

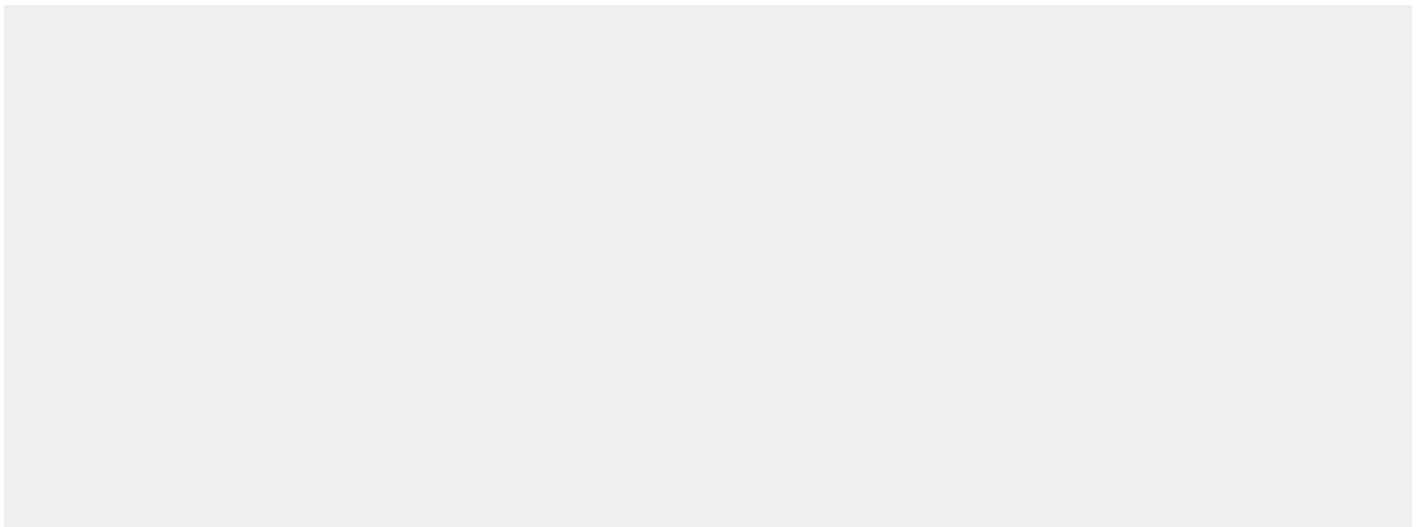
00180: WebApp Attack PCAP Analysis

Objective

In this lab you will analyze a capture file of a web application attack in order to identify the attack vector and deduce the vulnerability the attack exploited.

Scenario

Use Wireshark to conduct analysis of a web application based malware campaign
Understand how to identify and analyze web application infection chain
Analyze web application malware artifacts




EITest Malware Campaign

Scenario


We will analyze the web traffic from the "EITest" Malware campaign that delivered web application based malware in the form of the Angler Exploit Kit.

This malware campaign is characterized by compromised websites aka "web apps" that would either deliver the exploit or redirect the victim to a page hosted by the attacker for the drive-by download. The exploit kits can vary but this campaign focused on delivering the Angler EK, and Flash exploits.

- ☐ 1. Log into the system using the username user and the password password.


 This lab is equipped with an auto-scoring technology designed to track your progress as you successfully complete the lab. A short-cut is available on the Kali Linux VM Desktop to check your score. Double-click on the short-cut and a web browser window will open displaying your current score. You can periodically refresh the page as you complete more of the lab to see your current score.

The final score is calculated and recorded when you either complete or cancel the lab. If you save your lab, the score is held until you resume the lab and cancel or complete it.

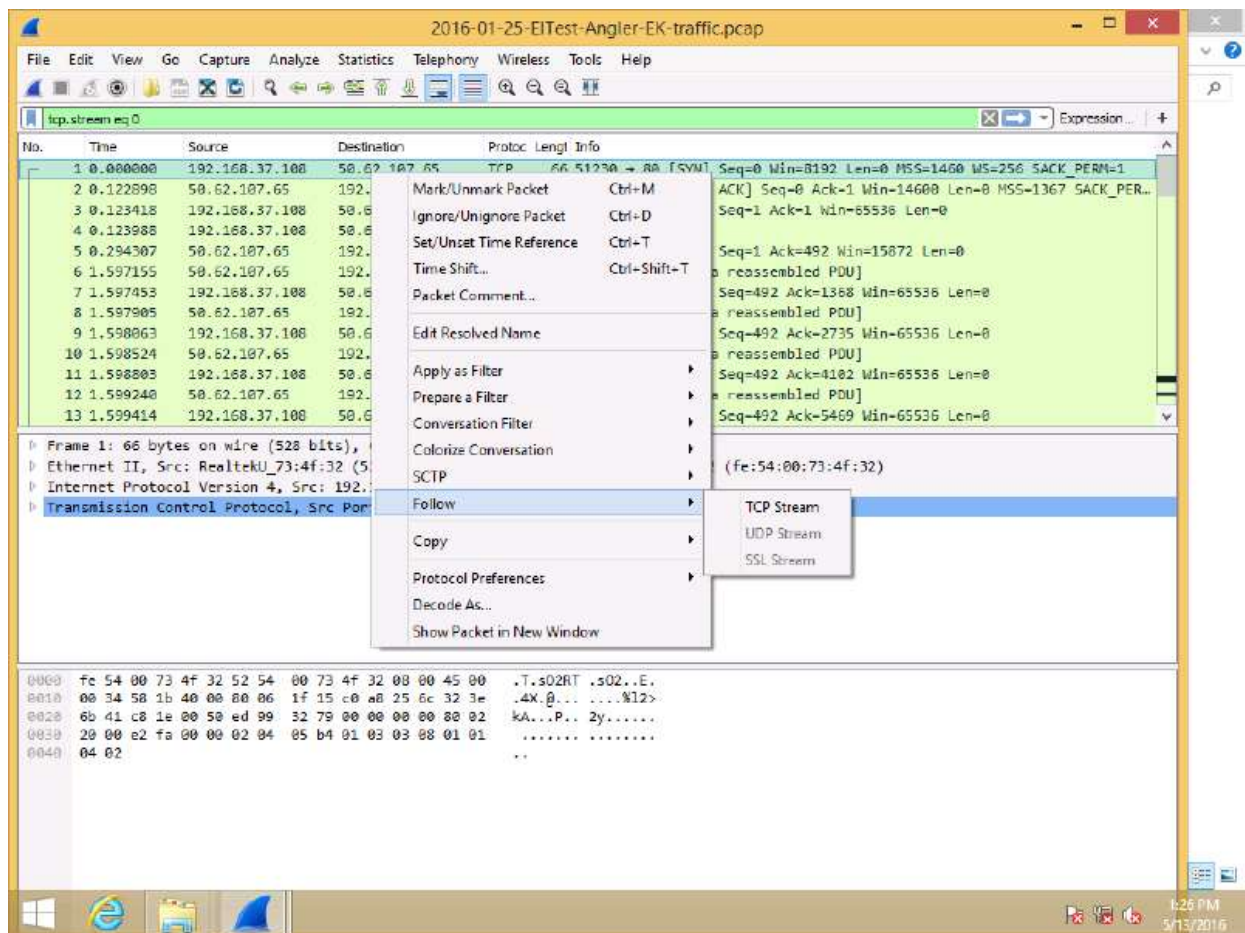
 Login credentials are also in the machines tab with the individual VM.

- ☐ 2. Browse to the Folder on the Desktop called pcaps and open the capture file.

Capture file courtesy of www.malware-traffic-analysis.net


 For more information on this web app exploit focused malware campaign, you can visit <https://blog.malwarebytes.org/threat-analysis/2014/10/exposing-the-flash-eitest-malware-campaign/>

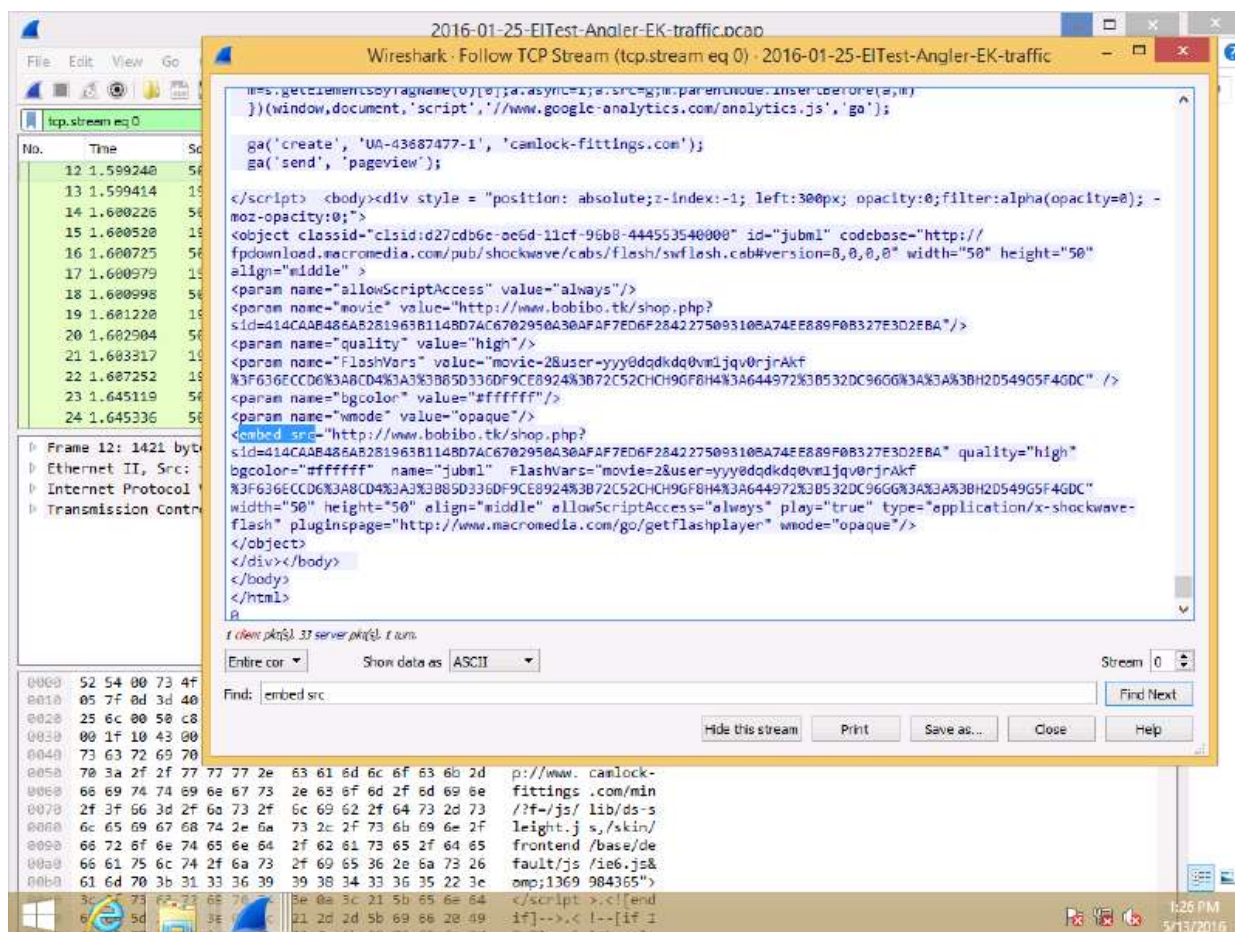
- ☐ 3. Follow the TCP stream and reconstruct the session. Click on the first packet, right click and select Follow TCP stream. A new window will pop up that will provide a more user-friendly view of the session for analyzation.



- ☐ 4. After we reconstruct the session, do a search for some html tags of interest. You may need to resize the window to see the "Find" search function at the bottom.

Once you've found the "Find" function, type in "src" and press enter to briefly look at all the source tags for anything of interest. This search will also include any lines with the string "embed src" and "script src" as well, which are definitely of interest.

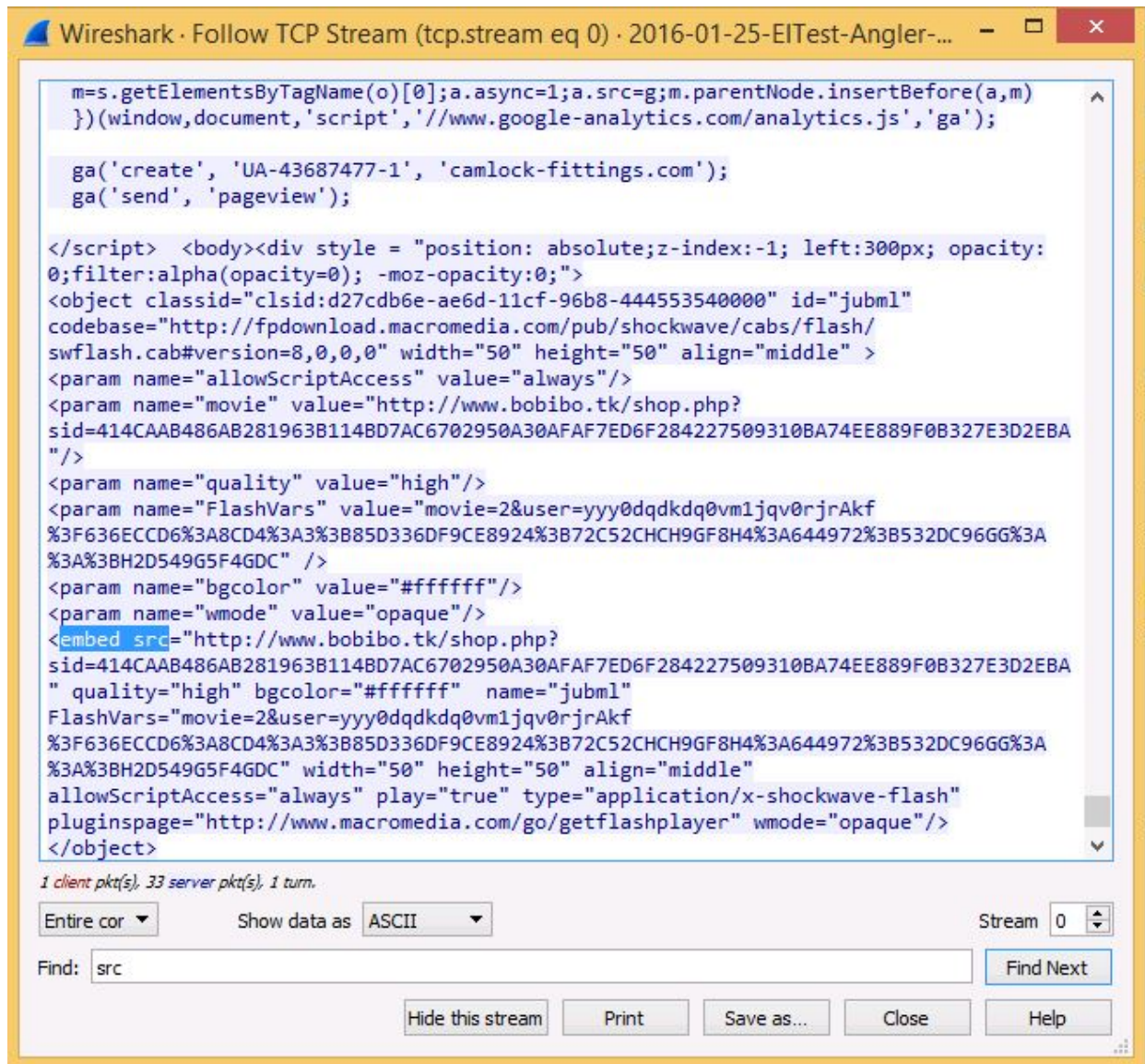
 Anytime a <script> tag is seen in the HTML, the Javascript interpreter gets invoked. Of interest to us are the HTML tags that allow external content to be embedded on the page we are analyzing. Specifically, we are interested in the <script src=...> and the <embed src=...> tags. The latter allows for the indirect introduction of scripts into the HTML which will then be interpreted by the Javascript interpreter.



- ☐ 5. When we look at the embedded source value we can see it's website called bobibo.tk/shop.php? sid= <long number>

If we continue further on with the source parameters, we find the name value to be "jubml" which is a known IOC for the ELTest malware campaign. This value was consistently set to ELTest for most of 2014 and 2015 but recently switched to "jubml."

At this point, we have identified that the website/webserver in question (www.camlock-fittings.com) has been compromised has some embedded source tags known to be associated with the ELTest malware campaign.

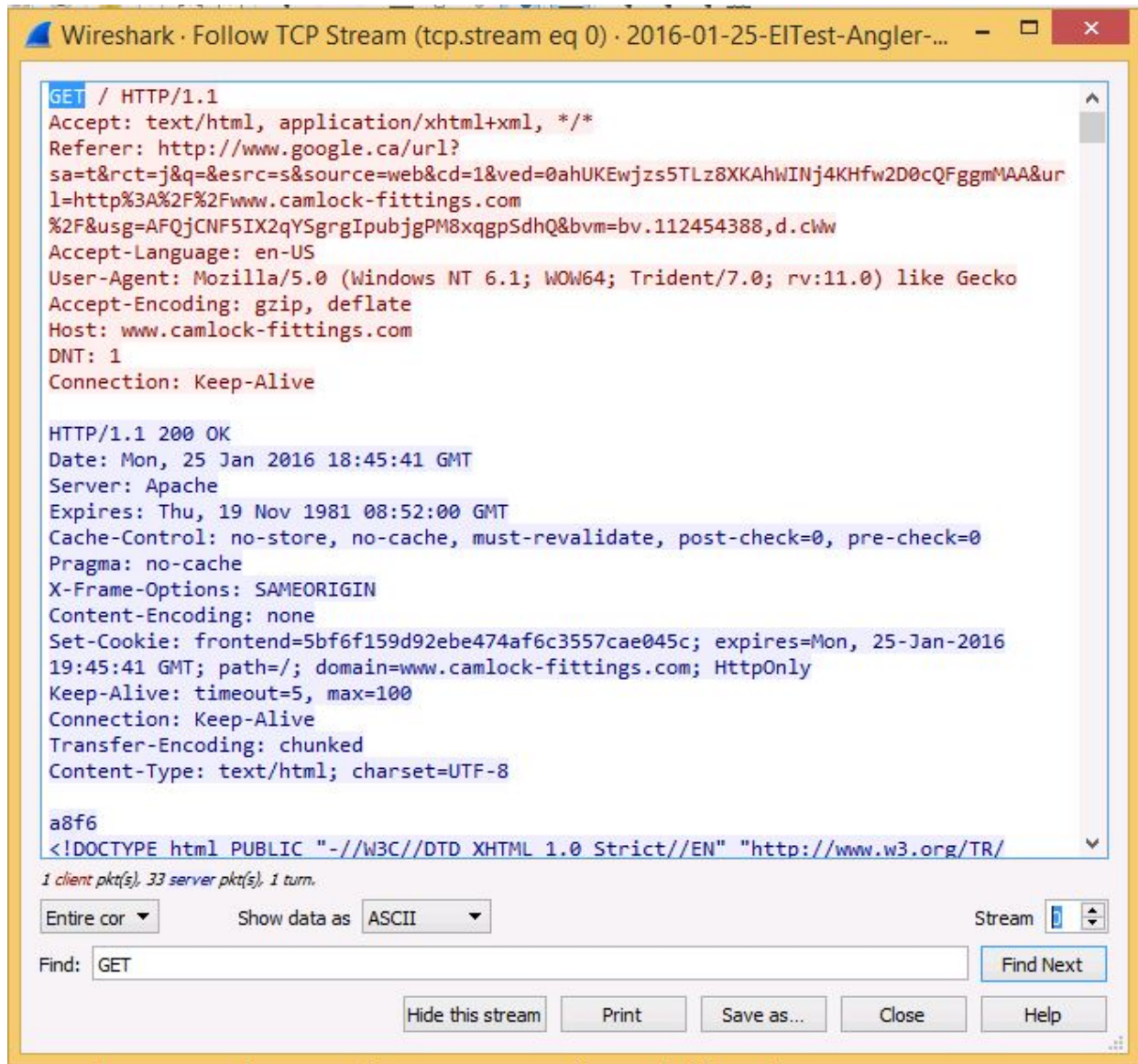


- ☐ 6. We know the site has been compromised. Look at the web traffic GET requests to see what happens after someone browses to the compromised site. In the Find box type "GET".

So the first hit on our search criteria "GET" returns the first packet in the stream. We see the referrer site is google and the GET request is sent to the www.camlock-fittings.com site.

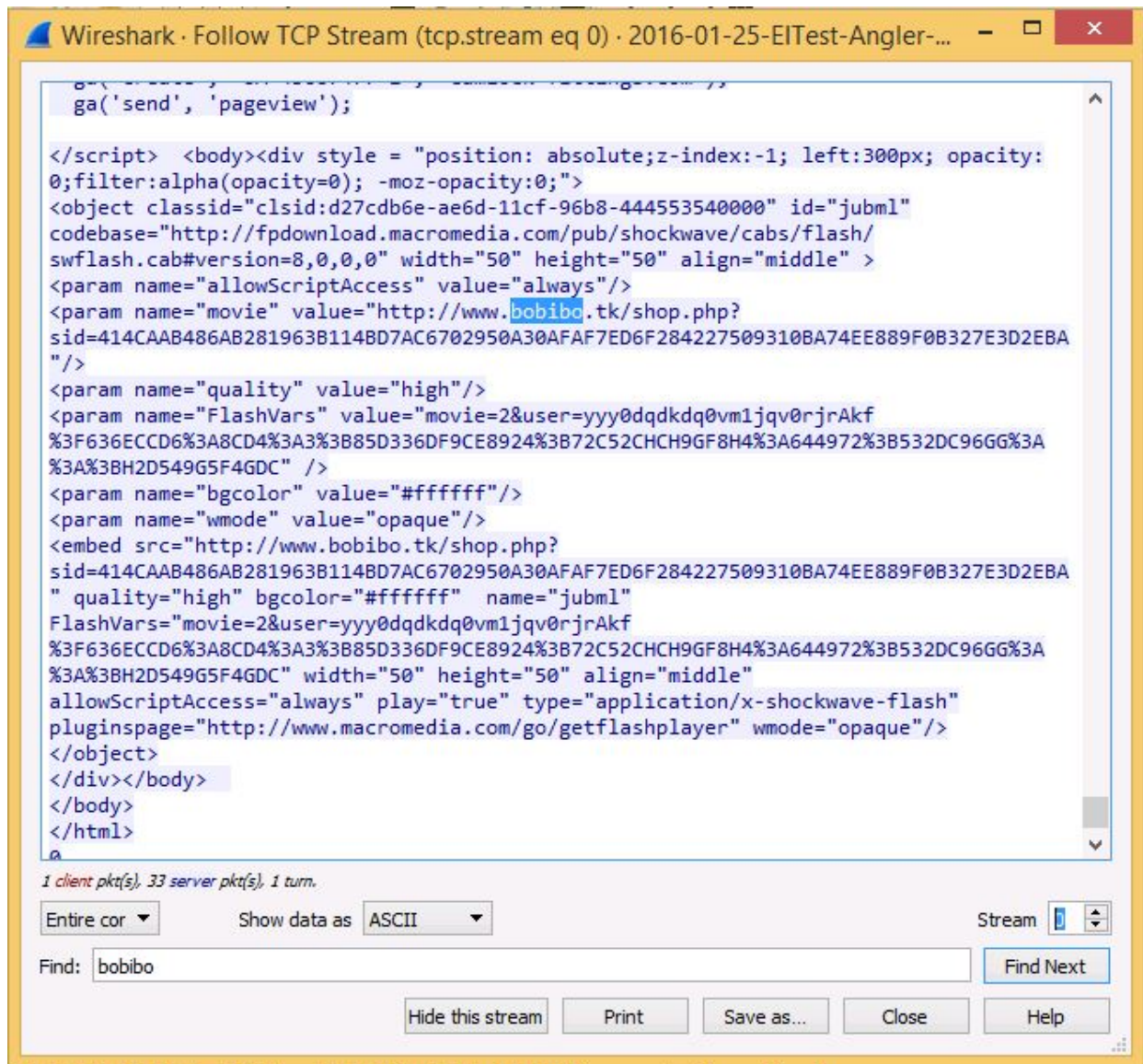
See the Server Status code "200" ok returned and then a bunch of HTML pushed to the client. None of this is strange.

In the next step, we'll look for a possible redirect from the compromised page to the malicious page that is hosting the malware.



- ☐ 7. In the Find box, search for the website we found embedded in the page earlier. Type in "bobibo" and look for any instance of URLs with this domain name in them.

Near the bottom of the reconstructed session, we see the "embed src" tag again. This is the malicious redirect to the malware-hosting site.

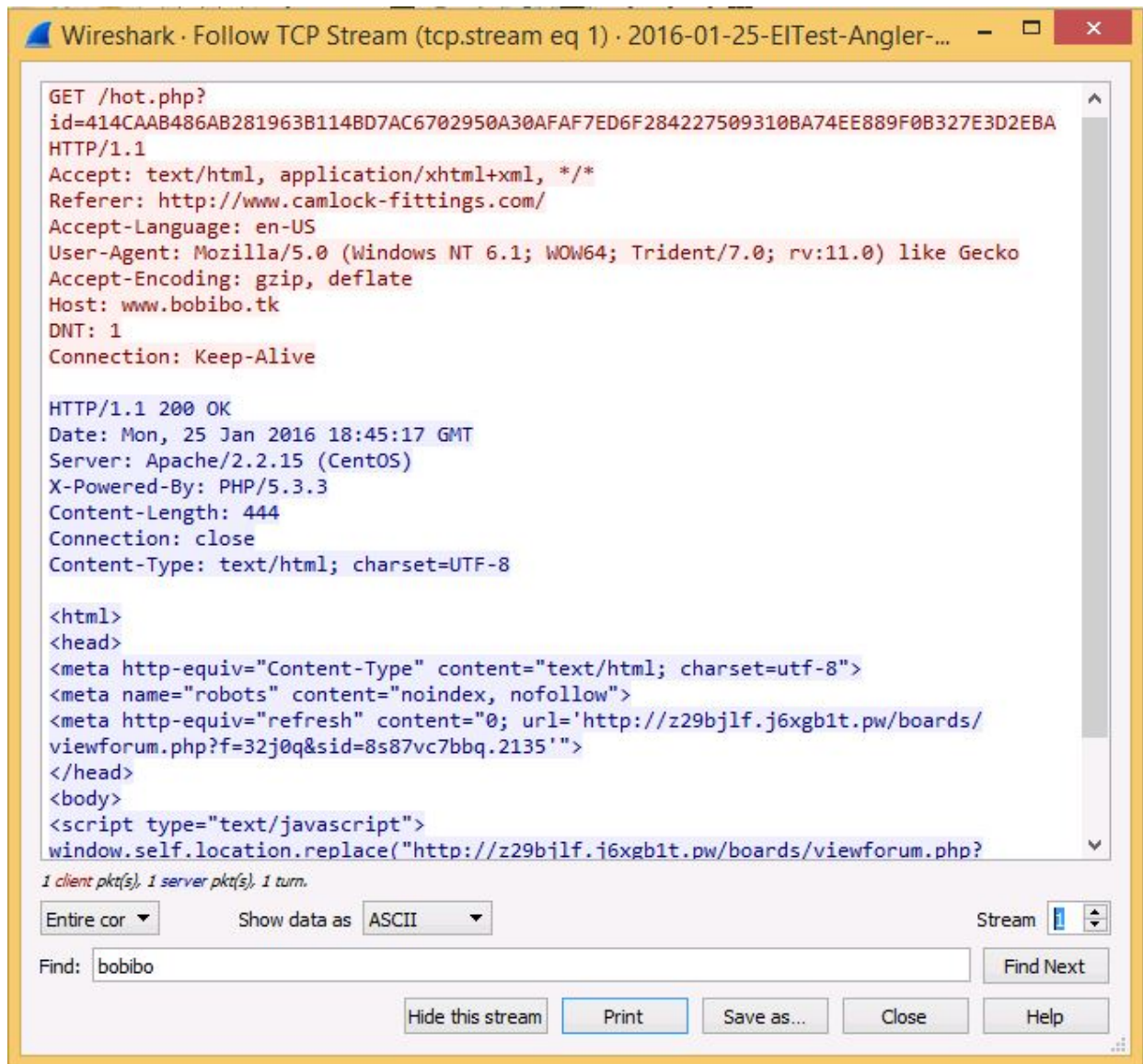


- ☐ 8. Near the bottom right of the windows we've been working in, you'll see the word "Stream" and some up and down arrows. Let's look at the next TCP stream in this packet capture to see if we can find any other interesting pieces of information.

Click the up arrow to go to Stream 1.

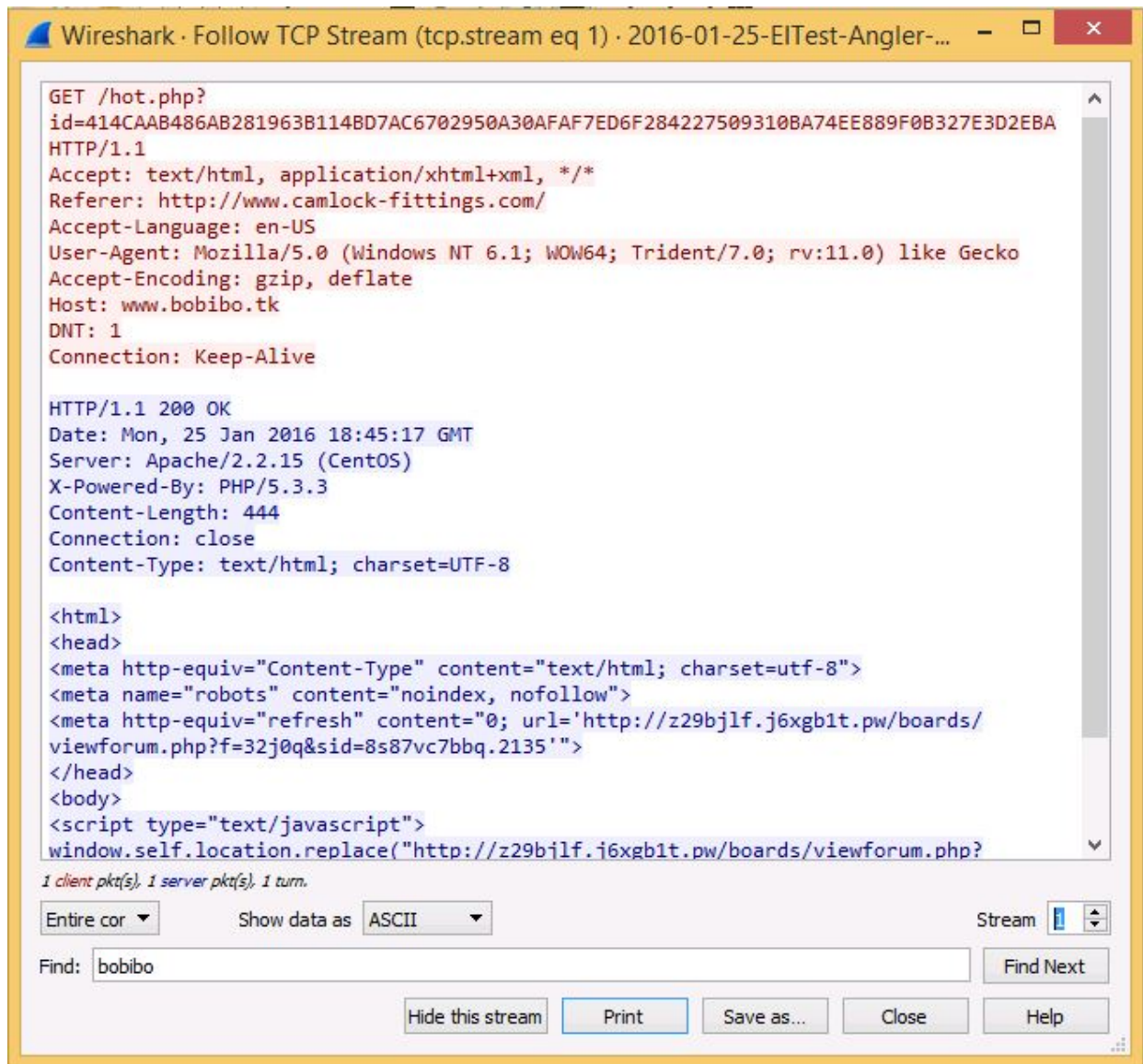
- ☐ 9. The first thing we see in Stream 1 is a GET request to the Bobibo domain's web server to some sort of resource with a long and strange "id" value.

If you look at the Referrer header, you'll see that we came to the bobibo domain via the compromised camlock-fittings page as we suspected.



- ☐ 10. If we begin to look at the HTML code pushed down from the server in response to the GET request something interesting shows up.

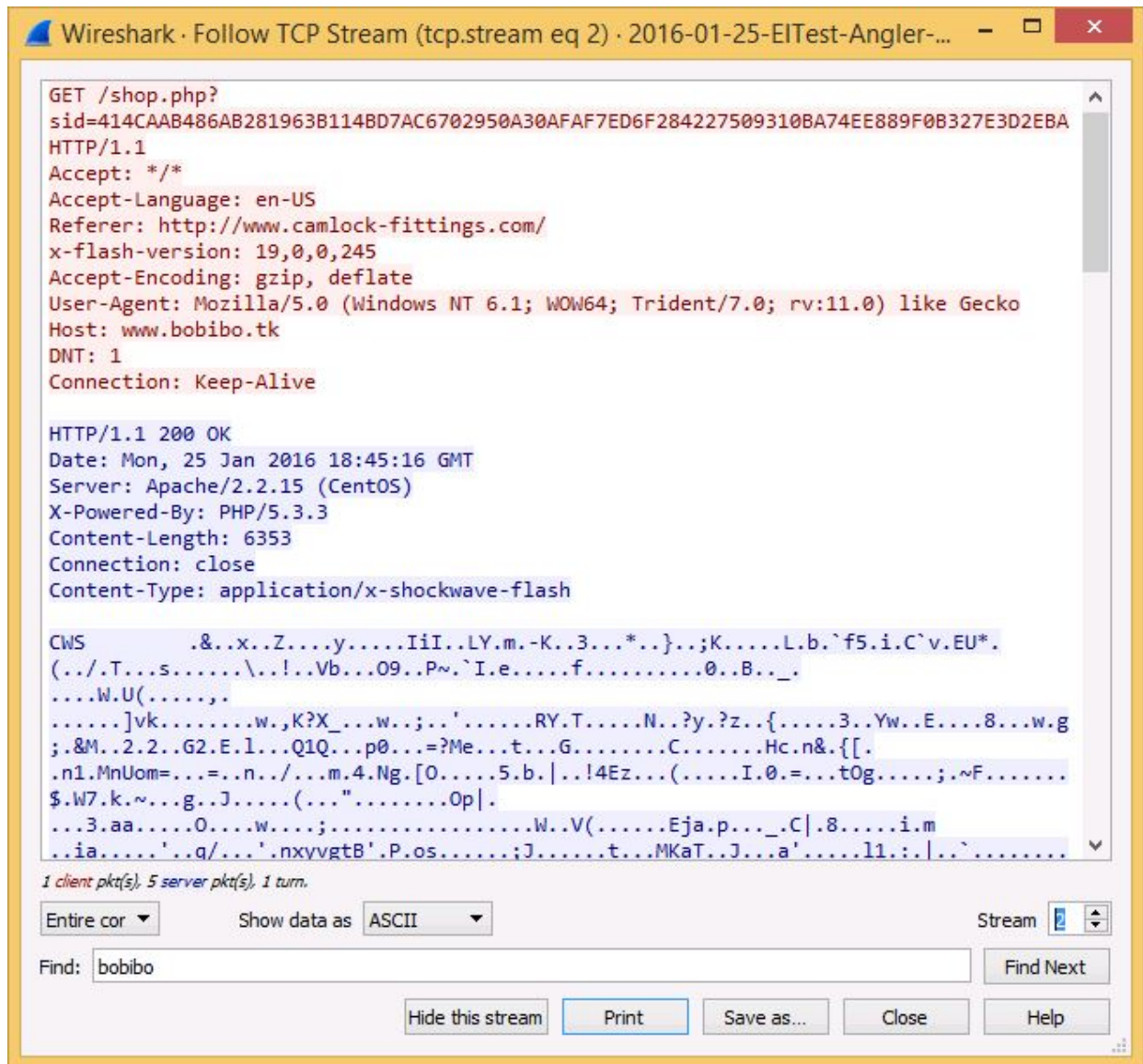
We see a "meta-http-equiv" tag with a refresh value and content of 0 and a URL. The content "0" means wait zero seconds and then go to the URL that's listed. This HTML tag automatically redirects the user to another page. So, we will need to make note of this URL and see exactly where it takes us and what happens there.



- ☐ 11. Continue to look at the next stream, click the up arrow and go to stream 2.

Here we see another GET request to the known malicious site bobibo. This time, the response is that the server pushed down a Flash object. We know this by looking at the header Content-Type.

You can use the Export Objects function in Wireshark to reassemble and export this flash application. It's also found in the pcaps folder on the Desktop in the subfolder called artifacts.

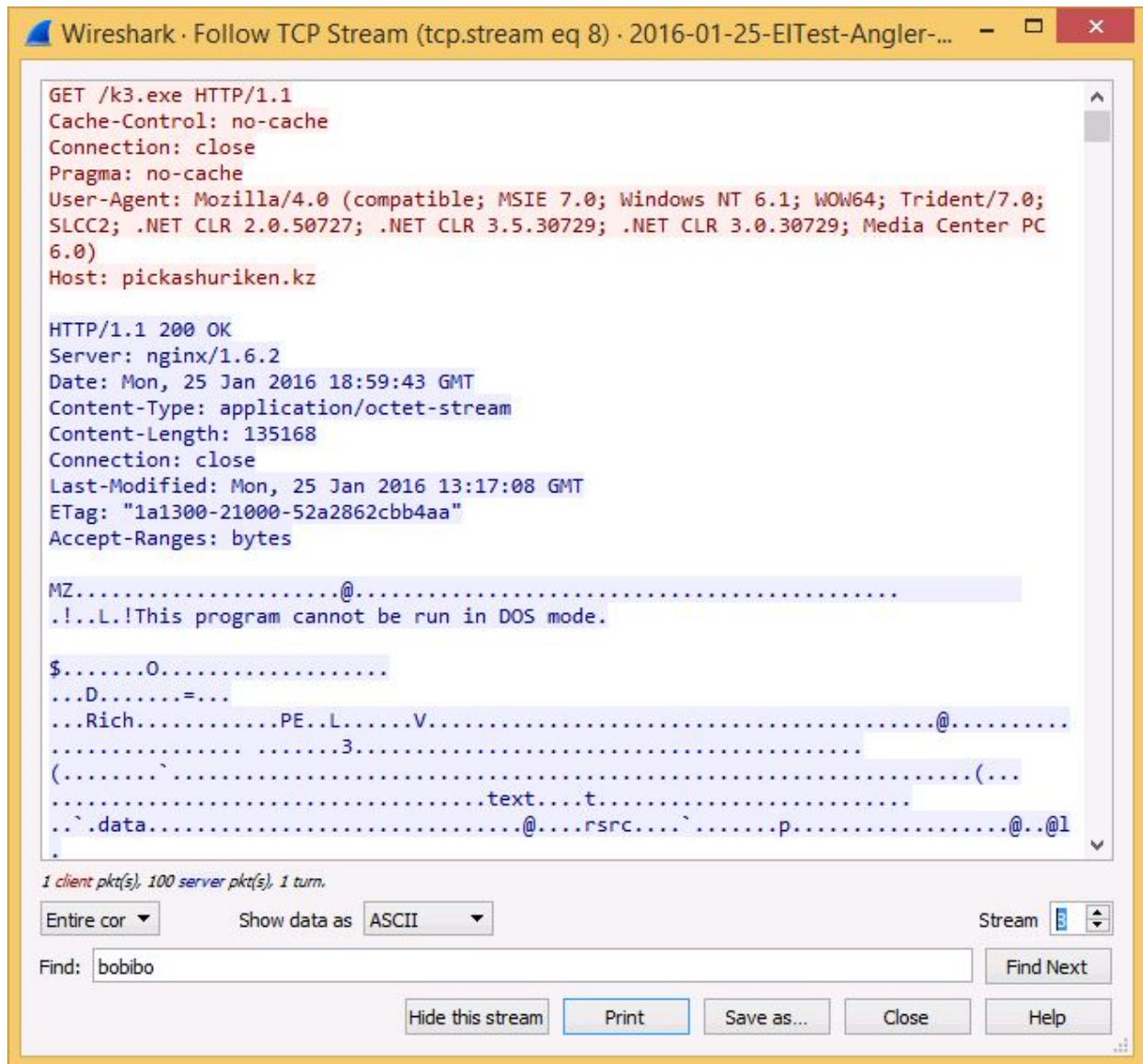


- 12. Go to stream 8. We see a GET request for a file called k3.exe

In the response, we see the typical MZ header indicative of a compiled executable of type PE. While scrolling through the data, we start seeing some indicators of Windows API calls to functions (GetModuleFileNameA) and also some Visual Basic associations.

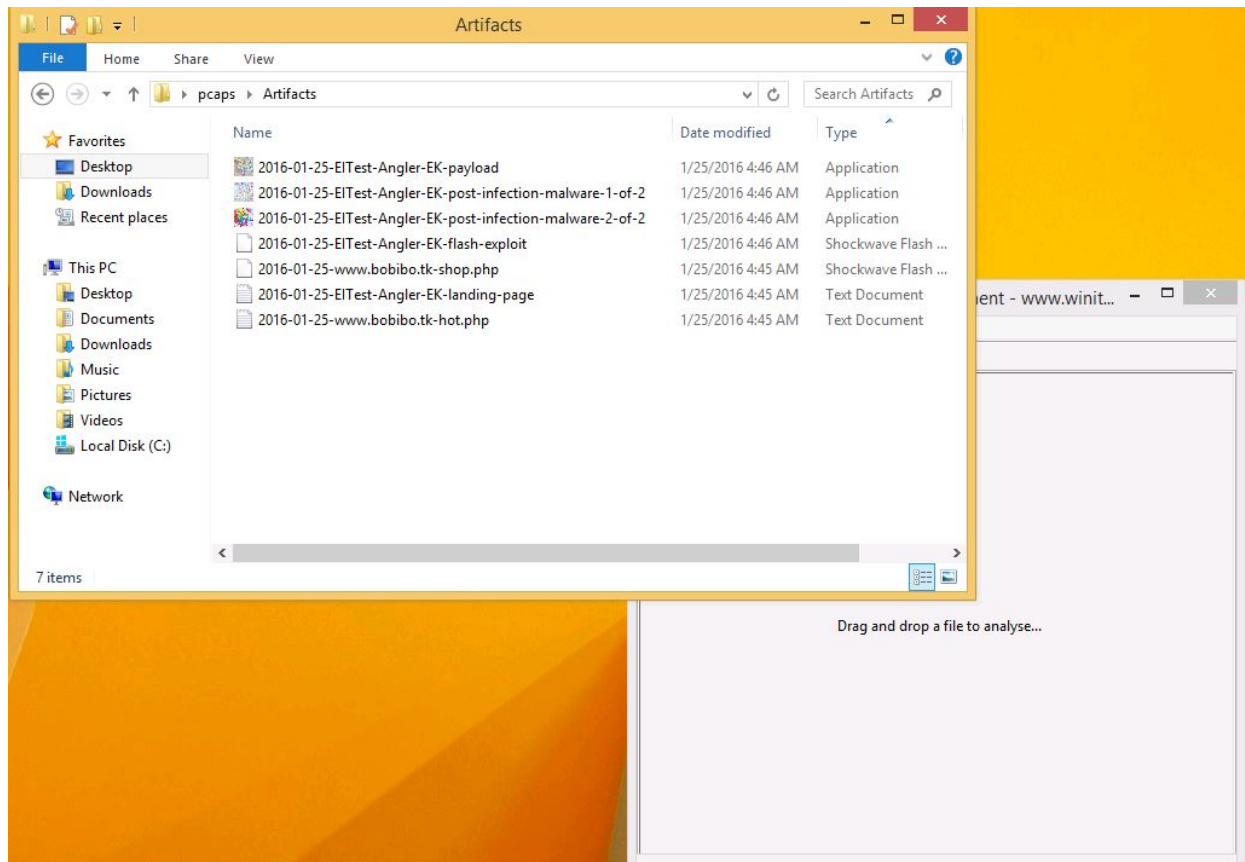
We would need to further analyze this malware in a dynamic and static manner to understand what it does.

⚠️ ***GetModuleFileName retrieves the fully qualified path for the file that contains the specified module. It is frequently used to start services. Malware can call this function as well. The function itself isn't malicious but a piece of malware can call it in order to load a specific module and start a malicious service.



- ☐ 13. Under C:\Program Files you will find several malware analysis tools. Use PESTudio to open the three malware applications in the artifacts folder under pcaps on the Desktop. Look under the "Type" column in the File Explorer Window to find the files of type "Application."

Look at the various Imports, functions and strings for things indicative of file downloads/uploads, connections to the internet, system enumeration etc.



- ☐ 14. Continuing from the previous Task, open the "Malware 1 of 2" application file first in PEStudio. Let's discuss what we see:
- Click on Indicators and note that the file ignores both DEP and ASLR. These two built-in exploit mitigation controls in Windows are important in stopping exploits and malware. While other files also ignore these built-in controls at runtime, this is a small but important piece of information for us during the analysis process.
 - Click on Imported Symbols. Something to consider about this piece of malware is that it is written in Visual Basic. Therefore, we may not see a lot of the common indicators or normal Windows API function calls that we observe on other pieces of malware. Of interest here is the import of `DLLFunctionCall` which allows the Visual Basic script to access C runtime functions from the Windows API.

In the next step, we will continue to look at the output from our PEStudio analysis.



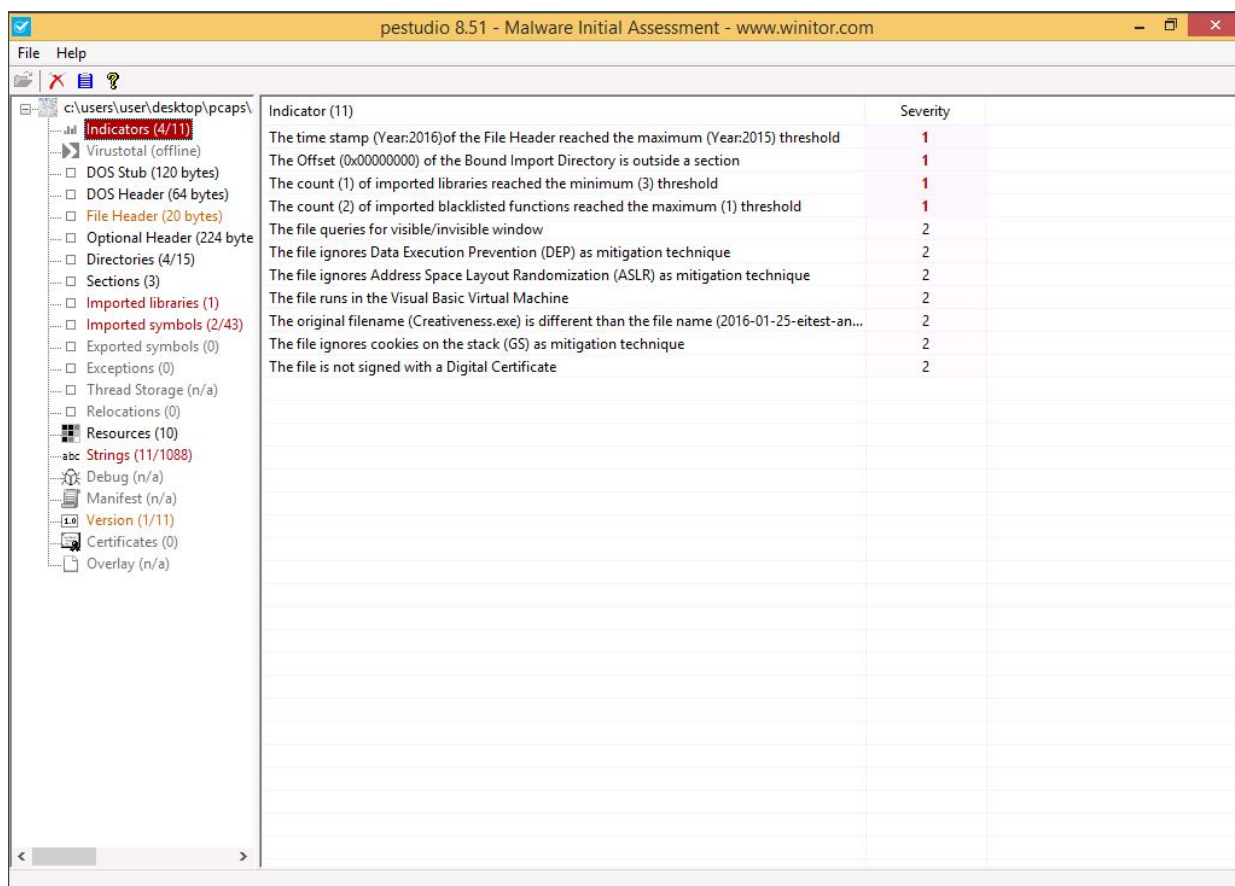
A Visual Basic script might make a DLL function call for the following reasons:

Access to C Run-Time Functions:

The C run-time library has many useful functions that would not be available to Visual Basic programmers were it not for DLLs. For example, the `_dos_getdiskfree` function allows you to calculate the total amount of disk space and the free disk space available on a drive.

Access to Windows API (Application Programming Interface) Functions that Require Callback Routines:

Some Windows API functions require a callback function. A callback function is a function that Windows will call while executing the API call. An example of this sort of function is `EnumTaskWindows`, which will give the handle of all windows that are owned by a particular task.



15. Click on Strings on the left side of PESTudio menu. The file creativeness.exe is a known piece of malware. This malware calls back to an IP address in the Republic of Moldova. Nothing else here is singularly malware related but everything that is blacklisted has been known to be used by malware in contravention to its original intent.

Reference: <https://www.hybrid-analysis.com/sample/f872488c51e5b538fd25740ac9d63091ca04aa6cc2bb0702cbfc78d42de981ad?environmentId=1>

pestudio 8.51 - Malware Initial Assessment - www.winitor.com

	Type	Size	Section...	Blacklisted (11)	Item (1088)
Indicators (4/11)	ascii	5	?0x01DE	x	.rsrc
VirusTotal (offline)	ascii	12	?0x0206	x	MSVBVM60.DLL
DOS Stub (120 bytes)	ascii	12	.text:0x...	x	MSMAPI32.OCX
DOS Header (64 bytes)	ascii	54	.text:0x...	x	C:\Timeslice\Haikal\Tagala\Ozmond5\Qoph\Alise4\VB6.OLB
File Header (20 bytes)	ascii	11	.text:0x...	x	EnumWindows
Optional Header (224 byte)	ascii	17	.text:0x...	x	GetModuleFileName
Directories (4/15)	ascii	8	.text:0x...	x	VBA6.DLL
Sections (3)	ascii	4	.text:0x...	x	.P.H
Imported libraries (1)	ascii	12	.text:0x...	x	MSVBVM60.DLL
Imported symbols (2/43)	ascii	15	.text:0x...	x	DllFunctionCall
Exported symbols (0)	unicode	16	.text:0x...	x	Creativeness.exe
Exceptions (0)	ascii	40	?0x0000	-	!This program cannot be run in DOS mode.
Thread Storage (n/a)	ascii	4	?0x0076	-	Rich
Relocations (0)	ascii	5	?0x00AD	-	.text
Resources (10)	ascii	6	?0x01B6	-	.data
Strings (11/1088)	ascii	6	?0x0245	-	_ ^ ~ SiG
Debug (n/a)	ascii	13	.text:0x...	-	HladeoStavros
Manifest (n/a)	ascii	7	.text:0x...	-	VB5I6&*
Version (1/11)	ascii	12	.text:0x...	-	Creativeness
Certificates (0)	ascii	13	.text:0x...	-	Streetwalker1
Overlay (n/a)	ascii	7	.text:0x...	-	Stavros
	ascii	4	.text:0x...	-	DEST
	ascii	4	.text:0x...	-	DEST
	ascii	18	.text:0x...	-	MSMAPI.MAPISession
	ascii	11	.text:0x...	-	MAPISession
	ascii	4	.text:0x...	-	jbtG
	ascii	10	.text:0x...	-	Palliator4
	ascii	12	.text:0x...	-	Doctrinarily
	ascii	9	.text:0x...	-	Crevised3
	ascii	7	.text:0x...	-	Stavros
	ascii	4	.text:0x...	-	Kk.M

16. Here is a summary of the infection chain for this malware campaign.

- Step 1 – Compromised website with injected EITest script.
- Step 2 – EITest script causes the host to retrieve Flash file from EITest gate.
- Step 3 – Flash file from EITest gate is used to generate HTTP GET request to Angler EK landing page.
- Step 4 – Angler EK landing page has a script that determines if the computer has any vulnerable applications.
- Step 5 – Angler EK sends an exploit tailored to the vulnerable applications (For example, out-of-date versions of IE, Flash player, Silverlight).
- Step 6 – The exploit, if successful, will cause Angler EK to send the payload and execute it as a background process
- Step 7 – The host is infected by the malware payload sent by Angler EK.

IOCs:

Regex to find embedded redirection URLs: \.php\?(s)id\[A-Z0-9]{50}

MD5 for Flash redirector: 596c8a11b03c2ef9d890f85c631ba5e8

String "creativness.exe"

Reference: <http://malwarebreakdown.blogspot.com/>

Congratulations.