check what websites our users are currently logged in to [4]. By attempting to use an image tag in our site to load the favicon of the site we wish to know if the user is logged in on, we will either be presented with the favicon or an onload error in which case we know the user is not logged in. Same Origin Policy is strict for HTML pages, but it allows to receive images from other origins which is how we can get away with this. For example, to see if a user is logged on to Facebook, all we would need to do is see if ¡img src="https://www.facebook.com/login.php?next=https

Next, I wanted to get information about the users connection since I had seen this was possible to do with fairly limited JavaScript. For this I created a script called speed-test.js that more of less runs a slightly more sophisticated version of the code in figure 3.

```
var startTime, endTime;
var download = new Image();
download.onload = function() {
    endTime = (new Date()).getTime();
    showResults();
}
```

Figure 3: Speed test

This let me draw assumptions about what kind of network our user was on. This script also gets the user's IP.

Next I wanted to see what was possible as far as acquiring information about the users internet history. Using a module from Google Analytics we can get access to the last site that the user visited. If I wanted to go further with this I could have added a cookie to each user's browser that continually tracked the pages they were visiting (potentially even after they had left my site), but for this project I was only really concerned with what I could extract within the context of my own site and not beyond.

And then for fun I decided to get the users battery percentage and gyroscope information if they were on mobile. [3]

But now for the kicker. The site is structured as a very normal e-commerce platform. You have a grid of items to select from and when you click on one you may select the number of it that you would like and add it to the cart. Once you are in the cart you checkout and are brought to a form where you are inquired to fill out some necessary information. What this page does not make obvious however is that it is a sting for autofill phishing [2]. By having any number of

hidden fields with information that was not relevant to my site, if a user was to try and autofill to fill out the form they would have unknowingly completed input fields for any number of other prompts (whether that be their job, their address, and even something as compromising as their credit card number if they were careless enough to have their browser remember it).

# 5 Defense

The benefit to understanding how to take advantage of these oversights is that it is much easier to make an informed response in deciding how to respond to them. One of the big questions I had going into this project was whether or not the amount of information a browser could get out of you merited anyone going out of their way to inhibit it from doing so. If all you do is check Facebook and Twitter for example, who really cares if your IP is easily accessible? But this analysis is not to make a moral judgement about privacy, it is simply to point out what could be observed about you and what you could do to prevent that from happening. First of all, there is a very common theme in a lot of the information extrapolation processes above: they run methods in the background unknown to the user. A potential countermeasure could be to begin using NoScript [5]. NoScript simply makes it so that by default you have to approve a script running rather than it running by default. You can enable JavaScript, Java and plugin execution for sites you trust with a left-click. However, NoScript is only available on desktop distributions so this means mobile browsing still remains more at risk. That being said, using NoScript on desktop would cover a lot of bases and prevent nearly all but the form autofill phishing vulnerability outlined above.

As far as preventing autofill fishing, the best course of action is to disable that feature on your browser or to just avoid using it when you aren't 100% certain about its integrity.

Lastly, and most obviously is just to use common sense online and particularly when it comes to purchases. It is entirely up to the site admin (me in this case) to choose how they go about dealing with your information the second you hit the submit button. You effectively lose ownership. In the case of this website, each order was processed by using nodemailer to send on submission of the form as pictured in figure 4. This isn't exactly the most secure way of doing this as far as the client is concerned but for me as the site owner trying to get this site up it was the most practical and that was the trade off I decided to make.

# 6 Conclusion

In summation, from my measly little food cart of a web app I was able to learn about a users:

- history with my site

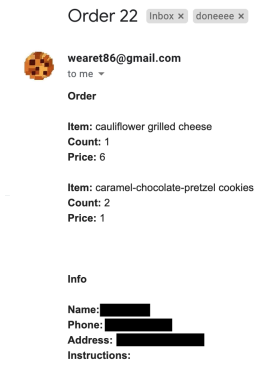


Figure 4: Order from t86latenight.com

- session time
- language
- country
- city
- general lat / long
- browser
- operating system (pc or mobile)
- service provider (pc or mobile)
- other sites they are currently logged on to
- screen resolution
- browser plugins (somewhat)
- name
- phone number
- email (if they used autofill and had their email remembered by their browser)
- order information (obviously)
- battery percentage
- gyroscope information
- network speed

- ip
- last site visited

If this amount is worrying, think about how much more someone with years of experience would likely be able to add to that list. For example, someone who knows more about networking and WebSocket connections than I would probably be able to get an idea of what other devices are on a users local network. This is all to say: be careful. It's supposedly possible to identify an individual with just the metadata of 3 "anonymized" credit card purchases [1] so think about how easy it would be to identify an individual when taking all of the above into consideration.

A neutralized site is still live at t86latenight.com (meaning all of the scripts used to extract the information above here have been disabled). A repository of its source code can be found at: https://github.com/Ryan-Sheehan/COMP-116-FINAL

# References


[1] John Bohannon. Credit card study blows holes in anonymity. *Science*, 347(6221):468–468, 2015.

[2] Samuel Gibbs. Browser autofill used to steal personal details in new phishing attack, Jan 2017.

[3] Alex Hern. Your battery status is being used to track you online, Aug 2016.

[4] Robin Linus. Your social media fingerprint.

[5] Darlene Storm and Darlene Storm. Snowden at sxsw: We need better encryption to save us from the surveillance state, Mar 2014.