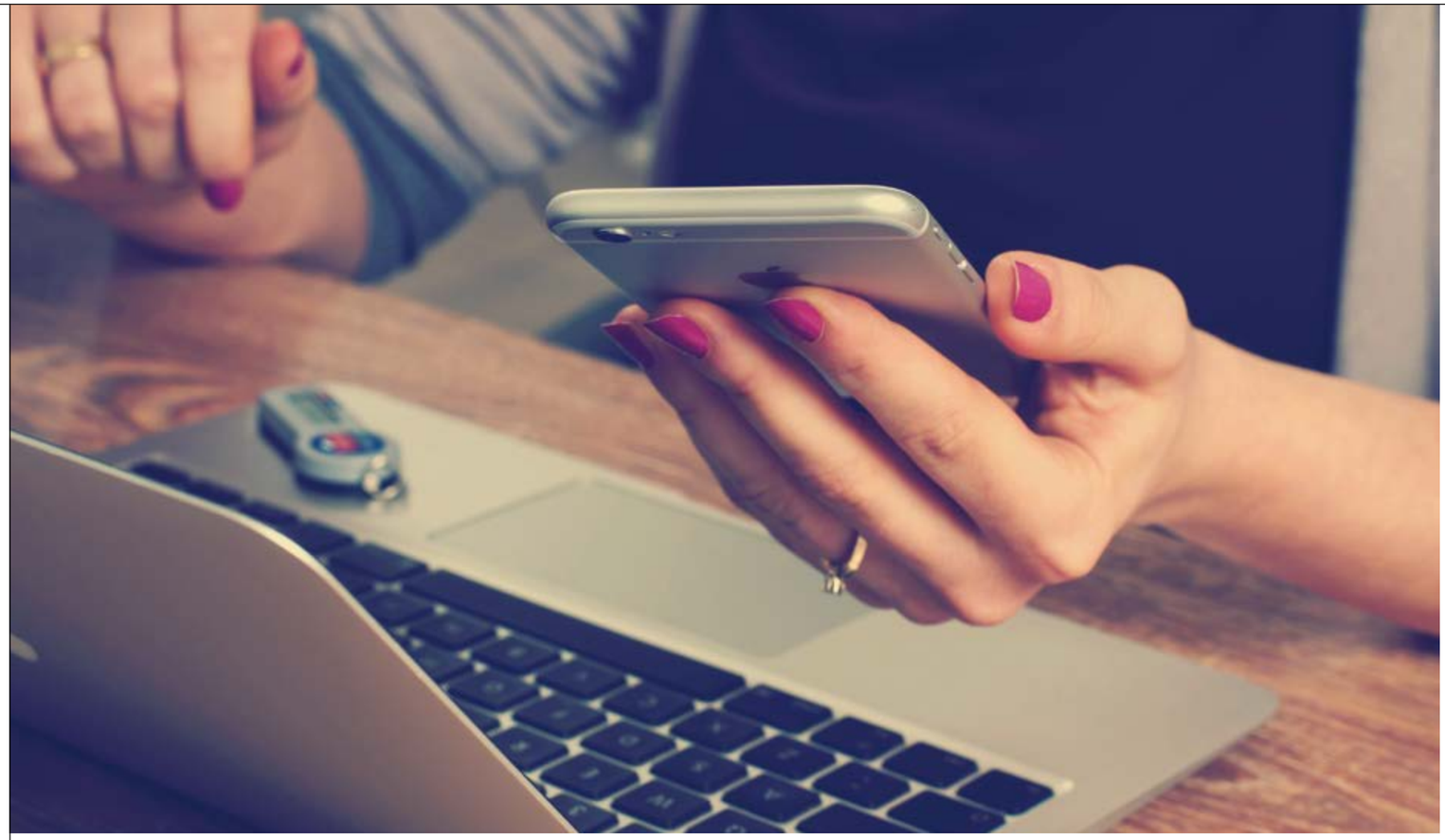


APPLICATIONS – OUR DIGITAL SERVANTS

Dr Simon Wiseman *examines potential security flaws behind apps that have designed to make our lives easier*

We use applications all the time, whether on the desktop, mobile or via the web. They handle our sensitive data and let us control critical operations, at home and at work. Applications are supposed to be our servants, doing what we want in a digital world.

In reality applications do what their authors tell them to do. This usually aligns with our wishes as users – but not always, for example in-app advertising often annoys. Other features prove much more dangerous, and the software author is often unaware that they have provided the feature. Of particular concern are application features that give attackers control of your environment through the content that they provide.

A simple example from the early days of the web, is the feature of web browser script functionality that allows scripts to create files. Modern browsers no longer allow scripts to do this because it was found to be unsafe, but attackers could create a web page with a script that creates an executable file. This could then be placed in a location where the operating system automatically runs it when the user logs in. So just by visiting the attacker's website, a user gets to run the attacker's own application, which can then do anything that the user can do as well as probably doing quite a few things that the user wouldn't want to do.

There is now much greater awareness of this kind of issue in the developer community, and it is unlikely such dangerous features will be introduced into new applications without someone calling into question the wisdom of doing so. But that's only true of features that are known. What happens if the feature is so obscure that nobody considers it? What happens if the feature is undocumented so nobody knows it is even there?

A recent case of an obscure feature catching out the security community involves the ability to embed LNK files in Microsoft Office documents. A LNK file is what you get if you create a shortcut to an application or document on your Microsoft Windows desktop. The file contains the name of the application to run and any parameters to give it. LNK files are harmless as they just help you find stuff and embedding files in Office documents is an important feature that lets you, for example, embed Excel spreadsheets in Word documents. But the two features can be used together – an LNK file can be embedded into a Word document. This is quite obscure and has no real purpose. It is just an unintended consequence of how embedding works.

The issue is that LNK files can invoke applications such as Powershell – a general-purpose scripting engine used in Windows to automate all sorts of system activity. This means that an attacker can get a user to run a malware script by putting it into an LNK file that is then embedded in a Word document.

FIXES TO THE PROBLEM

Once attackers started exploiting this kind of attack, fixes were applied to block them. In particular, it is no longer possible to run Powershell in this way – providing that you have the latest operating system installed and that it is fully patched.

But it is still possible to embed LNK files in Office documents and have them run other applications. This is considered safe, because nobody (willing to disclose a vulnerability) has yet thought of a way of exploiting it.

This highlights the fact that obscure features of applications are dangerous. It may be their capability isn't fully understood or that combinations of well-known features work in strange ways. So we can't really trust the applications we run to only work as we intend.

Another recent case involved an old feature of Word that is no longer documented. Word documents can contain field codes, which display some value that is automatically updated based on other data, such as the number of a particular page.

One field code (DDEAUTO) allows data to be extracted from external sources, like a database and displayed in the Word document. This uses the Dynamic Data Exchange feature of Windows – the mechanism that allows applications to share information, such as when using a mail merge.

The DDEAUTO field code allows the application that is the source of the data to be specified, along with any parameters needed. The intended usage is to start up a database application, but it can run a scripting engine such as Powershell, as with the LNK feature already discussed. This means that an attacker can get Word to run their code on your behalf when you open a specially crafted document.

The DDEAUTO field code is left over from early versions of Word. It is no longer described in the documentation, and doesn't appear in the user interface for creating fields. But it still works. Since being exploited in an attack, it is now well known, but before then it was an undocumented feature, so anyone checking for malware in documents would most likely have overlooked it.

POTENTIAL HIDDEN ISSUES

The question is, are there other features in Word that are completely undocumented? Perhaps there is another field code that was implemented but never properly released, and nobody knows it is there. There are so many possibilities that testing will never find such a hidden feature. Because access to the source code would be needed, along with a lot of time to analyse it. If we can't be sure of precisely what features our applications have, how can we be sure that none of them are dangerous?

Take for example a CSV file. This is a text file that represents a table of data. Each line contains the data of one row. The fields in a row are separated by commas. Normally, CSV files contain numeric data and text strings, but Excel also allows them to contain formulae – any field starting with an equals sign is treated as a formula. This comes as a surprise to most people that use CSV files, as the format is thought of as a simple way of exchanging basic data.

On top of this, each row of a CSV file can have a different number of fields – including none (a blank line) or just one (no commas) – and text fields can contain non-printing characters. What this means is that any file can be treated as a CSV file. For example, take any PNG image file, change its name to have a CSV file extension and then open it with Excel. The spreadsheet that appears will not be useful, but it works. All this is pretty bizarre, but how could it be potentially exploited?

CONTENT THREAT REMOVAL ASSUMES ALL DATA IS UNSAFE AND ALLOWS NONE TO PASS

Formulas give spreadsheets their power. Most people use them for calculations on data within the spreadsheet, but several formula functions allow data to be pulled from other sources. In particular, an Excel formula can get data from an external web server or from another application using DDE. This gives attackers ways of getting Excel to run their code and so is obviously dangerous, but attackers can use CSV files to hide what they are doing. A PNG image file can be treated as a CSV file. A CSV file can contain formulas. A formula can cause attacker's code to run. This means a PNG image, carefully constructed and opened in the right context, can be dangerous.

Can we be sure our applications don't have features that allow this to happen? No, because we don't understand what functionality they have and even if we did, the interactions between features would be so complex we could not understand the implications.

Most anti-malware solutions look for known attacks, either by checking for characteristic patterns in the data or checking for characteristic behaviour by opening the data in a sandbox. Both approaches fail to stop new attacks and attacks that have been crafted to evade detection.

CTR defends against malware without attempting to detect it

Picture credit: Getty

Deep Content Inspection (DCI) is an alternative, long used by government, which has re-appeared in the commercial space as Content Disarm and Reconstruct (CDR). With DCI/CDR, data is analysed in detail, looking for executable data and malformed or unusual structures that might lead to code execution. Any code is considered harmful and is not allowed to pass. This is in contrast to malware detection, where only code known to be harmful is blocked, making it less strong.

CONTENT THREAT REMOVAL

However, DCI/CDR only works if the applications that handle the data are fully understood. Unfortunately, the obscure, undocumented and bizarre features being exploited by attackers mean this isn't the case. In practice, the technique only works against known attack techniques, much like the malware detection strategy. This limitation was recognised some time ago, and government systems started to move away from DCI to a different technique based around transformation. This is now being delivered into commercial systems as Content Threat Removal (CTR) by companies such as Deep Secure.

CTR defends against malware without attempting to detect it. Rather than try to determine if data can

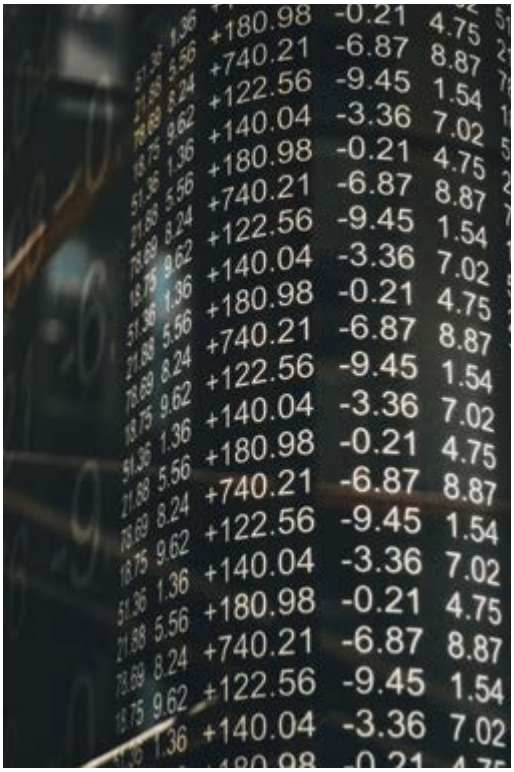
WE CAN'T REALLY TRUST THE APPLICATIONS THAT WE RUN TO ONLY WORK AS WE INTEND THEM TO

be safely allowed into a system, CTR assumes all data is unsafe and allows none to pass. It extracts the useful business information that the data is carrying, then builds completely new safe data to hold it. This way no unsafe data ever gets into the system, but required business information does.

With CTR there is no need to understand all the ways that data can be unsafe. All that is required is to know one safe way of representing some information. With malware detection, if the developers overlook some vulnerability the defences fail. With CTR the data is always discarded, so if there's an unknown vulnerability the defence still works. The CTR developers do need to know a safe way of representing information, which means it is possible to test a single feature to make sure it works as expected. Testing cannot find unsafe features, but it can check a feature is safe.

Using CTR, the obscure embedded LNK files serve no useful business purpose and so are removed by the transformation. The undocumented DDE field codes are discarded because they are undocumented. The bizarre behaviour of a CSV file is made safe by removing all formulae, or those formula functions that are unknown. So, when these issues were publicised users of malware detection and DCI/CDR defences needed an update, but CTR users remained safe ●

Dr Simon Wiseman is CTO Deep Secure with over 30 years of experience in the field of Government computer security. He is responsible for the technical strategy at Deep Secure, devising unique solutions to hard cyber security problems. He has pioneered work on the use of data transformation to defeat attacks in digital content.



Obscure features are a worry, but there are cases of well-known features that have bizarre behaviour. One that recently came to light relates to the way Excel handles (CSV) files

Here the browser had a handy feature, in that scripts that create files are useful as web applications can integrate closely with your desktop. But it also had unintended consequences as it allowed malware to be delivered to the desktop

Picture credits: Ilya Pavlov and Tyler Easton