virus BULLETIN Covering the global threat landscape

# VBA IS NOT DEAD!

*Gabor Szappanos*
Sophos, Hungary

VBA macro malware was the dominant beast of the second half of the 1990s. This was the era of the simple WordBasic and later VBA macro viruses, the latter starting their sharp rise on the prevalence lists with the appearance of Concept, and then slowly declining after 2001, with occasional appearances lower down the prevalence charts in the years that followed. Figure 1 shows virus prevalence by type using data aggregated from *Virus Bulletin*'s monthly malware prevalence stats.

In the past five years, macro viruses (and more generally, macro malware) could be considered practically extinct – thanks mostly to the security improvements that were introduced over that period of time to their main target, the *Microsoft Office* products.

However, this does not mean that our virus lab no longer encounters malicious *Word* documents and *Excel* workbooks – on the contrary, they have appeared on our virus radar quite frequently recently, but for a different reason. The new sightings are exploited documents using one of the *Office* vulnerabilities to drop or download some trojans or backdoors.

In the past couple of months, we have observed the resurgence of malicious VBA macros – this time, not self-replicating viruses, but simple downloader trojan codes.

Figure 2 shows the prevalence of the document-based infection reports received by our virus lab in a three-month period, covering March, April and May. We can see that, following the usual suspects (the CVE-2012-0158 and CVE-2010-3333 vulnerabilities), the VBA downloaders are the next most prevalent.

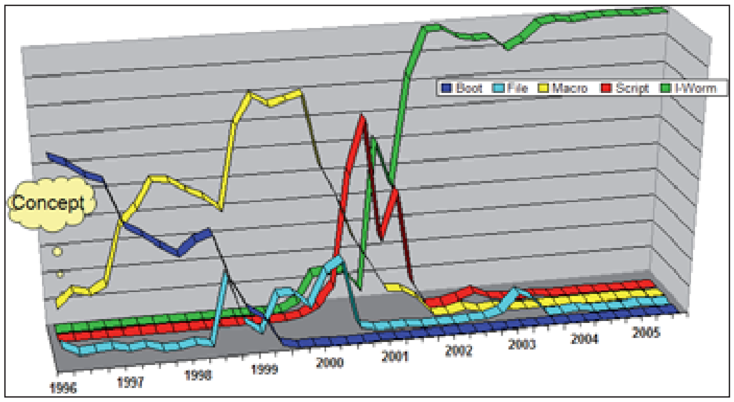


*Figure 1: Virus prevalence by type.*

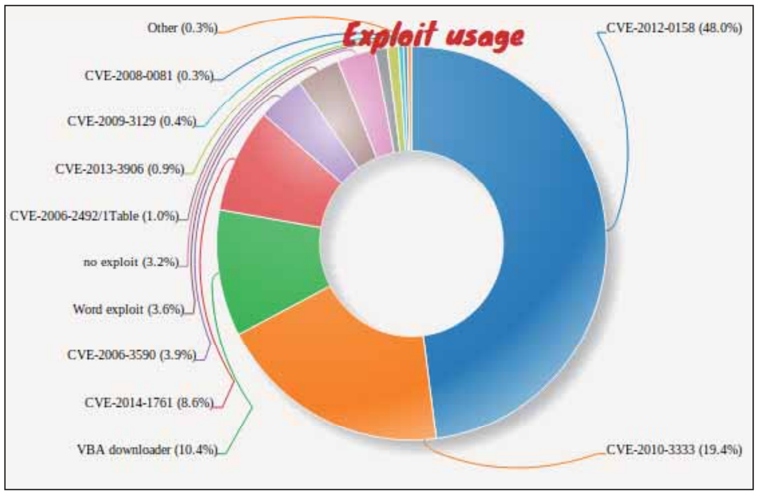


*Figure 2: Distribution of malicious document infection vectors.*

*(The 'Word exploit' category is a medley of exploited documents where we do not identify the exact exploits – however, in the majority of cases the exploit in use is CVE-2012-0158. The 'no exploit' category represents the same Zeus dropper RTF embedded executables, which require the user to double-click on the embedded object, as were mentioned in our 2014 Q1 trends report [1].)*

## SOCIAL ENGINEERING TO THE RESCUE

Visual Basic for Applications (VBA) is the macro programming environment of the *Microsoft Office* suite. Unlike the BASIC language, which is commonly perceived as a tool for beginners, VBA is quite a capable programming environment, as has been well demonstrated by macro viruses in their prime. However, the new wave of trojans described in this analysis doesn't stretch the limits of the language, and in fact are very simplistic.

There is a complication that arises from using a VBA macro instead of an exploit to activate the infection process. In all *Office* suites starting from *Office 2007*, the execution of VBA macros is disabled by default. Consequently, the VBA code will not execute in newer versions of *Office*. Furthermore, the user is warned on the *Word* menu bar about the fact that macros have been disabled, as shown in Figure 3.

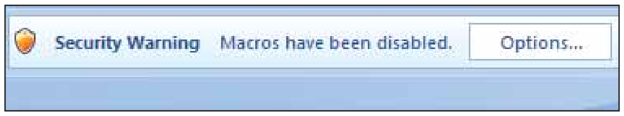

Security Warning Macros have been disabled.

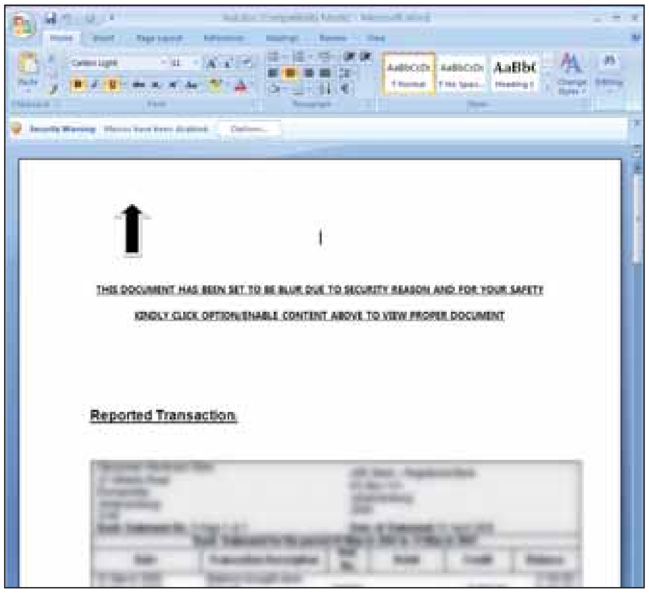
*Figure 3: Word security warning.*

But the malware authors were prepared for this obstacle, and overcame it by deploying simple social engineering tricks.

They prepared the content of the documents in such a way that it would lure the recipient into enabling the execution of macros, and thus open the door for infection.

The malware authors have been busy – since the first appearance of this group of malware at the end of January 2014, at least 75 different variants have appeared. A wide variety of document content has been used – a few examples of which are described in the following paragraphs.

The most peculiar (and one that resembles the social engineering technique used in a Napolar distribution campaign at the end of 2013 [2]) is one in which a blurred transaction report is placed in the document content, encouraging the user to enable macros in order to access the full content. Conveniently, instructions are provided as to how to enable the macros, including an arrow pointing to the exact location where the user is supposed to click (see Figure 4).



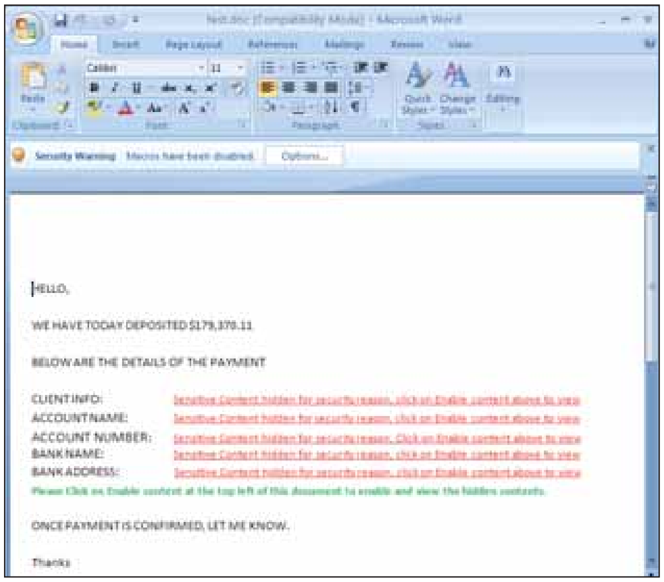
THIS DOCUMENT HAS BEEN SET TO BE BLUR DUE TO SECURITY REASON AND FOR YOUR SAFETY

KINDLY CLICK OPTION/ENABLE CONTENT ABOVE TO VIEW PROPER DOCUMENT

Reported Transaction

*Figure 4: An arrow indicates where to click.*

A different variation of the same approach with less fancy graphical content is shown in Figure 5.



HELLO,

WE HAVE TODAY DEPOSITED $179,370.11

BELOW ARE THE DETAILS OF THE PAYMENT

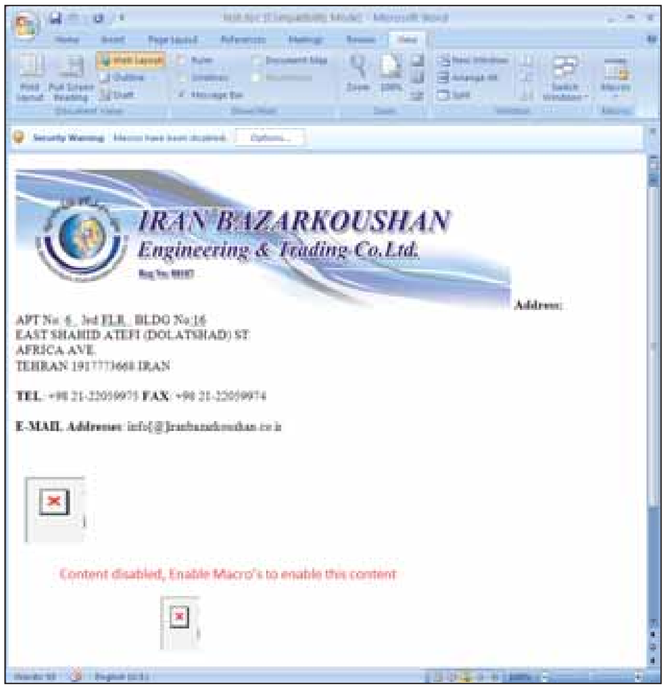
CLIENT INFO: Sensitive Content hidden for security reason, click on Enable content above to view
ACCOUNT NAME: Sensitive Content hidden for security reason, click on Enable content above to view
ACCOUNT NUMBER: Sensitive Content hidden for security reason, Click on Enable content above to view
BANK NAME: Sensitive Content hidden for security reason, click on Enable content above to view
BANK ADDRESS: Sensitive Content hidden for security reason, click on Enable content above to view
Please Click on Enable content at the top left of this document to enable and view the hidden contents.

ONCE PAYMENT IS CONFIRMED, LET ME KNOW.

Thanks

*Figure 5: Less fancy graphical content.*

In other cases, content is marked as confidential and the user is encouraged to enable macros in order to view it, as shown in Figure 6.



IRAN BAZARKOUSHAN
Engineering & Trading Co.Ltd.

Address:
APT No 6, 3rd FLR, BLDG No 16
EAST SHAHID ATEFI (DOLATSHAD) ST
AFRICA AVE
TEHRAN 1917773668 IRAN

**TEL**: +98 21-22059975 **FAX**: +98 21-22059974

**E-MAIL Addresses**: info[@]iranbazarkoushan.co.ir

Content disabled, Enable Macro's to enable this content

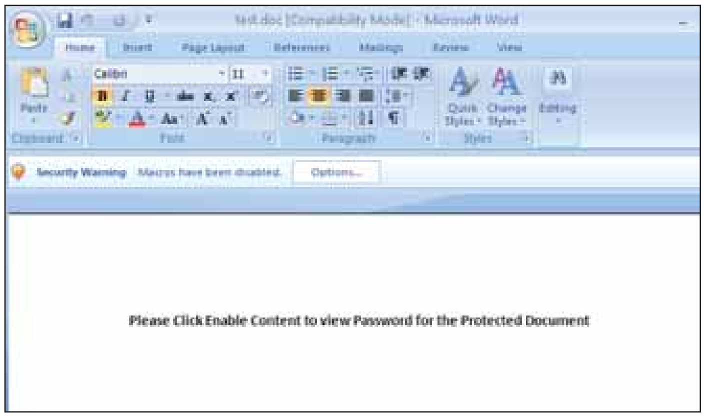
*Figure 6: Again, the user is instructed to enable macros.*

Some examples are more minimalist in design, just giving out basic information, as shown in Figure 7.

A few of the samples we encountered were rather esoteric and vague, building upon the possibility that the receiver of the document will be as clueless about the point of the message as I was while reading it, and enable the macros purely through curiosity (see Figure 8).

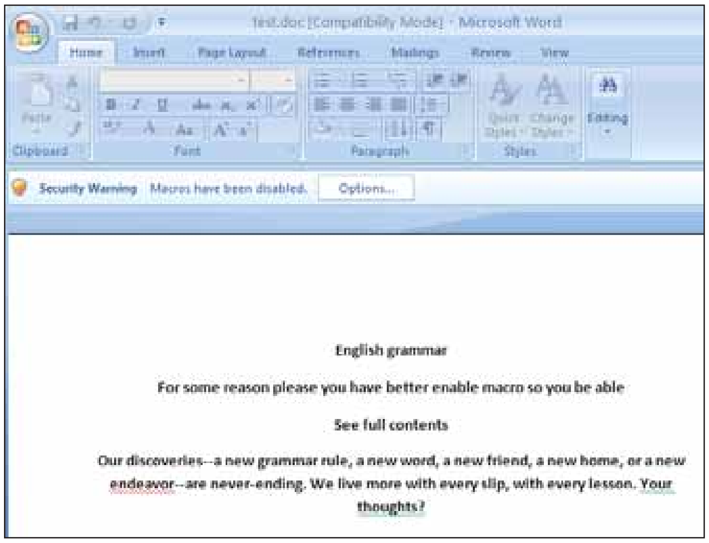
Finally, there were cases where the malware authors left everything to the user's imagination, giving no hints whatsoever (see Figure 9).

Regardless of the document content, the result is always the same: when the user is convinced to enable macros, the malware is ready to run, and on reopening the document, the VBA code will execute.



Please Click Enable Content to view Password for the Protected Document

Figure 7: More of a minimalist design.


test.doc [Compatibility Mode] - Microsoft Word
Security Warning
Macros have been disabled.
Options...

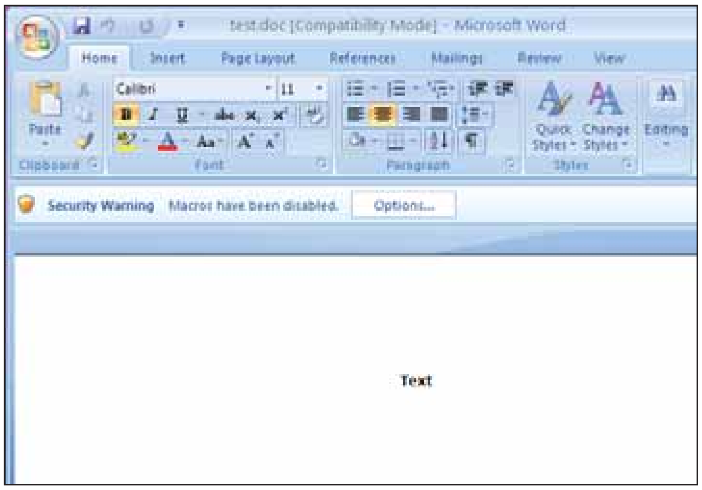
English grammar

For some reason please you have better enable macro so you be able

See full contents

Our discoveries--a new grammar rule, a new word, a new friend, a new home, or a new endeavor--are never-ending. We live more with every slip, with every lesson. Your thoughts?

Figure 8: Hoping the receiver will enable macros through curiosity.


test.doc [Compatibility Mode] - Microsoft Word
Security Warning
Macros have been disabled.
Options...

Text

Figure 9: No clue is given at all.

## MULTI-PLATFORM DOWNLOADER CODE

The macro code, designed for automatic execution on opening, has the following structure (the order in which the individual functions appear and the name of the main function varies in the different variants):

```
Sub Auto_Open()
        main_code()
End Sub
Sub main_code()
...
End Sub
Sub AutoOpen()
        Auto_Open
End Sub
Sub Workbook_Open()
        Auto_Open
End Sub
```

The main code is either in or called from the Auto_Open() function (which is executed when *Excel* is started). It is also invoked by the two other event handler functions, AutoOpen (which is invoked when a *Word* document is opened or *Word* is started) and Workbook_Open (which is invoked when an *Excel* workbook is opened).

### Automacros

*Microsoft Office* programs provided a couple of ways a programmer could automatically execute macros when a specific event occurred. Some of them were tied to menu commands, while the automacros were connected to global application events. If the document contained macro procedures that were using one of the predefined, special names, these procedures were called by the *Office* application when the specific event occurred.

In *Microsoft Word*, these events were tied to starting the *Word* application (the event could be captured with a macro procedure named AutoExec), exiting *Word* (AutoExit), opening a document (AutoOpen), closing a document (AutoClose), or creating a new document (AutoNew).

*Microsoft Excel* had a wider selection of automatic macros, but included similar functions, such as starting *Excel* (Auto_Open), exiting *Excel* (Auto_Close), opening a workbook (Workbook_Open) and closing a workbook (Workbook_Close).

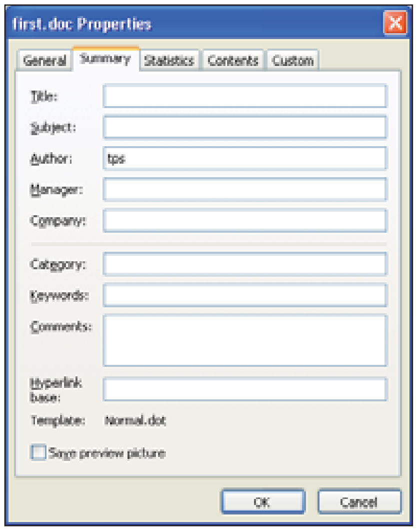
The structure of the trojan's macro code ensures that the code is executed whenever the document is opened. Even though the code itself is cross-application, and Workbook_Open and Auto_Open could make it work in *Excel*, we have not observed any *Excel* workbooks in the distribution campaigns, only *Word* documents. Still, the dual structure covering both *Word* and *Excel* remains in the code – probably because the malware authors were too lazy to clean up the proof-of-concept code they used as 'inspiration' – exactly as we observed in last year's Napolar campaign [2].

Although all of the samples we encountered followed the simple code template described above, we could distinguish two different strains among them.

The most common strain first called the UrlDownloadToFileA *Windows* function to download the final payload from a hard-coded URL, then saved it either to the %TEMP% folder or the %APPDATA% folder, and finally ran it using the Shell function. The dropped executable was usually registered for automatic execution during system start-up in one of the registry autorun locations, such as HKCU\Software\Microsoft\Windows\CurrentVersion\Run.

Of the 74 documents we identified as belonging to the VBA downloader campaigns (at the time of writing this paper), the vast majority (60) belonged to this group.

On looking more deeply into the *Word* documents, some interesting characteristics could be observed. The metadata for each document includes document creation/modification times, revision number, and the author of the document. Looking into these properties revealed that the different documents were all attributed to the same author name, 'tps'. It is reasonable to assume that the largest number of malicious documents come from the same source.



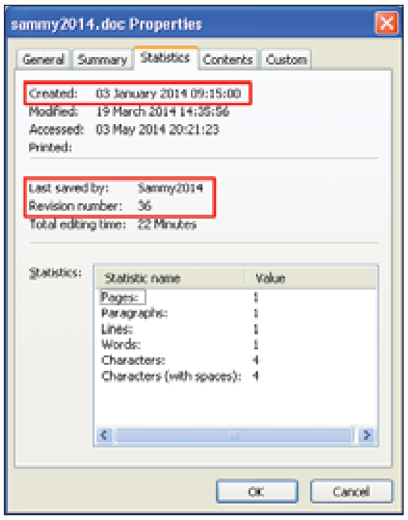
| | |
|---|---|
| Title: | |
| Subject: | |
| Author: | tps |
| Manager: | |
| Company: | |
| Category: | |
| Keywords: | |
| Comments: | |
| Hyperlink base: | |
| Template: | Normal.dot |

☐ Save preview picture

*Figure 10: Looking at the file properties reveals the same author name.*

Another useful piece of information is the name of the last user to have saved the document. This revealed some interesting names. Some just indicated the malware author's favourite gadgets ('DELL XPS', 'Xpera Z'), while others gave a hint as to the author's name or nickname ('Sammy Sam', 'Sammy2014', 'samy14' and 'samy2014').

Even more interesting is the fact that the creation date (highlighted in Figures 11 and 12) was the same for most of these documents, as shown in Figure 13.



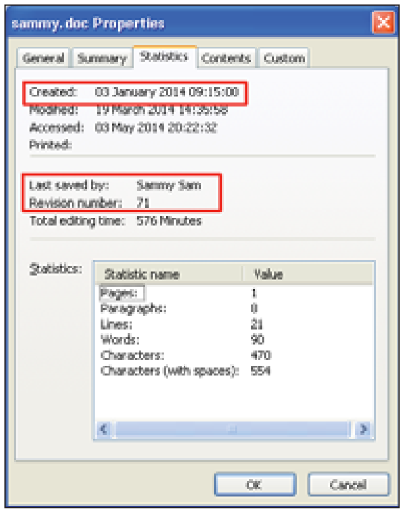
Created: 03 January 2014 09:15:00
Modified: 19 March 2014 14:35:56
Accessed: 03 May 2014 20:21:23
Printed:

Last saved by: Sammy2014
Revision number: 36
Total editing time: 22 Minutes

Statistics:

| Statistic name | Value |
|---|---|
| Pages: | 1 |
| Paragraphs: | 1 |
| Lines: | 1 |
| Words: | 1 |
| Characters: | 4 |
| Characters (with spaces): | 4 |

*Figure 11: File last saved by 'Sammy2014'.*



Created: 03 January 2014 09:15:00
Modified: 19 March 2014 14:35:58
Accessed: 03 May 2014 20:22:32
Printed:

Last saved by: Sammy Sam
Revision number: 71
Total editing time: 576 Minutes

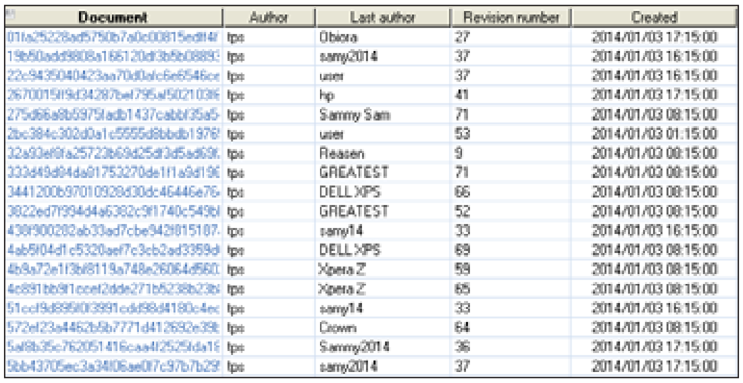
Statistics:

| Statistic name | Value |
|---|---|
| Pages: | 1 |
| Paragraphs: | 0 |
| Lines: | 21 |
| Words: | 90 |
| Characters: | 470 |
| Characters (with spaces): | 554 |

*Figure 12: File last saved by 'Sammy Sam'.*

This suggests that the development of this VBA trojan family started on 3 January, and the author didn't bother to create a new document each time a new variant was created, but simply kept modifying the previous version by adding new social engineering content and occasionally changing the macro code – the increasing revision number also supports this assumption. Despite the fact that the last-saved-by name differs within the group, the creator is always 'tps' – this is another indication that this group of malware is derived from the same root document. The development of this strain appears to be an ongoing project, with the most recent version at revision number 91.

The rest of the 74 documents are likely to have been developed independently from the previous group. They use a different method, reminiscent of the VBScript downloaders



| Document | Author | Last author | Revision number | Created |
|---|---|---|---|---|
| 01fa25228ad5750b7a0c00815edf4f | tps | Obiora | 27 | 2014/01/03 17:15:00 |
| 19b50add9808a166120df3b5b08893 | tps | samy2014 | 37 | 2014/01/03 16:15:00 |
| 22c9435040423aa70d0afc6e6546ce | tps | user | 37 | 2014/01/03 16:15:00 |
| 2670015ff9d34287bef795af5021036 | tps | hp | 41 | 2014/01/03 17:15:00 |
| 275d66a8b5975fadb1437cabbf35a5- | tps | Sammy Sam | 71 | 2014/01/03 08:15:00 |
| 2bc384c302d0a1c5555d8bbdb1976! | tps | user | 53 | 2014/01/03 01:15:00 |
| 32a93ef0fa25723b69d25df3d5ad69f. | tps | Reasen | 9 | 2014/01/03 00:15:00 |
| 333d49d94da91753270de1f1a9d196 | tps | GREATEST | 71 | 2014/01/03 00:15:00 |
| 3441200b97010928d30dc46446e76- | tps | DELL XPS | 66 | 2014/01/03 00:15:00 |
| 3822ed7f994d4a6382c9f1740c549bl | tps | GREATEST | 52 | 2014/01/03 00:15:00 |
| 438f900282ab33ad7cbe942f015107- | tps | samy14 | 33 | 2014/01/03 16:15:00 |
| 4ab5f04d1c5320aef7c3cb2ad3359d | tps | DELL XPS | 69 | 2014/01/03 08:15:00 |
| 4b9a72e1f3bf8119a748e26064d560 | tps | Xpera Z | 59 | 2014/01/03 08:15:00 |
| 4c891bb9f1ccef2dde271b5238b23b | tps | Xpera Z | 65 | 2014/01/03 08:15:00 |
| 51ccf9d8958f03991cdd98d4180c4ec | tps | samy14 | 33 | 2014/01/03 16:15:00 |
| 572ef23a4462b5b7771d412692e39 | tps | Crown | 64 | 2014/01/03 08:15:00 |
| 5af8b35c762051416caa4f2525fda1 | tps | Sammy2014 | 36 | 2014/01/03 17:15:00 |
| 5bb43705ec3a34f06ae0f7c97b7b29 | tps | samy2014 | 37 | 2014/01/03 17:15:00 |

*Figure 13: The creation date is the same for most of the documents.*

from a decade ago. The remote file from the hard-coded URL is downloaded using the MSXML2.XMLHTTP object, and then the downloaded content is saved to the %APPDATA% folder or %USERPROFILE% folder, and finally executed using the Shell function.

In this strain of VBA downloaders, in each case, the author and the last-saved-by name are the same, and the creation and last-saved dates are roughly the same. This suggests a different mode of operation: rather than the same template document being modified over and over again, new ones were created most of the time.

## FINAL PAYLOAD

The downloaded payloads demonstrated a wide range of ordinary crimeware. We observed:

- Zeus-related AutoIt executables:

  https://www.virustotal.com/en/file/ace7db9b08d65b2c4f0c011a54571039e45ca4010aaf482c73ee7ef860019d8c/analysis/

  https://www.virustotal.com/en/file/8b6757271611fe6ff0f758bffee20e488d19b583be0352584f191d21d241bcfd/analysis/

  https://www.virustotal.com/en/file/c85fe4874ddfa9684569819db42fb321641088cbe15e6e54d28de49de26f155c/analysis/

- A DotNET injector:

  https://www.virustotal.com/en/file/f1b0432594bf9651ad50d755fe1967fd2a16f112e5c0df6ebc5311166857d30c/analysis/

- Old-fashioned RATs like Simda:

  https://www.virustotal.com/en/file/f456df5aa1af278ccd9958bf7f0b18ab36ed05ccb73aed8d7faec8a5898d8bcb/analysis/.

## CONCLUSION

Our earlier report pointed out [1] that ordinary cybercriminals turned to *Office* exploits as an alternative delivery method for their creations. Current trends show that they have moved one step further into the *Office* realm: they have discovered the long-forgotten VBA macros and added them to their repertoire. This emphasizes the fact (I can't even count how many times it has been proven in the past) that there is no need for fancy exploitation. When the aim is to infect a large number of users, good old social engineering never fails to deliver the results.

Finally, a piece of advice: there is no justification as to why the content of a document can only be displayed properly if the execution of macros is enabled. If you receive a document with this advice, be aware: you are probably being attacked.

## REFERENCES

[1] http://nakedsecurity.sophos.com/advanced-persistent-threats-the-new-normal.

[2] http://www.virusbtn.com/virusbulletin/archive/2014/04/vb201404-VBA.

**Editor:** Martijn Grooten
**Chief of Operations:** John Hawes
**Security Test Engineers:** Scott James, Tony Oliveira
**Sales Executive:** Allison Sketchley
**Editorial Assistant:** Helen Martin
**Perl Developer:** Tom Gracey
**Consultant Technical Editors:** Dr Morton Swimmer, Ian Whalley

**© 2014 Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England.**
**Tel: +44 (0)1235 555139. Fax: +44 (0)1865 543153**
**Email: editorial@virusbtn.com**
**Web: http://www.virusbtn.com/**