Understanding and Teaching Heuristics

Randy Abrams
Director of Technical Education
ESET

This paper was originally presented at the AVAR conference in Seoul in 2007,
and included in the Conference Proceedings.

Table of Contents

Abstract

This paper is designed to provide a basic understanding of what heuristics are and how they are used in the anti-malware industry.

Topics covered include signature based detection, generic signatures, passive heuristics, and active heuristics or emulation. A very basic compression algorithm is developed and taught so as to enhance understanding of how compression works and why it poses problems for signature based detection. Encryption and polymorphism are also explained in easy to understand terms and examples.

A variety of false positives from a variety of unspecified products are used to reveal some of the types of thinking that go into creating heuristic approaches.

For those who already understand the subject, the approach used should provide insight into effective methods of teaching complex technical subjects to less technical students, or even to technical people who are simply unfamiliar with the subject.

So who is Hugh Ristic anyway?

Heuristics are used by virtually every anti-malware product in one form or another, but most people do not know what the word means. For those who look up the various definitions they are left without a functional understanding of how heuristics relate to security or why they are important.

Highly technical definitions leave non-technical and semi-technical users substantially uninformed. One successful approach to teaching the subject is through the extensive use of analogies to demonstrate the concepts.

The Oxford English Dictionary defines heuristics as "Enabling a person to discover or learn something for themselves." The American Heritage Dictionary of the English Language defines heuristics as "Of or relating to a usually speculative formulation serving as a guide in the investigation or solution of a problem." Neither of these definitions is very helpful in describing why heuristics are important and how they work.

Visual analogies can be quite useful. The goal of this exercise is to identify a breed of dog that most people will not have a "signature for; that is, most people have never seen one. The "Catahoula Leopard Dog" works well for this purpose.

As we try to decide which is the animal we seek we will probably rule out the fish as we probably unconsciously have decided that we are looking for a mammal. We have "ruled out" the fish. The bear is an interesting example. Computers have to be given very detailed instructions about anything they do. Consider how you would define a dog in such a manner that a person who has never seen either a dog or a bear would not mistake a bear for a

dog upon viewing a picture of a bear. The point is that heuristics can be quite difficult. The Doberman below the fish is generally familiar to people and so it is eliminated from the search.

Few people are fooled by the cat, even though the word "Catahoula" has the word "cat" in it. Most people are expecting a dog with spots since leopards have spots, but the beagle is a known quantity. We arrive at the Catahoula Leopard dog in the lower right corner through the process of elimination, but in order to use this approach we had to use some heuristics.

The National Center for Supercomputing Applications (NCSA) describes heuristics as "Guidelines that a system administrator uses to intervene where the two-phase commit or abort would otherwise fail."

Signature based detection is an example of where a two-phase commit or abort is likely to fail. If a file contains an unknown threat, the signature will fail. Heuristics are used to overcome this by using rules to identify unknown threats. We can simplify the various definitions and arrive at a concise and use definition that is generally functional and understandable by most people:

A rules-based approach to solving problems.

What is the problem to be solved?

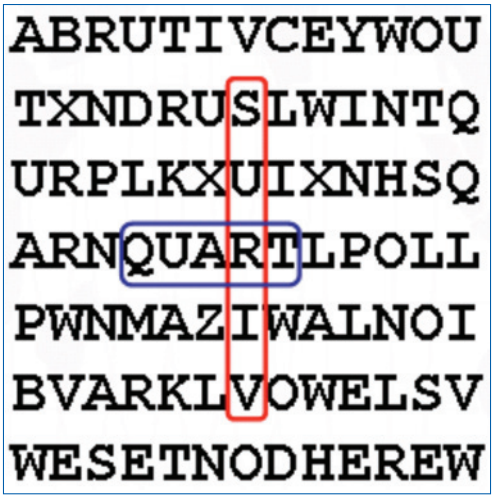
Now that we have a rough idea what heuristics are, we must define the problem we are trying to solve. The problem we as anti-malware vendors face is trying to positively identify malicious software we have not seen before. There have been other approaches to protecting computers against malicious software, but they fail to solve the problem of identification.

Sandboxing and virtualization can help isolate or contain software, but they do not indicate if the software is good or bad. White listing simply attempts to approve something that one hopes is good, but bad programs can be approved as well.

Signature based detection

A visual example of signatures is presented in a grid with many letters. There are several words made up of adjacent letters and participants are asked to find a couple of the words. The next challenge is to find a word they don't know. This, it is explained is where signatures break down.

Now find a word in the grid you don't know... This is where signatures fail as well. If I tell you there is a word "wint" you can find the patter "WINT" in the grid, but unless you know the word you will not identify it until it is taught.



A note on false positives

When a virus scanner detects a file that is clean – should not have been detected – this is called a false positive. People have criticized heuristics in antivirus as being prone to false positives. Signature based detection is prone to some degree of false-positives. One of the reasons for signature based false positives is that signatures do not search for the entire virus, but rather a portion of it. To demonstrate the principal we try to locate one specific elephant, but we cannot remember his name. We search for 'elephant' on Google and arrive at more than 83 million hits. Our elephant is in there, but there are also over 83 million false positives. When we narrow our search to "rude elephant" we arrive at about 966 hits. There are still about 965 false positives, but we are closer. Once we search for "very rude elephant" we find "Rudy the very rude elephant", the one we seek. Virus signatures that are not specific enough will yield false positives, just as a search that is not specific enough yields useless hits. It would have been inefficient to have to type in the whole book, or even a chapter or a paragraph to find Rudy, and it is inefficient to search for more than is required to uniquely identify a threat with traditional signatures.

Some different types of heuristics

There are many approaches to heuristics. Some of these include generic signatures or detection, passive heuristics, and active heuristics. We will examine these in a bit more detail.

Generic Signatures or Detection

Generic detection measure how similar an unknown object is to something already known to be malicious. If an object is similar enough in critical aspects then the unknown object can be asserted to be a variant of the known quantity.

A visual example of this is the two dogs below. If a person has seen an Irish (or red) setter before, but had never seen the English setter to the right, they could probably assert that the dog is some type of setter. Conceptually this is similar to generic detection.

With computer programs let's assume there is a virus "Win32/Makebelieve" and this virus has the following known characteristics:

- Adds 3 specific registry entries
- Contains SMTP engine
- Downloads email address list
- Sends email with subject "World Ends Tonight – Film at 7"

We are scanning a file and discover it has the following characteristics:

- Adds same 3 specific registry entries
- Contains SMTP engine
- Downloads email address list
- Sends email with subject "You won 10 Million Ueros"

The astute reader will note that the word "Euros" is spelled incorrectly. Virus writers are notoriously bad at spelling! We can see that the known virus and the file we are scanning are similar enough in actions that we can assert with some degree of confidence that the unknown file is infected with a variant of Win32/Makebelieve.

False positives can provide insight into how some researchers are attempting heuristic approaches. One approach I encountered involved attempting to identify code that is only present in malicious software. In order to accomplish this task a large collection of good software and a large collection of malicious software are compared and the components only found in the malicious software are isolated. If one of these components is found in a file it is declared to be malicious.

The batch file I was scanning was not harmful, but was detected. The file consisted of only the following lines.

```
echo off
rem ------------------------------------------------------------------------------
rem SETUP.CMD
rem Calls SETUP.VBS to copy files to specified directory and
rem create IIS4 virtual directory to host specified English Query domain
rem Usage: SETUP model drive
rem ------------------------------------------------------------------------------
cscript.exe setup.vbs %1 %2
```

In attempting to identify what caused the detection I trimmed the batch file down to the following before detection ceased:

```
rem -----
rem SETUP.CMD
```

The reason for the detection is that in the researcher's databases they had not encountered a file containing a "REM" statement that also had more than six dash characters following it. The researchers concluded that this heuristic technique would require a much larger clean file set than initially believed to be the case. In theory it is a perfect approach. In practice this is a difficult heuristic to use.

Passive Heuristics

Passive heuristics involve scanning a program and attempting to determine what the program is trying to do. Another false positive is a great way to show that this can actually be done.

The file detected was a Microsoft catalogue file called DEFAULT.CAT. Catalogue files are used by Microsoft as databases of hashes to accomplish digital signing of files without physically altering the file. Catalogue files are nor executable or harmful.

0110	6F00	6C00	6500	7400	6500	3E00	3E00	3E30	o.l.e.t.e.>.>.>0
0120	2130	0906	052B	0E03	021A	0500	0414	9F7F	!0...+........Ÿ□
0130	4896	8E59	F9CD	2621	1779	A7FC	CA83	988C	H–ŽYùÍ&!.y§üÊƒ˜Œ
0140	3FB5	3062	060A	2B06	0104	0182	370C	0202	?µ0b..+....‚7...
0150	3154	3052	1E4C	007B	0044	0045	0033	0035	1T0R.L.{.D.E.3.5
0160	0031	0041	0034	0032	002D	0038	0045	0035	.1.A.4.2.-.8.E.5

In this case the scanner identified the code CD 26, which for a DOS ".COM" file would indicate a direct write to the hard drive. The false positive was corrected however, the scanner did accurately identify a potentially dangerous program. The actual detection indicated that the file was potentially destructive and if the .cat was renamed to .com and executed on a DOS machine it may well have been.

Active Heuristics

Active heuristics are referred to by a variety of names by different vendors. Some call the technique "sandboxing", others call it "virtualization" or "emulation." In all cases the idea is to create a safe virtual environment, run the code to be inspected and watch the behaviors in order to assess risk. In addition to being able to observe behaviors, in some cases runtime compressed or encrypted files can be made to decompress or decrypt themselves. Polymorphic viruses can sometimes be detected with this technique as the fact that they look different does not mask their true behavior.

In order to understand how this heuristic approach can be effective in defeating compression and encryption, as well as polymorphism, a short course is in order.

Understanding Compression

Fundamentally compression uses symbols to represent patterns.

For example, we have all seen counting with lines. I II III IIII ~~IIII~~. The number five can be represented as "5". This replaces 5 symbols with just one.

We are going to quickly create our own compression code. Computers only use ones and zeros. The exact sequence of ones and zeros is the only difference between a document, a spreadsheet, a picture and a sound file. Below is the "file" we are going to compress.

0000 0000 0111 0111 0111 0110 0010 0010 0010 0010 0010 0001 0001 0001

The "file" has been broken up into sets of four digits to simplify the explanation. We have 14 sets of 4 digit sequences, or 56 characters. We need to say the same thing with fewer characters.

Our compression algorithm, or program, will use a 2 digit number. The first digit will be the number of identical consecutive sets of 4 digits. The second number will identify each unique 4 digit set. Here is our "key":

0000=0 0001=1 0010=2 0011=3 0100=4 0101=5 0110=6 0111=7 1000=8 1001=9

We can see that 0000 0000 is 2 consecutive identical sets, and looking at our key we have decided to represent the specific set with the single character "0" so 0000 0000 becomes 20. Looking at our "file" the next three sets of digits are consecutive, identical sets that we have defined as "7." 0111 0111 0111 becomes 37 with our algorithm.

If we complete the process our file becomes compressed into ten characters "20 37 16 52 31." From 56 characters down to 10 is pretty good compression. Now look at "20 37 16 52 31" and find the pattern or "signature" 0110 0010! It cannot be done. This is how compression breaks traditional virus signatures. With runtime compressed malware the malicious software will often use an unknown compression algorithm, but it decompresses itself when it runs, as it must be decompressed in order to execute and do whatever it was intended to do. If we put that program in an emulator then in some cases we can make the sample decompress itself. After the sample has decompressed itself, both signatures and heuristic methods can be used to determine if the program is a threat.

Understanding Encryption

Compression is a type of encryption. The goal of compression is to save space, but the result of compression is encryption. The goal of encryption is to hide data. Encrypted files can get smaller, but they can also get larger or stay the same size.

Consider the following familiar encryption algorithm.

A=1 B=2 C=3 D=4 E=5 F=6 G=7 H=8

In this example the word "BAG" becomes "217." Did you ever stop to think that when you see a phone number it might actually not be a phone number, but rather a couple of encrypted words?

Let's try one more encryption algorithm

A=100 B=101 C=102 D=103 E=104 F=105 G=106

In this example, the word "BAG" goes from a 3 character word to the 9 character sequence "101100106".

In both cases if you are searching for the word "bag" you will miss it… unless you know it is encrypted and what the algorithm is.

For encrypted files to execute they must be decrypted, or decrypt themselves at runtime. In an emulator a self-decrypting file can be made to decrypt itself.

Poly want a cracker?

Polymorphic viruses do the same thing each time they replicate, but they look different. The technique breaks traditional signatures, but since the emulator is looking at the actual behaviors is does not matter what the file looks like. The concepts of doing the same thing but looking different may not be intuitively obvious to some people, so let's look at some examples of things that do, say, or mean the same thing, but look different.

If I am searching for the specific pattern (traditional signatures look for specific patterns) "5", then I see it here:

5

Now if I write out 5 as 2+3 then I have said the same thing, but it looks different. In fact there are an infinite number of ways to "say 5" without using the specific symbol.

1-3+9*(1037/32*(32/1037))+(11-33)+(10*2) also means 5!

I could tell you to Run 5 minutes, rest 10 minutes, and walk 3 minutes. If I say the same thing, but this time I say Run 5 minutes, rest 5 minutes, rest 5 minutes more, and walk 3 minutes, I have told you to do exactly the same thing, but it looks different. If you only care about the behaviors of running resting and walking then it really doesn't matter how it looks. Active heuristics care about behaviors.

Active Heuristics Continued…

Active heuristics can identify if a program is listening on ports, writing to specific registry keys, sending email, using IRC, writing to system files, and a wide variety of other behaviors.

Another false positive can show this in action. A file I was scanning while working at Microsoft, "Timeserver.exe," was identified as having possible back door functionality. Below is the actual report.

…\Windows for Smartphones\samples\win32\CMTime\TimeServer\ TimeServer.exe
-> Virus W32/Malware (Connect port 00037 [DGRAM], IP 000.000.000.000.
Possible backdoor functionality [UNKNOWN] port 00000037.

The company in question fixed the false positive, but the product did accurately identify the behavior of the program listening on a port, just as a backdoor would.

Minimizing Heuristic False Positives

The keys to minimizing heuristic false positives are to create very smart and precise rules, and then to give them highly intelligent weighting. Just as if you were trying to determine if a person walking into a bank was a crook, there are a variety of observations you would make, but not all are as important in determining if the person is bad. If the person is wearing a shirt you would expect that, but if they are wearing a mask it is more suspicious. If the person has a gun that might not carry a lot of weight if the person is a known security guard.

An example of how this works follows.

Is the program encrypted? 1 point * 1.5

Encryption may be assigned a multiplier of 1.5 because it can be suspicious, but copy protected programs may also be encrypted.

Does it open a port to listen? 1 point * 2

Perhaps listening on a port is a bit more suspicious than encrypted programs. These values are being arbitrarily assigned in this presentation to demonstrate concepts.

Does it write to existing files? 1 point * 3

Depending on what the file is this activity may be more suspicious. This activity may also make a threat more similar to a known threat for generic detection.

Does it write to the registry? 1 point * 1

What doesn't write to the registry today? This is an important behavior to note, but not worth a lot by itself. Where it writes to the registry becomes more interesting.

Does it parse files in a temporary directory? 1 point * 5

This would be a highly suspicious behavior in most cases. Mass mailing worms sometimes do this,

When the scan is done the product will add up the total number of rules and values and if a predetermined threshold is exceeded the file is declared to have some level of risk. It is possible to have multiple thresholds. At one level a file might be called "potentially destructive", where at another threshold the file is declared to be a type of password stealer or keystroke logger.

It is the combination of rules and weighting that allow a product to proactively identify new threats while maintaining low levels of false positives.

Detections of threats

Signature based detections have a very unique message. Something like "Win32/Stration.YQ worm" is very specific and indicative of a signature based detection. The slightly different

"Win32/Nuwar.Gen worm" uses ".gen" to indicate that this has been detected with generic detection. Another detection that is indicative of the use of heuristics is "probably a variant of Win32/PSW.Maran Trojan". This detection is not an exact identification of a threat, but an indication that the threat looks much like the password stealing (PSW) Maran Trojan. When you see something like "a variant of Win32/Exploit.WMF Trojan" it is also a heuristic detection. "Probably unknown NewHeur_PE virus" is a heuristic detection as well. Signature based detections are never "probably", "a variant", "possibly", "potentially", or anything but exact.

How effective are heuristics?

In independent testing, products have detected as much as 86% of "in-the-wild" viruses with heuristics alone. For full zoo collections that contain viruses, worms, and Trojans, product have been able to detect 50 to 60 percent of the threats with heuristics. This indicates that heuristics can help to prevent infection by brand new threats, but currently cannot deal with all threats. There is no silver bullet. No single technology will protect against everything, however, heuristics are a very useful component in securing against brand new unknown threats. As the anti-malware industry continues to study heuristics improvements in detection will accompany decreases in false positive rates. Already today, products with strong heuristic technologies are rivaling more signature reliant products in terms of false positive rates.

A little Trivia!

Catahoula Leopard Dogs have nothing to do with cats, hula dancers, or leopards.

Catahoula Leopard Dogs are used for herding.

The Catahoula Leopard Dog is the official dog of the state of Louisiana.

This is the first recorded instance of a Catahoula Leopard Dog being used to teach computer science at an international security conference!

Corporate Headquarters

ESET, spol. s r.o.
Aupark Tower
16th Floor
Einsteinova 24
851 01 Bratislava
Slovak Republic
Tel. +421 (2) 59305311
www.eset.sk

Americas & Global Distribution

ESET, LLC.
610 West Ash Street
Suite 1900
San Diego, CA 92101
U.S.A.
Toll Free: +1 (866) 343-3738
Tel. +1 (619) 876-5400
Fax. +1 (619) 876-5845
www.eset.com

UATH20081209