Mock Malware Outbreak

A Guided Simulation to Prepare Your Organization

Written By: Chris Frenz

Prepared by the AEHIS Incident Response Committee

Thanks to the great work by our AEHIS Incident Response Committee, specifically committee chair and AEHIS member Chris Frenz, AEHIS is proud to make this guidance document available to members and the information security community alike. As our organization explores more ways of developing and promoting best practices among information security leaders, this document will help executives understand the risks they face in real time, while exposing their teams to much-needed 'live fire' training.

AEHIS
TRANSFORMING SECURITY

About AEHIS

The Association for Executives in Healthcare Information Security (AEHIS) was launched in 2014 to provide an education and networking platform to healthcare's senior IT security leaders. With over 850 members, AEHIS is advancing the role of the chief information security officer (CISO) through education, collaboration, exchange of best practices and advocacy in support of secure health information for the protection of both healthcare organizations and consumers. For more Information, please visit aehis.org.

Mock Incidents

Mock incidents are a means for organizations to walk through the incident response plans they developed and to help familiarize staff with the procedures. Moreover, they are a means for organizations to identify ways in which their incident responses could be improved upon. Such exercises are critical for any organization serious about improving their security posture to conduct on a regular basis. First off, time is critical in any incident response scenario. The faster you can identify an incident is happening and the faster you can begin to contain the incident, the greater your potential for damage mitigation. Regularly drilling staff on incident response will help to improve incident identification and containment times as staff will be much more familiar with the needed processes and procedures. Moreover, for similar reasons, the organization's ability to recover from an incident will be improved upon, as staff members will be much better versed in what needs to be done. Faster recovery means the potential for less downtime and hence the potential for less impact on the organization's bottom line. Quite simply put, you do not want to find out your systems are down for a week longer than needed because no one on your staff remembers how to use your backup system, or some other related scenario, and the best way to prevent these scenarios is to drill them regularly. It's no different than the fire drills we all used to experience in elementary school, where when the alarm went off we all practiced swiftly and efficiently exiting the building, because having a plan and being familiar with it helps prevent people from panicking and doing the wrong thing when disaster strikes. It should be the same planned and prepared response when a cyber disaster strikes.

Improved staff response times, however, are just one of the benefits of conducting a mock incident. Let's consider for a second all of the various security controls we have in our environment. Perhaps we have firewalls, AV software, an IDS system, account lockout policies, and any variety of the thousands of things that we could be doing to protect an organization. Sure, these controls have been implemented, but how many of these controls have actually ever been tested to ensure they really work? Do we really know if that IDS system will effectively detect malicious traffic on our network or how effective any of the controls we have in place really are against stopping threats we may face? Sure, a firewall may be implemented but are the in-place rules stopping everything they should? Is our antivirus solution configured optimally for our needs? Conducting a mock incident can really help you to identify deficiencies in the controls an organization has in place, as they will help you to concretely determine which controls were effective and which were not against whatever threat you are simulating. Learning what works and, even more importantly, what does not work can really help provide critical feedback as to how an organization's security posture can be improved upon.

Lastly, conducting a mock incident, can not only help improve the timeliness and efficacy of staff members and the efficacy of existing security controls, but it can also help an organization improve it's incident response plan itself. Perhaps an organization learns via their mock incidents that they do not have adequate in-house expertise to properly handle certain areas of response and require outside expertise? As a result, they improve their plan by vetting some outside experts they can call on in the event an actual incident occurs so they are not scrambling to find qualified talent in the middle of disaster. Questions like this become readily apparent as you step through the process of responding to a mock incident.

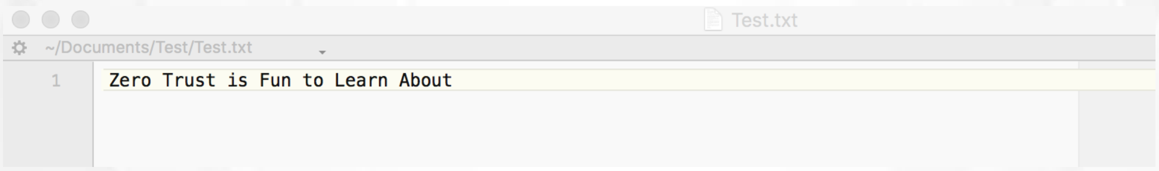
While many organizations choose to conduct tabletop mock incidents, which are not without value and a good starting point, organizations should eventually make the move towards conducting simulations of other threat vectors and other incident types that commonly target their industry vertical as well. Some organizations do this already to some extent with mock phishing exercises where they send mock phishing emails to employees to flag those who click the links as being in need of more phishing awareness training, but expanding this to other threat scenarios is highly useful as well. A mock incident scenario designed to illustrate the impact a mass malware outbreak can have on an organization will be described below.

Note: For those organizations that don't yet have an incident response plan in any form, a good reference on creating one can be found in the SANS Incident Handlers Handbook - https://www.sans.org/reading-room/whitepapers/incident/incident-handlers-handbook-33901. Don't let the lack of a plan keep you from considering the scenario below though as it may also help you to concretely demonstrate why you need one.

Antivirus Signature Basics

Before we get into the details of the mock incident itself, lets take a high level overview of how antivirus software works. Antivirus software works by using a set of signatures that are designed to uniquely identify malicious files. In their simplest forms, antivirus signatures consist of a hash of a file. A hash is a one-way function that for any given input provides a unique, but consistent output. That is, as long as the input remains unchanged, the output of the hash function will always remain the same. If the contents of file are changed at all, the output of the hash function will change. In theory no two inputs to a hash function should be able to provide the same output, so they provide a great way to create a fingerprint of any file.

To get an idea of how this works, lets do a really simple test. Open a command prompt on whatever computer is closest to you and quickly create a new folder and change directories into it, to make everything easy to visualize. Now open up a text editor and type any string of characters your heart desires, just make sure its multiple characters long. For illustration purposes, a text file containing "Zero Trust is Fun to Learn About" is going to be used (Figure 1).



```
Zero Trust is Fun to Learn About
```

Figure 1: The Test.txt text file.

Now let's save the file and compute the hash of that particular file. In the example, the file is called Test.txt. For Windows users this can be done with the command:

```
FCIV -md5 filename
```

where filename is the name of the file you just created. For Linux users this can be done with the command:

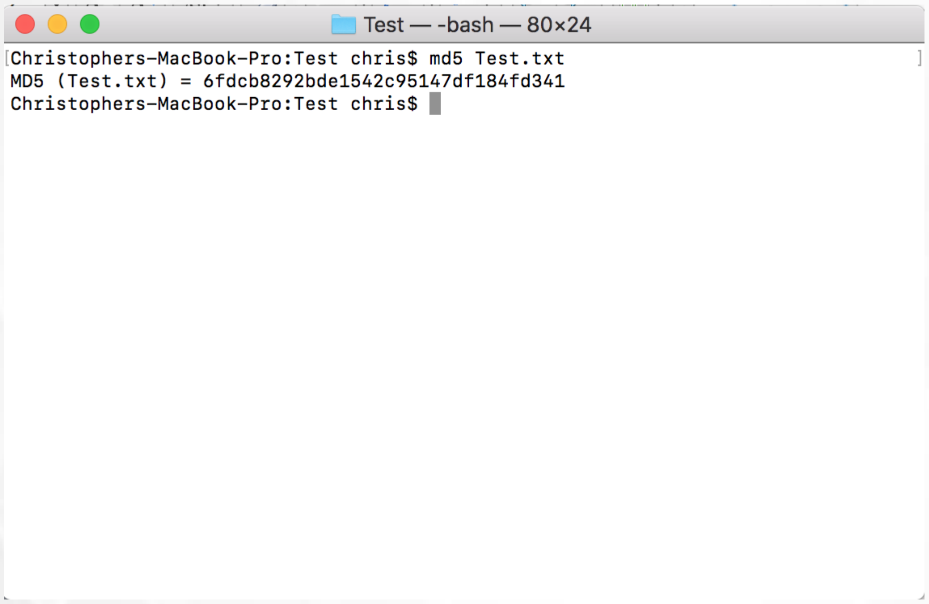
```
md5sum filename
```

and on OSX

```
md5 filename
```

can be used.

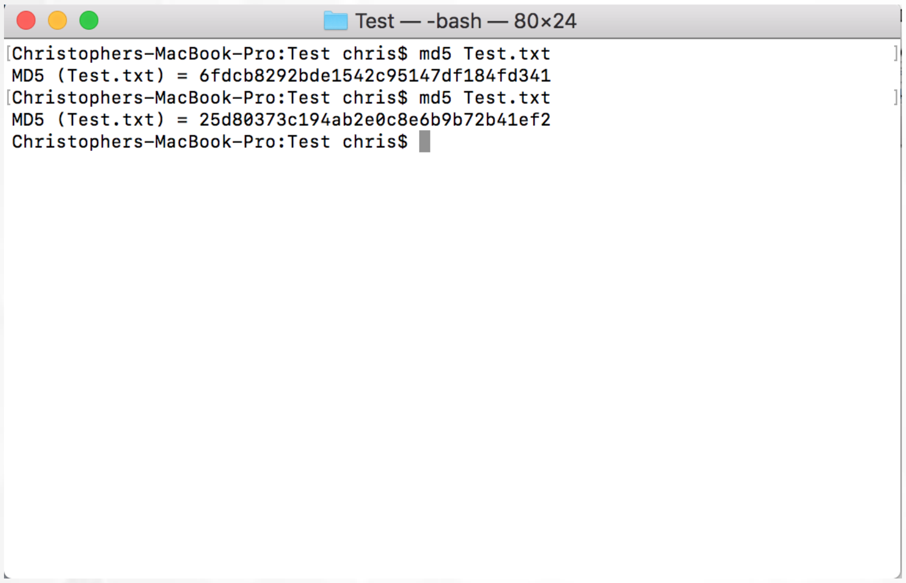
The results of computing the MD5 hash of the file are demonstrated in Figure 2. Now try renaming the file, creating a copy of it, or perhaps moving it to a new directory. You will notice, that as long as the file contents remain unchanged, the hash of the file also remains unchanged. Thus, this hash value can serve as our signature for this particular file and can be used to identify any instances of this file in any location on our computer or other computers. This is the basis behind which antivirus signatures work.



```
Christophers-MacBook-Pro:Test chris$ md5 Test.txt
MD5 (Test.txt) = 6fdcb8292bde1542c95147df184fd341
Christophers-MacBook-Pro:Test chris$
```

Figure 2: The hash of our test file.

To better show the uniqueness of hashes and their specificity to one particular file, let's conduct one final test. Let's open the Test.txt file back up and lets edit a single character in the file, thereby making a very minor change to the contents. For demo purposes the capital L in Learn is going to be replaced with a lowercase l to make learn – a seemingly minor change. Now let's recompute the hash value (Figure 3).



```
Christophers-MacBook-Pro:Test chris$ md5 Test.txt
MD5 (Test.txt) = 6fdcb8292bde1542c95147df184fd341
Christophers-MacBook-Pro:Test chris$ md5 Test.txt
MD5 (Test.txt) = 25d80373c194ab2e0c8e6b9b72b41ef2
Christophers-MacBook-Pro:Test chris$
```

Figure 3: The hash values before and after the character change.

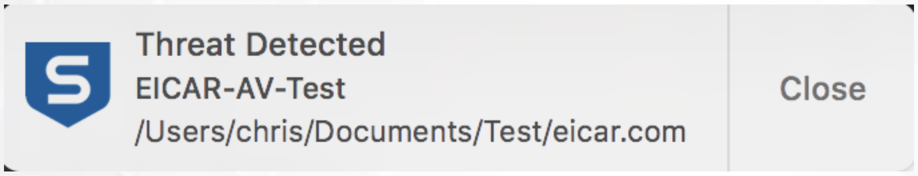
Notice how a change in a single character changed the hash value? By making the character change we effectively changed the file's signature as our modified file is no longer an exact match to the original file. This shows the level of specificity that these signatures can have.

The EICAR Test File

At this point, you are probably wondering how any of this pertains to setting up a mock incident or helps with understanding zero trust, but it just so happens there is a particular file that is completely harmless but for which the vast majority of antivirus products have just such a signature for – The EICAR Test File. The EICAR Test File is a 68 byte long ASCII file that contains the following characters:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

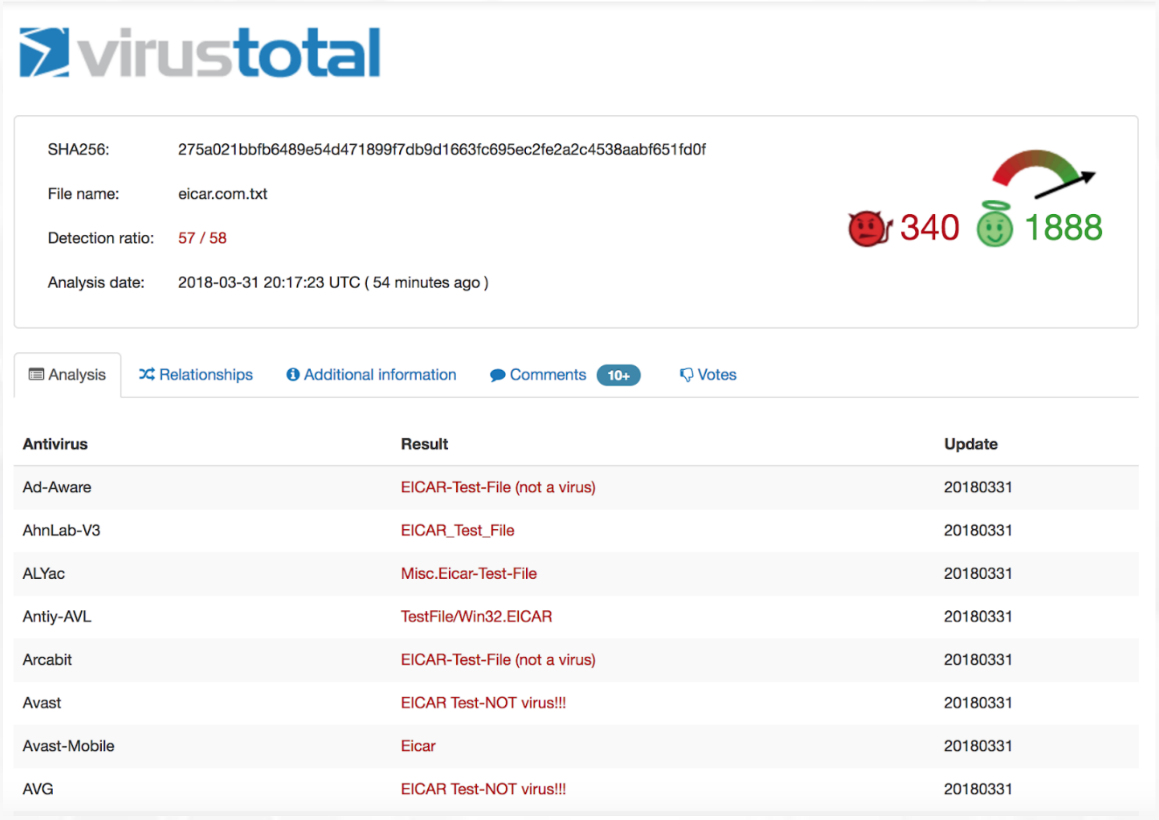
The file is a valid DOS program that if successfully executed will print EICAR-STANDARD-ANTIVIRUS-TEST-FILE to the screen. The EICAR Test File can be downloaded from http://2016.eicar.org/86-0-Intended-use.html or we can simply paste the above string of characters into a text editor and save it as eicar.com. Don't save it as a txt file for this particular test because many antivirus products will not scan content that is not considered potentially executable. Executing this file or perhaps even trying to save it will likely set off your antivirus software with an alert similar to the following (Figure 4):



Threat Detected
EICAR-AV-Test
/Users/chris/Documents/Test/eicar.com

Figure 4: An AV alert triggered in response to the EICAR test file.

An idea of the universality of the recognition of this particular file by antivirus products can be readily obtained from VirusTotal where, as of the time of this writing, 57 out of 58 products tested against it flag it as a virus (Figure 5). This makes the use of this file an ideal basis for developing a plan to simulate a malware outbreak within our organization.



SHA256:	275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
File name:	eicar.com.txt
Detection ratio:	57 / 58
Analysis date:	2018-03-31 20:17:23 UTC (54 minutes ago)

Antivirus	Result	Update
Ad-Aware	EICAR-Test-File (not a virus)	20180331
AhnLab-V3	EICAR_Test_File	20180331
ALYac	Misc.Eicar-Test-File	20180331
Antiy-AVL	TestFile/Win32.EICAR	20180331
Arcabit	EICAR-Test-File (not a virus)	20180331
Avast	EICAR Test-NOT virus!!!	20180331
Avast-Mobile	Eicar	20180331
AVG	EICAR Test-NOT virus!!!	20180331

Figure 5: VirusTotal output for the EICAR Test File

A Typical Malware Outbreak Scenario

While some malware outbreaks such as the SamSam ransomware are notable exceptions in that it gains a foothold in organizations through exploiting JBoss vulnerabilities or poorly secured RDP servers, the majority of malware attacks begin with a phishing email. In fact estimates by the Anti-Phishing Working Group estimate that 91% of cyberattacks begin with a phishing email. The 2015 Verizon Data Breach report showed that 23% of recipients will open a phishing email and that 11% of recipients will click the link in the email or open the attachment. While 11% may seem insignificant, one must consider that statistically speaking, in a phishing campaign of just 10 emails there is a greater than 90% chance of someone in the organization clicking on that malicious link. Meanwhile the 2016 Verizon Data Breach report showed that the click rate had increased to 13% and the Anti-Phishing Working Group estimated that in 2016 almost 93,000 phishing attacks occurred per month. Given these statistics, it becomes readily apparent as to just how exposed to cyber threats and malware our organization likely is and why organizations need to step up their defenses against such threats.

Note: In this booklet, we focus on mainly the testing of defenses via mock incidents, but anyone looking to get a broader feel for the types of controls that can be put in place to defend against malware, should consider checking out the OWASP Anti-Ransomware Guide - https://www.owasp.org/images/6/64/Anti-RansomwareGuidev1-7.pdf. The guide provides a defense in depth listing of controls that can be used to prevent, mitigate, respond to, and recover from a ransomware (or other malware) attack.

While, spam filters, phishing awareness training, endpoint security controls, and other defenses may be used to stop such an attack early in the kill chain, there is a very real possibility that such an attack will lead to the compromise of a system in your environment and that compromised system will be used as a staging ground to laterally move through your organization and compromise other systems. This is the type of scenario that we are trying to simulate.

A potential setup for such a simulation will be described in the proceeding paragraphs, but keep in mind that each environment is a bit different and the scripts and setups may have to be tweaked a bit to work within your particular organization. The scenario described below were the steps used to execute this test in a NYC area hospital. Before we get into the steps to run through this particular scenario, let's do some prep work first, by creating a list of all PCs that you intend to target. Active Directory or a like source is a great way to export such a list. Save this target list as computers.txt. This list of PCs will be fed into the script as targets. Next, choose a Windows PC to serve as the PC that will be simulating ground zero and serve as the initial point of compromise. This is the PC that our script will run on and will serve as a staging ground to try to "compromise" all of the other PCs that are in the target list. In addition to the script and the target list, two other components will be needed. The first is the EICAR test string mentioned above and the second is the PSEXEC utility which is downloadable from Microsoft. You may need to disable AV before downloading these files as most AV packages will prevent you from saving the EICAR test string. Some AV packages will also block the use of PSEXEC given that it played a role in facilitating the worm-like behavior observed during the WannaCry attacks. It is recommended that you save all four components into the same directory on the ground zero PC.

Disclaimer: While the EICAR test string is theoretically harmless please test prior to conducting a large-scale simulation to ensure compatibility and safety within your environment. For example, some AV packages can be set to auto-quarantine a PC when a virus is detected. You don't want all of your PCs dropping off the network due to this behavior being triggered by the EICAR test string by turning a security feature into a DoS vector.

The Script

The demo script below was written in Perl and can be executed on a Windows PC using Strawberry Perl. If Perl is not your thing, a like script can readily be authored in Python, Powershell, or whatever your scripting language de jour is.

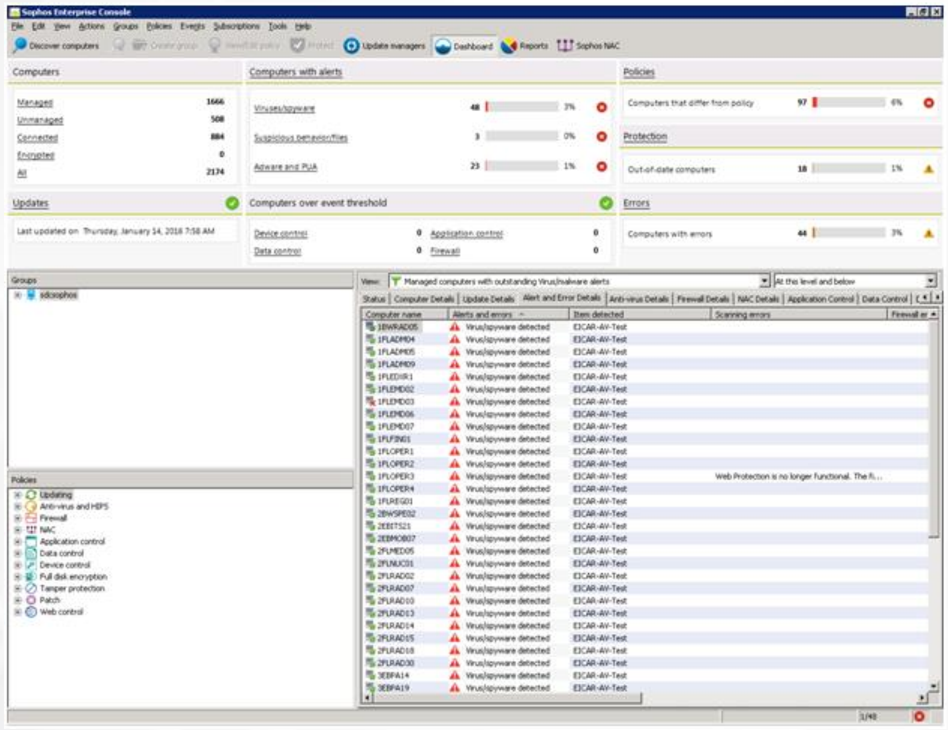
```
use File::Copy;
use Parallel::ForkManager;
use strict;
use warnings;

#allows "malware" to be spread to multiple machines simultaneously
my $MaxProcs=10;
my $pm=new Parallel::ForkManager($MaxProcs);

#opens file that stores list of PC names
open(my $computers, "<", "computer.txt")
  or die "cannot open < computer.txt: $!";

while(<$computers>){
  $pm->start and next;
  my $computer=$_;
  $computer =~ s/\r?\n$//;
  my $path1='eicar.com'; #eicar test file
  my $path2="\\\\$computer\\C\$\\Users\\Public\\Desktop";
  my $path3="\\\\$computer\\C\$\\Documents and Settings\\All Users\\Desktop";
  if(-e $path2){
   #Win7
             #copies file eicar test file to PC
   copy("$path1","$path2") or print "$computer - Copy failed: $!\n";
             #psexec commands with bad passwords to generate suspicious audit entries
             system ("PsExec.exe \\\\$computer -u administrator -p abc C:\\Users\\Public\\Desktop\\eicar.com");
             system ("PsExec.exe \\\\$computer -u administrator -p abc C:\\Users\\Public\\Desktop\\eicar.com");
             system ("PsExec.exe \\\\$computer -u administrator -p abc C:\\Users\\Public\\Desktop\\eicar.com");
             #executes eicar file to trigger on access scanning
             #must be run from an account with appropriate admin privileges on target system
             system ("PsExec.exe \\\\$computer C:\\Users\\Public\\Desktop\\eicar.com");
  }
  elsif(-e $path3){
    #WinXP
    copy("$path1","$path3") or print "$computer - Copy failed: $!\n";
             system ("PsExec.exe \\\\$computer -u administrator -p abc \"C:\\Documents and Settings\\All Users\\Desktop\\eicar.com\"");
             system ("PsExec.exe \\\\$computer -u administrator -p abc \"C:\\Documents and Settings\\All Users\\Desktop\\eicar.com\"");
             system ("PsExec.exe \\\\$computer -u administrator -p abc \"C:\\Documents and Settings\\All Users\\Desktop\\eicar.com\"");
             system ("PsExec.exe \\\\$computer \"C:\\Documents and Settings\\All Users\\Desktop\\eicar.com\"");
            }
            else{print "$computer - Not Windows";
 }}


close $computers;
```

The script starts off with some use statements which are used to import some Perl libraries called within the script and then using the Parallel:ForkMangager library to enable to the script to simultaneously target 10 PCs at once. Next, the file computers.txt is read in which loads the list of target PCs into the script. For each computer in the target list, the script then begins to copy the EICAR test string, found in the eicar.com file, to the desktop of each PC. While, hopefully, Windows XP is no longer being used as a desktop OS in your organization, the script does account for the differences in paths found between Windows XP and newer versions of Windows, such as 7 and 10, and can be used to target XP and newer desktops. A system call is then made to use PSEXEC to remotely launch the execution of the EICAR test string. This is due to the fact that the network copy alone will not be enough to trigger some AV software or to trigger certain configurations of some AV software. Execution of the EICAR test file ensures that almost all AV software will trigger an alert. You may notice that the script makes multiple PSEXEC calls and that the first three attempt to use a local administrator account along with the password abc to copy the EICAR file. This was done intentionally to generate events in the Windows security log to help track down the ground zero PC. While such password guesses may or may not happen in a real ransomware attack, the script was designed to intentionally leave a trail so that even smaller hospitals with more limited security staffing and limited security tools could perform the exercise to completion. The last PSEXEC command will attempt to copy the string with the access level of the account used to run the script. When running the script, be sure to verify that you are using account with sufficient privileges to perform the copy.

Running the Simulation

Before executing the script, it is important to get the appropriate buy-in among senior leadership within the organization, but it is also recommended to keep to a minimum the number of people who are aware of the simulation to ensure that the results are realistic. If people know a test is going on, it's going to impact their behavior and responses. It's also recommended that all but the most senior leadership in IT and IT security are unaware of the fact that a simulation will be occurring. It is really important to get an accurate assessment of how quickly such a threat could spread through your organization, how long the threat would take to be discovered, and how long it would take to isolate the threat to ensure that further damage potential is mitigated.

Once approvals are granted and the appointed date and time have arrived, you can kick off the script and within a few minutes you will begin to see PCs on the target list begin to fill up your AV console with infection alerts as shown in Figure 6.



Computers	
Managed	1666
Unmanaged	508
Connected	884
Encrypted	0
All	2174

Computers with alerts		
Viruses/spyware	48	3%
Suspicious behavior/files	3	0%
Adware and PUA	23	1%

Policies		
Computers that differ from policy	97	6%

Protection		
Out-of-date computers	18	1%

Updates

Last updated on Thursday, January 14, 2016 7:58 AM

Computers over event threshold			
Device control	0	Application control	0
Data control	0	Firewall	0

Errors		
Computers with errors	44	3%

View: Managed computers with outstanding Virus/malware alerts | At this level and below

Status | Computer Details | Update Details | Alert and Error Details | Anti-virus Details | Firewall Details | NAC Details | Application Control | Data Control

Computer name	Alerts and errors	Item detected	Scanning errors	Firewall er
1BWRAD05	Virus/spyware detected	EICAR-AV-Test		
1FLADM04	Virus/spyware detected	EICAR-AV-Test		
1FLADM05	Virus/spyware detected	EICAR-AV-Test		
1FLADM09	Virus/spyware detected	EICAR-AV-Test		
1FLEDIR1	Virus/spyware detected	EICAR-AV-Test		
1FLEMD02	Virus/spyware detected	EICAR-AV-Test		
1FLEMD03	Virus/spyware detected	EICAR-AV-Test		
1FLEMD06	Virus/spyware detected	EICAR-AV-Test		
1FLEMD07	Virus/spyware detected	EICAR-AV-Test		
1FLFIN01	Virus/spyware detected	EICAR-AV-Test		
1FLOPER1	Virus/spyware detected	EICAR-AV-Test		
1FLOPER2	Virus/spyware detected	EICAR-AV-Test		
1FLOPER3	Virus/spyware detected	EICAR-AV-Test	Web Protection is no longer functional. The fi...	
1FLOPER4	Virus/spyware detected	EICAR-AV-Test		
1FLREG01	Virus/spyware detected	EICAR-AV-Test		
2BWSPE02	Virus/spyware detected	EICAR-AV-Test		
2EBITS21	Virus/spyware detected	EICAR-AV-Test		
2EBMOB07	Virus/spyware detected	EICAR-AV-Test		
2FLMED05	Virus/spyware detected	EICAR-AV-Test		
2FLNUC01	Virus/spyware detected	EICAR-AV-Test		
2FLRAD02	Virus/spyware detected	EICAR-AV-Test		
2FLRAD07	Virus/spyware detected	EICAR-AV-Test		
2FLRAD10	Virus/spyware detected	EICAR-AV-Test		
2FLRAD13	Virus/spyware detected	EICAR-AV-Test		
2FLRAD14	Virus/spyware detected	EICAR-AV-Test		
2FLRAD15	Virus/spyware detected	EICAR-AV-Test		
2FLRAD18	Virus/spyware detected	EICAR-AV-Test		
2FLRAD30	Virus/spyware detected	EICAR-AV-Test		
3EBFA14	Virus/spyware detected	EICAR-AV-Test		
3EBFA19	Virus/spyware detected	EICAR-AV-Test		

1/48

Figure 6: AV Console beginning to display machines infected with the EICAR Test String.

As this incident is running keep the following considerations in mind as they will help you to evaluate things that can be improved upon:

1. How long until the issue is noticed?
2. Did any users within the organization report the issue?
3. How long was the time between the issue being noticed/reported and IT/IS beginning to investigate?
4. Are there any parts of the network or groups of machines this is not successfully spreading to?
5. Related to the above, are any existing controls working to mitigate the spread of this infection?
6. How is IT/IS going about investigating this issue?
7. How long between when the investigation begins until the ground zero PC is found and the source of the infections isolated from the network?
8. Was the incident response plan followed?
9. Was the incident response plan adequate or did it fall short in certain regards?
10. How was communication between departments?
11. Were any potential clinical issues raised and discussed?
12. Were any potential impacts on patient safety considered?

Post Simulation

The post simulation period is actually one of the most critical aspects in conducting a simulated mass malware outbreak in that it allows you to identify what controls worked in your environment to contain the attack and what controls did not work. It also allows you to determine areas where additional controls may be warranted. For example, in a NYC area hospital where this test was run, network segmentation was shown to be an effective way to keep the threat isolated to just a particular part of the network. At that time the hospital had ACLs that restricted the flow of traffic between VLANs, but the efficacy of network segmentation as a control encouraged the hospital to pursue an even more segmented zero trust architecture in which two PCs on the same VLAN and plugged into the same switch could no longer communicate with each other unless there was an explicit need to. While analyzing the post simulation data it is an ideal time to ask the question of what additional mitigations and detective controls do we require?

Moreover, the post simulation period is also a great time to assess the preparedness of the organization's employees. Having an incident response plan is great, but it's only useful if people know what it is and what it requires them to do. Saying that the systems engineer is responsible to recover xyz system from backup is great, but you don't want the first time you test your restore procedures to be when a real incident hits and find there are hours lost because your engineer doesn't remember how to properly perform a restore (or worse yet find you can't restore because you find your backups were not done properly). Now is the time to identify issues like these so when an actual disaster hits you are well practiced and prepared.

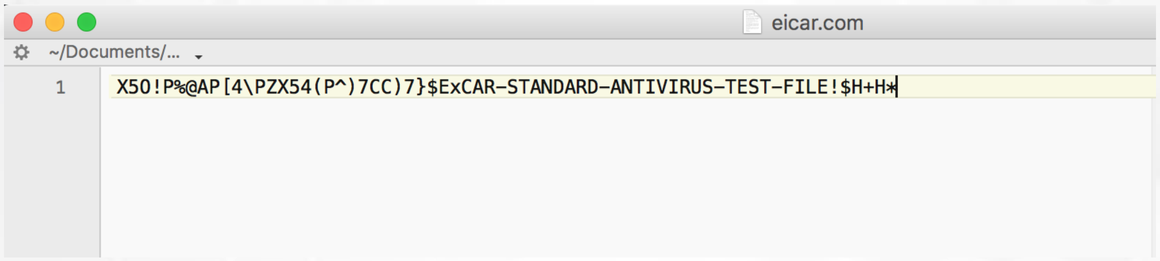
Even in cases where people and controls performed as expected, it is also important to give consideration to ways in which the response process could be optimized. Perhaps IT noticed some of the alerts, but perhaps the issues would have been noticed faster if end users were more aware of the importance of calling IT immediately when an AV prompt went off. Perhaps there are ways in which workflows could be optimized. For example, in the NYC area hospital IT staff were receiving AV alerts in their email, but they found that changing this workflow to instead link these alerts to their ticketing system improved how quickly alerts were detected and acted upon and improved future response times.

Moreover, it is important to see how involved non-IT departments became in dealing with this issue. While the EICAR string is harmless, did anyone ask the important questions of could access to our EHR, PACS, and other systems be impacted and how can we prevent patient care issues from arising from such a loss of access? One of the big failings that are frequently observed in running and participating in both tabletop and simulated incidents is a failure of clinical leadership to want to become involved in cyber issues. Have your team consider how bad things would have been if each one of those systems that became compromised by the EICAR string had become unavailable. What would the impact have been on hospital operations? What ways could you address those impacts? Should any additional controls, workflows, or incident response plan changes be introduced to minimize any issues such a malware attack would cause?

Take your answers to all of these questions and use it to further harden your environment and improve your response plan. Try the incident again perhaps at a later point in time to see if the improvements can be quantified and if any room for further improvement can be identified. Once you are confident of having a reasonable ability to deal with a threat such as this, perhaps its time to begin to step things up a notch and consider that there will not always be a signature for every threat that your environment will face. In fact, in many cases, it can be surprisingly easy to get a signature-based defense to fail.

Shortcomings of Signature Based Defenses

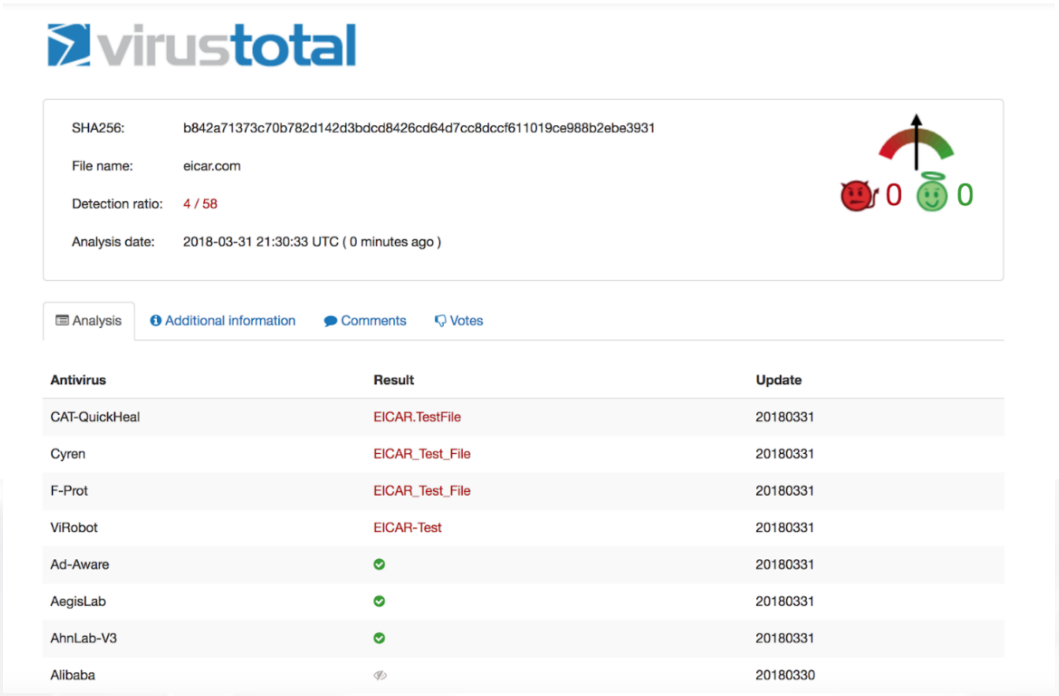
Traditional approaches to security have largely centered on the idea of blocking things that are known to be bad. The problem is "bad" changes all of time and many threats your organization will face don't fall into this known category. Let's set up a real simple test to see why security professionals need this change of mindset. Remember what happens to the hash when even a single character is changed in a file? The hash of that file will also change meaning the signature of the file is effectively changed. Let's change a non-executable part of the file, by changing the I in EICAR to the letter x as shown in Figure 7.



```
X5O!P%@AP[4\PZX54(P^)7CC)7}$ExCAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Figure 7: A slightly modified EICAR File.

Now, let's run that very slightly modified file through VirusTotal again and check the results (Figure 8).



SHA256: b842a71373c70b782d142d3bdcd8426cd64d7cc8dccf611019ce988b2ebe3931
File name: eicar.com
Detection ratio: 4 / 58
Analysis date: 2018-03-31 21:30:33 UTC (0 minutes ago)

Antivirus	Result	Update
CAT-QuickHeal	EICAR.TestFile	20180331
Cyren	EICAR_Test_File	20180331
F-Prot	EICAR_Test_File	20180331
ViRobot	EICAR-Test	20180331
Ad-Aware	✓	20180331
AegisLab	✓	20180331
AhnLab-V3	✓	20180331
Alibaba		20180330

Figure 8: The VirusTotal results for our modified EICAR Test String.

The implications of what we see are indeed quite scary. Almost none of the antivirus engines recognized the modified version of the file at all. While this may be a somewhat over simplistic view of how antivirus software uses signatures it does illustrate the limitations of using signature based defenses, in that modifications can be made to segments of malicious code to change their signatures and help them to evade detection by AV. It is not uncommon for malware authors to conduct their own such testing to ensure their variants will evade detection and it is not uncommon for numerous variants of the same malware to be released. In fact, a recent Verizon Data Breach Investigations Report estimated that between 70% and 90% of the malware an organization sees is unique to that particular organization. Thus while traditional antivirus and their signature based approach may be effective at detecting known threats, they are far less likely to detect an unknown threat and turning a piece of malware into a non-detectable variant and effectively making an "unknown" threat is far easier than many would imagine. This is one of the primary reasons organizations need to begin to deploy security controls that do not rely solely on signatures to stop threats.

If you were to repeat the test above with a modified EICAR test string, would the threat be noticed at all spreading through your network? The EICAR test string itself is easily noticeable because it triggers your AV, the modified EICAR will likely not. Do adequate controls exist to identify this more silent threat? Would the atypical network traffic be picked up because baseline traffic is well established and monitored? Would the security event in the event log be detected and correlated to reveal the ground zero PC? How long would the threat persist if a signature-based defense were to fail? What defenses can we put in place to prevent an outbreak of malware for which no signatures exist?

Conclusions

Now that you have worked your way through these initial exercises, hopefully you have a much better feel for why you should make simulated attacks and mock incidents a routine part of your security preparedness strategy. It is important to keep in mind however, that the exercises presented here just scratch the surface of what can and should be done. Stay tuned for more releases in this series that will walk readers through how to conduct even more simulated incidents.