
Endpoint Security Testing

Preparation Checklist

By Lidia Giuliano  [@pink_tangent](https://twitter.com/pink_tangent)  tangentmelb@gmail.com

Last Updated: 17th August, 2017

1. Overview

Testing endpoint security (EPP) solutions can feel like an overwhelming task when you're starting from scratch. Knowing where to source the samples or even how to mutate, I mean, "lions, tigers and bears" right! That's where my research over the past 9 months is here to help you focus on your business goals and objectives, and hopefully set you up for success.

2. Purpose

The purpose of this document aims to help you in your preparation and planning for your EPP testing and enable you to determine what is the best solution and outcome for your business needs and objectives. Knowing what *"you need"* vs what a vendor's sales and marketing teams tells *"you what you want or what is the new thing you must protect against"* are two very distinct differences. My success from BlackHat 2017 is that you walk away knowing you solved a critical business problem and can talk to that with confidence.

3. Scope

It's important to note that the scope of this testing was business related and not a penetration test. The goal was not to find zero days in the software as there are plenty of companies that do that successfully today. Don't let anyone convince you that these EPP solutions have no zero days reported against their software. It's software, written by people, there will always be vulnerabilities and it is the nature of the business we are in. We all love to build, we all like to test and break.

The purpose of these scenarios were to test the EPP capabilities and layers to see where their limits were. This was accomplished using different sets of malware, old, new, static, dynamic, memory resident, hidden in powershell, using encoding mechanisms, mutating malware, turning cloud services off, etc for the sole purpose to see which solution could detect and prevent malware.

4. Malware Sources

One of the many questions I am always asked is *“Where do I get my malware? Where does one possibility even source 20,000-30,000 pieces?”*. There are a lot of online repositories available with lots of samples. Sourcing the malware is actually easy, just make sure you aim for diversity of malware which is discussed in the Section 5. Additionally, refer to my [malware-source-list](#) on GitHub if you are still stuck.

To date I have 7 different malware sources, including personal sources we used collected from honeypots. I encourage you to also ask your vendor where they source their malware from, but add it to your testing collection. If you choose to only use their sample sets or suggestions, logically the detection and prevention rates will more than likely be in their favour. Variety, variety and more variety!

Use new malware samples as many repos have daily adds, but also go the old route and get the Zeus and Poison Ivy type samples. Many will require you to subscribe (free), others paid (like VirusTotal) and others will ask you to explain your access reason. Be honest! You are doing a POC and the malware provided by those sites owner will help you test the EPP capabilities. Don't be shy and build a solid collection! Lastly, make sure you have a variety of malware families, ransomware, exploit kits, worms, viruses, backdoors, RATs, adware, spyware and lots of annoying PUPs.

As a side note, the greyware and PUP-types files may not be a big deal for your use case, but if it is, some solutions do not have much in the way of mutated PUP detection. I found this an easy attack vector in over half of the EPP agents I tested, whereas on the flip side others generate a high false-positive rate as they are over cautious. It's a balance, so have a good think about what is important to your business goals and objectives.

5. Malware Types

You will find that the majority of the malware types that are available to download will be PE files (portable executable files). Given this, try to find a variety of file types. In some cases I built my own, malicious zip files, jar files, false-positive directories, I renamed file extensions, I wrote weaponised scripts which called or sometimes downloaded the malware I wanted. I tried all possible ways to make the malware attack diverse and execute in a variety of ways. It will increase your testing time, sure, but the outcomes are well worth it.

As per slide 15 of the BHUSA presentation, these are file types I tested. If you have more ideas or had success with other files types or execution methods that I did not list, please email me.

Now that you have your files, don't forget to mutate them, and then double mutate and triple etc. Mutation is important as it helps you create a never seen before sample. Many in the security industry refers to these as zero days. Regardless of the name you call it, the purpose of mutation is to be able to see which products still rely heavily on signatures and/or sandboxing to do their detections and detonation.

File Types:

- Portable Executables (PEs)
- Other compiled code, vbs, .bin, .com,
- Compressed files, .zip, .jar, 7-zip, etc
- Native windows scripts, batch and ps
- Obfuscate the content in the scripts
- Rename extensions to other file types
- Known Good Files
- Create a FP directory

**** Important NOT just binaries ****

With marketing hype calling out machine learning every which way, the mutation testing really brought it home for me especially with the more traditional AV solutions. I don't want ML to get a bad name as its gains are beneficial for the InfoSec community.

The hype will be called out in my white paper

In these next two sections, we address file mutation. Definitely a big area of interest and many of the questions I was asked was about how to do it and what to use.

5.1 Malware Mutation using a PE Packer

One of the simplest and easiest way to mutate a file is using a PE packer. See figure 1. You can use something like mpress or pklite. There are plenty available, try a bunch, some are easier than others.

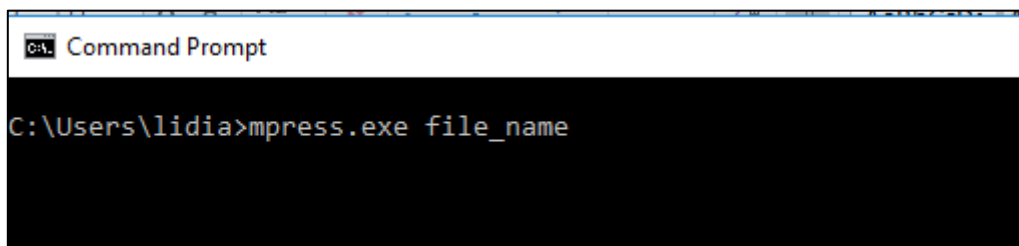


Figure 1: Mutate Files using a PE Packer

The above is useful if you have one file to mutate, but what if you have 100's or 1000's of samples. Execute a for loop on the command line as seen in figure 2 or create a batch. This essentially looks at all the files and for each file in the "malware-1" directory create a new mutated file using mpress tool. Easy! Get the before and after hash value if you want to be 100% sure..

Extra Notes:

- In the case of the "malware-1" directory which was created for your mutated malware, make sure you place a **copy** of the malware here otherwise you will clobber your original malware set
- **Beware** I found using mpress caused a lot of file corruption, probably 15-20% of my original data, so test out a few different PE packer software, mpress isn't bad, but it is important you are informed.

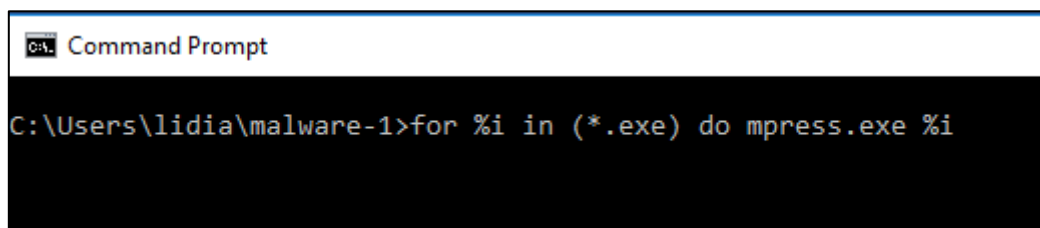


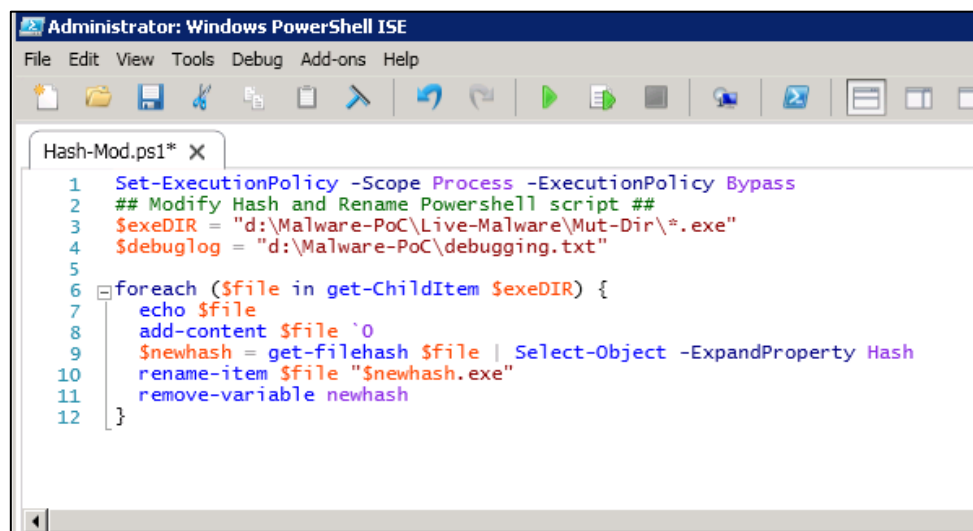
Figure 2: Bulk Mutation using a PE Packer

Once completed, archive, password protect and zip up the malware. On windows I used 7zip as it enabled me to set a password. Remember it's live malware, so password protect it and keep it safe.

5.2 Malware Mutation using a Hash Modifier

Another way to change the hash value of your file is using a hash modifier which adds a null byte at the file. I cheated and used the example from the TestMyAV website. I did need to make some modifications to it as it *DOES NOT* work on older versions of powershell. So just keep that in mind if you want to use what's on their website. Or make sure you upgrade to a newer version of powershell on your machine where you intention to store and mutate your malware samples.

The following image, figure 3, illustrates the script used for hash modification.

A screenshot of the Windows PowerShell ISE (Integrated Scripting Environment) window. The title bar reads "Administrator: Windows PowerShell ISE". The menu bar includes "File", "Edit", "View", "Tools", "Debug", "Add-ons", and "Help". The toolbar contains various icons for file operations and execution. The script editor shows a file named "Hash-Mod.ps1" with the following PowerShell code:

```
1 Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
2 ## Modify Hash and Rename Powershell script ##
3 $exeDIR = "d:\Malware-PoC\Live-Malware\Mut-Dir\*.exe"
4 $debuglog = "d:\Malware-PoC\debugging.txt"
5
6 foreach ($file in get-ChildItem $exeDIR) {
7     echo $file
8     add-content $file `0
9     $newhash = get-filehash $file | Select-Object -ExpandProperty Hash
10    rename-item $file "$newhash.exe"
11    remove-variable newhash
12 }
```

Figure 3: Powershell Script for Mutating Files

Mutating malware using this powershell script is no different than using a PE packer. Compared to mpress I got no errors using this method, but note the techniques are different. Here are the steps:

- Create your directory where your malware will be mutated
- Place a *copy* of your malware into that directory
- Change the directory path in your powershell code to point to your new directory, in this example, Mutate-Test.
- Execute the script using the ISE editor or via the command line

```

1 Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
2 ## Modify Hash and Rename Powershell script ##
3 $exeDIR = "d:\Malware-PoC\Live-Malware\Mutate-Test\"
4 $debuglog = "d:\Malware-PoC\debugging.txt"
5
6 foreach ($file in get-ChildItem $exeDIR) {
7     echo $file
8     add-content $file `0
9     $newhash = get-filehash $file | Select-Object -ExpandProperty Hash
10    rename-item $file "$newhash.exe"
11    remove-variable newhash
12 }

```

```

PS D:\Malware-PoC\Live-Malware\Mutate-Test> D:\Malware-PoC\Live-Malware\Scripts\Hash-Mod.ps1

Directory: D:\Malware-PoC\Live-Malware\Mutate-Test

Mode                LastWriteTime         Length Name
----                -
-a---             1/3/2017  4:07 PM      2097158 0004EF4AB7BAD11B746692FF8BF08FC7FFF3B8C161CBCBCEB9
                        37BB49D4201D17.exe
-a---             1/3/2017  4:07 PM      2125827 018C40C826ED2380BD5EED0E5648AD5534C7C841BBAFA9C659

```

Figure 4: Mutating Malware using Powershell

Figure 4 is a screen snapshot the script running successfully with 100 files mutated in under 10 seconds. Results are the illustrated in figure 5. In less than one minute files have a new hash value and are ready to test. If you want to test that the hash values did changed, there are a few ways.

```

C:\Users\lidia\malware-1> powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\lidia\malware-1> get-filehash .\test-file.txt

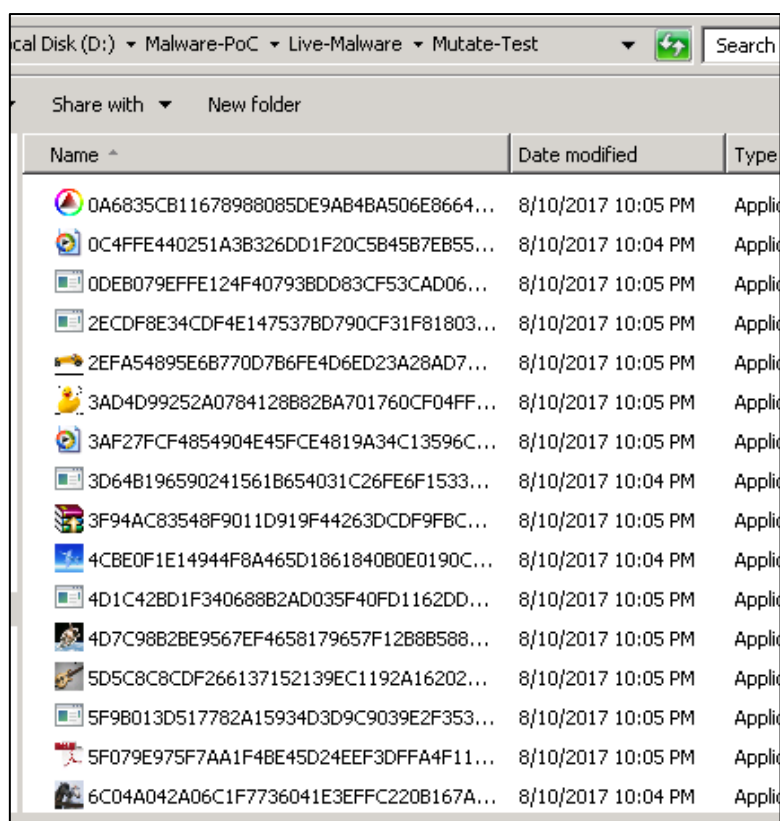
Algorithm      Hash
-----
SHA256         E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855

```

Figure 5: Check the hash value

Figure 5 shows one method you can use to check the hash value of a file. In the above example, I have executed powershell and I am running the get-FileHash function. Important to note here that I have not specified the hashing algorithm in the above example, so you can amend a “-Algorithm MD5”, “-Algorithm SHA1”, “-Algorithm SHA256”, etc.

If you are testing on Linux, you can check the files hash value using “md5sum”, “sha1sum”, “sha256sum” etc again depending on which hash algorithm you want to test against.



Name	Date modified	Type
0A6835CB11678988085DE9AB4BA506E8664...	8/10/2017 10:05 PM	Applic...
0C4FFE440251A3B326DD1F20C5B45B7EB55...	8/10/2017 10:04 PM	Applic...
0DEB079EFFE124F40793BDD83CF53CAD06...	8/10/2017 10:05 PM	Applic...
2ECDF8E34CDF4E147537BD790CF31F81803...	8/10/2017 10:05 PM	Applic...
2EFA54895E6B770D7B6FE4D6ED23A28AD7...	8/10/2017 10:05 PM	Applic...
3AD4D99252A0784128B82BA701760CF04FF...	8/10/2017 10:05 PM	Applic...
3AF27FCF4854904E45FCE4819A34C13596C...	8/10/2017 10:05 PM	Applic...
3D64B196590241561B654031C26FE6F1533...	8/10/2017 10:04 PM	Applic...
3F94AC83548F9011D919F44263DCF9FBC...	8/10/2017 10:05 PM	Applic...
4CBE0F1E14944F8A465D1861840B0E0190C...	8/10/2017 10:04 PM	Applic...
4D1C42BD1F340688B2AD035F40FD1162DD...	8/10/2017 10:05 PM	Applic...
4D7C98B2BE9567EF4658179657F12B8B588...	8/10/2017 10:05 PM	Applic...
5D5C8C8CDF266137152139EC1192A16202...	8/10/2017 10:05 PM	Applic...
5F9B013D517782A15934D3D9C9039E2F353...	8/10/2017 10:05 PM	Applic...
5F079E975F7AA1F4BE45D24EEF3DFFA4F11...	8/10/2017 10:05 PM	Applic...
6C04A042A06C1F7736041E3EFFC220B167A...	8/10/2017 10:04 PM	Applic...

Figure 6: Result of powershell execution

As seen in Figure 6, after using the powershell script to modify the files, the timestamp has changed as well as the files hash value. Note that time presented above is in US-format and not International otherwise it would appear I have a time machine.

In general my advice is not to just mutate using one of the above methods discussed above, use both or a third, but be creative. The goal is to make the files as unseen as possible.

EPP solutions say they are all about behaviour detection, and using math models to detect these patterns or known attack types. In these scenarios, mutation does not change behaviour, the chocolate is still inside, just the hash (wrapper) value has changed. Therefore, if your EPP vendor is telling you they are no longer using signatures, then double or triple mutate your files, I guaranteed you will see some unexpected results.

Finally, perhaps your selected EPP solution you have decided to POC may have sandboxing technology. This is not machine learning, this is the robotic bomb diffuser detonating your malware in a safe environment. That takes time, system resources and cloud technology if the result is inconclusive. In short, no machine learning lives here. Be wary of these claims.

6. Identify Your User Profiles

Before you even begin executing malware, think about how many different users / profiles operate in your environment. Depending on the different level of access, it may be useful to execute the same scenarios across different users as the outcome will be different.

For example, do you have only one standard profile for every user who logs on, or do you have multiple profiles, a default non-privileged user, developer, local admin? This could drive your decision on the products you test. Your low-privileged user may not have access to their local drive and/or command line access. GPO rules may prevent them from even running powershell. Therefore, when you test your weaponised powershell scripts or try to run a scripting exploit, these may not execute as you already have a preventive control in place that could stop the malware from executing in the first place.

Or perhaps you have the complete opposite setup where you have one profile for all users, everyone can access their local drive, everyone has access to the command line and all the native windows functionality.

Identification of your different users in your organisation should be a consideration in your planning. Yes it means you have to test the one scenario 2-3 times depending on your amount of profiles, but you want to try and simulate your environment as close as possible. Additionally, it's important to identify high-risk profiles during this process.

7. Consider Your Testing Environment

Throughout this entire process, this was an extremely interesting topic and I could probably talk for hours on this. Firstly, **DO NOT test LIVE MALWARE in your production environment**. You will get escorted out of the building faster than your actual detonation.

7.1 What should you test in production

Some vendors will tell you to install the agent directly into your production environment. We all know this is not possible and throwing live malware round just to see what gets captured vs what is undetected will probably not sit well with management.

The point is, vendors understand that testing malware in a virtual environment has its challenges. Therefore, unless your organisation has a completely air gapped network or isolated segment to offer you and you are ok for the rebuild times which you are about to hit with, then you are good. Sadly, this is not the reality. Mimicking a production like environment can be difficult and time consuming, but not impossible. I was able to test against a company image which was prepared for me.

When the vendors says test in production, what they really mean to say is install their agent on your organisational endpoint and leave it in monitor-type mode. I would agree with this approach as it allows you to see what the agent thinks are malicious files when in reality it may be a company generated file or script. This provides a lot of insight and good learnings about your system. This is an observational and learning activity *only* and the objective is to collect information about your systems and how compatible the agent may or may not be with other processes and software on your network. Definitely no malware testing if you decide to setup and install the agent on a few production systems!

7.2 My vendor says they have their own cloud for testing

Great idea right! Low cost, setup time is zero, boss will be happy so all I need to do is find my malware.

Initially I too thought this was a great idea, especially from a convenience perspective. I also made the assumption I would infect their systems and asked how I could rebuild from a snapshot. There were no snapshots provided as part of their setup and I didn't want to have to deal with cross contamination. Additionally, the vendor was controlling the environment. I didn't want this and nor I would recommend this approach to carry out your POC. It also makes testing apples for apples impossible as your endpoint environment is not standardised. You are not able to state hand on heart you tested equally for all agents.

DO work together with your vendors and setup the malware policies in their SaaS or appliance, but you must control the endpoint environment. Some vendors are nervous using this approach as they lose visibility of what you are doing hence control but nobody knows your environment better than you. In my case, I had a copy of our desktop image and importing that into their cloud was never going to work anyway.

During my testing, I worked together with my vendors setting up policies and then we did one or two installations on my test environment so they had a level of confidence that I was not hiding anything. This was an important relationship to establish. Be transparent and open with your vendors, tell them what you are testing and how you are doing it as many really do want to help you be successful and there is no sarcasm there.

7.3 So Where Do I Test Then?

Most malware upon execution will always try and detect if it is running in a virtual environment and this is the most challenging part regarding EPP testing. I tried to use less common virtualisation platforms or those that are used at an enterprise-level. My assumption being that either those systems would do a better job at obfuscating the fact that they were a virtualisation platform, or malware writers would not check cloud-based platforms as companies are using these for their production servers.

So let's talk virtualisation platforms:

- VirtualBox – I used this for my targeted attack scenarios and not for bulk malware testing as I don't feel the tests were as effective. Just for fun, comparison, research and my own learnings, I tested the same malware scenarios on virtual box vs KVM vs AWS just to see if there were differences in execution, which notably there were. If VB is all you have at your disposal, then just make sure your endpoint systems are configured and the snapshot the same so agent testing is consistent. The benefit of VirtualBox is it runs on many platforms and its super easy to install and setup.
- KVM – This is a Linux based virtualisation platform, very easy to use and setup. I ran this on CentOS 7. I observed the execution of more malware in KVM compared to VB. Perhaps one reason for this is because malware is rarely re-written from scratch, it's changed slightly with writers just wanting quick success, so why change the formula if it already works.

Perhaps another reason I saw higher success rates with KVM is that many large organisations are moving towards open source platforms such as KVM and OpenStack for their cloud computing needs. Therefore, malware writers can't just assume KVM is for home users anymore and are aware that organisations are using it for their enterprises solutions.

- Amazon AWS or Azure – The rest of my testing was done in AWS and I had a lot of success. Malware writers can't assume AWS is simply a test network. Many of the endpoints I spun up were the free micro builds and very cost effective. AWS also gave me the power to test against vendors that claimed their agent could "run on anything"! AWS allowed me to identify the minimum system resources required to run some of the agents which I never factored into my initial non-functional requirements. I just assumed backward capability for all.

Finally, if you do decide to make AWS one of your testing platforms, they do have a strong malware policy. It's important you let them know what you are planning on doing. Their tenant setup is really good, so the likelihood of you infecting other machines on their shared infrastructure is extremely low. However, do the right thing create a jump host and a separate isolated network for your malware testing.

If you want to see how I set it up, drop me an email, it's still active today and I am happy to share my architecture and design with you.

- Other platforms – There are lots of platforms you can chose to test with, Hype-V, vSphere, Parallels on the Mac (I have never personally used this), VMware Player and so forth and so on.

Whatever you decide upon make sure your endpoint machine images are the same for all solutions and try to use a virtualisation platform that could be used at an enterprise-level. KVM is free and AWS I think I paid \$50 a month for it, so there are low cost options which will allow you to accomplish the testing you need. Lastly make sure rebuilding your image is a fairly easy process. At a best guess, I would have rebuilt my endpoint anywhere from 150-200 times.

8. Performance Testing

One of the challenging things about POCs is testing the deployment on a large scale, this is near impossible unless you can spin up 10,000 machines, auto deploy and see how software and reporting dashboard handles the load. My advice regarding this to ask your vendor to put you in touch with their customers who have deployed on a large scale and have an honest conversation with them. That will be best effort and you will get some unbiased information.

One of the comments I hear a lot of organisations say is that want to move away from the heavier agents to these newer "lightweight" agents. In fact, when one vendor said to me that their agent would work on anything, I replied with, "Anything?". They said "Yer, try it!". My response, I spun up the most basic AWS micro build I could find, and I wanted to put a pen through my eye at how slow the system ran once the agent was installed! You make a claim, I'm going to test it!

During your testing, I would encourage you to gather statistics about your system resources of before, during and after. I was really surprised by the results and it did make a difference in my final scoring process.

Some things you might want to consider testing:

- CPU utilisation (before, during and after)
- Storage (before and after) this will demonstrate growth of logs files and quarantined data
- Memory (before, during and after)

9. Other Testing Considerations

There were some additional things that I didn't consider in my initial requirements, so hopefully you can benefit from these learnings. Before you start testing, think about how important the following requirements are:

- MFA auth into their SaaS solution
- How important is RBAC to you and does it need to integrate into your organisational AD for authentication
- Think about the different types of policies you may need and how flexible is the solution to apply these in different situations, user profiles, locations, etc
- Most of these newer companies have poor reporting, so understand what types of executive and drill down reports you will need as part of your monthly governance reporting, what do you need?
- Cloud is thrown around everywhere and it is important without a doubt. Take into consideration any roaming users you may have and how will their agents function when they are unable to connect back to the SaaS for a week or two? How effective is the agent at detecting and preventing your malware scenarios with no internet connectivity?
- Ask about the retention period they keep your logs for? How far back will they go and do you need to pay extra for a longer retention period?
- Do you have SIEM requirements and what does their integration model look like?
- Lastly, many of you will just test against Win7, Win10, Win2K8 or 12 and maybe MacOS. If you know you need protection on other machines as you have no other mitigating controls, definitely spin-up that OS and test out the agent on older systems. You may not want to admit it, but most companies still have Windows 95 and 98 somewhere.

Some of these agents require a minimum of 2CPUs, a specific service pack or the .net framework. Ensure you verify and test that the agent will be compatible with your older infrastructure. Better to know upfront before you are ready to submit a purchase order.

10. Preparation Summary

In closing, I hope this preparation document provides you will all the fundamentals you need to get started. Here is the quick summary:

1. Diversity of malware samples, choose at least 5 sources
2. Choose from different sets of malware families, download new as well as old samples
3. Select a range of malware file types, just don't use only PEs
4. Now you have your malware, mutate once, twice, three-times
5. Choose your virtualisation platform to test your endpoint images against the malware
6. Consider your user base and ensure you have test scenarios for different profiles
7. Don't forget to test against legacy operating systems as some agents will not be compatible.
8. Test with no internet connection so you can simulate roaming users who don't always have internet connectivity.

Ready to start? Next take a look in the same GitHub repository for the testing how-to's and guides. Have fun and test safe!

11. Terms and Definitions

Term	Definition
PE	Portable Executable is a pre-compiled file that encapsulates all the data, libraries, and functionality into a single build ensuring it can execute independently.
File Mutation	Changing the anatomy of the file so in the case of malware mutation the content is the same but the outside wrapper is different. Intent is trying to fool the software to think it has never been seen before.

12. Revision History

Date	Responsible	Version Number	Summary of Change
2017-08-12	Lidia Giuliano	1.0	Initial Document
2017-08-17	Lidia Giuliano	1.1	Error in email address and added more information to packer section