MAY 2008

virus BULLETIN

Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

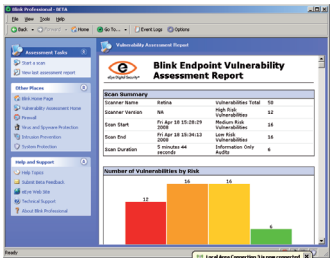
GROUP MENTALITY

Malware researchers are frequently faced with huge collections of files that must be analysed to determine whether or not they are malware. Grouping the files according to their binary similarity can save time and effort. Víctor Álvarez discusses the algorithms that can be used to give the malware researcher a helping hand.

page 4

BLINKING GOOD

John Hawes takes an in-depth look at the security features of eEye Digital Security's Blink Professional and finds a solid package with impressive breadth of power.



page 14

vbSpam supplement

This month: anti-spam news and events, and Sorin Mustaca describes a method for delivering protection against phishing websites.

'Banking organizations have failed to pledge that they will stop sending emails that add to the confusion.'

Helen Martin, Virus Bulletin

ONLINE BANKING CALL TO ARMS

According to a recent report released by UK payments industry association APACS, the rate of phishing attacks in the UK has increased dramatically over the last 12 months, with the number of incidents reported during the first quarter of 2008 up 200 per cent on the same period last year.

At least some degree of that increase may be due to an increased awareness among the public of phishing attacks and how to spot them (and consequently report them) – a theory supported by the fact that the number of people either deleting or taking no action when receiving a phishing email increased from 75 per cent in 2006 to 82 per cent in 2007 and the fact that losses from online banking fraud decreased by a third from £33.5m in 2006 to £22.6m in 2007.

However, it is clear that phishing is still big business – and users of online banking systems are advised by APACS that they should 'just remember that your bank will never send you emails asking you to disclose PIN numbers, login details or complete passwords'.

But are the banks themselves doing enough to help their customers steer clear of online fraud? A new banking code released by the British Bankers Association (BBA) last month included advice for customers on how to avoid falling victim to identity theft and online fraud. The suggestions set forth constituted sound, well-considered advice both in terms of physical security (e.g. don't keep your cheque book and cards in the same place; shred any printed information about your accounts; notify the bank if an expected statement or letter is not received) and online security (e.g. use up-to-date anti-virus and anti-spyware products and a personal firewall; never follow a link from an email directly to a bank or building society; treat emails claiming to be from your bank or building society with caution).

Much was made in the media of a cautionary note contained in the code, which warned that if customers fail to follow this set of guidelines to a reasonable degree banks may hold the customer responsible for any losses that can be deemed to have resulted from such lapses in security.

In practice, of course, it is unlikely that failure to follow the advice to the letter will result in customers being asked to foot the bill for losses – the burden of proof lies with the bank to demonstrate that the customer has behaved unreasonably or irresponsibly and it is unlikely that banks will invest the resources necessary to prove in individual cases that computers are not adequately secured. There is a fine line between scaremongering and giving users an incentive to take security more seriously, and the BBA code treads the line carefully – but in order for this ruling to have a positive effect it must be backed up with readily available information on what adequate protection looks like and how the average user can achieve it.

What was disappointing about the new banking code, and indeed remains disappointing in the banking and financial services industry as a whole, is that, while users are urged to 'always be suspicious of unsolicited emails that claim to be from your bank', banking organizations have failed to pledge that they will stop sending emails that add to the confusion. With phishing emails becoming increasingly stealthy – some even including warnings about the dangers of phishing – emails that are genuinely sent by banks (particularly those that contain links to the banking sites) compound the issue. A concerted and global effort to address the content and style of emails sent by banking organizations would go a long way towards helping reduce confusion.

VB has invited a panel of security experts from the banking and financial services sector to speak at VB2008 on the efforts their organizations are making to counter online fraud – it is hoped that such an open forum will facilitate the exchange of ideas and sharing of knowledge between the banking and anti-malware communities. VB2008 takes place 1–3 October 2008 in Ottawa, Canada. For details of the rest of the programme and online registration see http://www.virusbtn.com/conference/vb2008.

NEWS

MALWARE GETS TERMS OF USE

Concerned about the trustworthiness of their customers, creators of malicious software have started taking the precaution of including licence agreements in the packages they distribute in the underground.

According to researchers at *Symantec*, a EULA contained in the Infostealer.Banker.C or 'Zeus' malware package specifies that the purchaser of the malware may not distribute the product for any business or commercial purposes, may not disassemble or study the binary code of the bot builder, may not use the control panel as a means to control other botnets, must not send any portion of the product deliberately to anti-virus companies, and must agree to pay for any update to the product other than the fixing of programming errors.

In an attempt to make sure users abide by these rules, a warning note is added which advises the user that, should they violate the terms of the agreement and be found out, then not only will they lose any technical support for the product, but the binary code of their bot will immediately be sent to anti-virus companies.

The fact that the package was being traded freely in underground forums shortly after it was released suggests that it is as hard to enforce the terms of a licence agreement for malware as it is for legitimate software – or maybe that the user of a malicious software package is as unlikely to read a EULA as users of legitimate software.

GRAND THEFT PERSONAL INFORMATION

Large volumes of spam were spotted late last month coinciding with the release of the latest version of the computer game *Grand Theft Auto*. Taking advantage of the popularity of the game, spammers sent messages offering free entry to a prize draw to win a *PlayStation 3* loaded with the new version of the game. Of course, the prize draw did not really exist and the spammed emails contained various malicious programs designed to infect the recipients' computers and steal personal information.

Grand Theft Auto IV was released to a frenzied reception last month – UK newspapers reported at least two instances of violence having flared up among queues of customers waiting outside high-street shops to get their hands on a copy. With such an eagerly anticipated product – and retailers unable in many instances to fulfil demand – the opportunity was ripe for scammers to exploit. According to UK-based mail filtering company *ClearMyMail.com*, more than half of the spam it blocked on the day of the game's release related to *Grand Theft Auto*, with the majority of those messages containing viruses and spyware.

Prevalence Table – March 2008

Malware	Type	%
Cutwail/Pandex/Pushdo	Trojan	48.29%
NetSky	Worm	22.16%
OnlineGames	Trojan	10.20%
Mytob	Worm	8.41%
Virut	Virus	4.36%
Mydoom	Worm	4.35%
Bagle	Worm	3.65%
Zafi	Worm	2.27%
Agent	Trojan	1.73%
Small	Trojan	1.50%
Stration/Warezov	Worm	1.06%
Grew	Worm	1.03%
Sality	Virus	0.60%
Zlob/Tibs	Trojan	0.50%
Mywife/Nyxem	Worm	0.49%
Bugbear	Worm	0.23%
Grum	Worm	0.20%
VB	Worm	0.17%
Klez	Worm	0.14%
PrettyPark	Worm	0.12%
Bagz	Worm	0.12%
Delf	Trojan	0.11%
Nuwar/Peacomm/Zhelatin	Trojan	0.10%
Doombot	Worm	0.09%
Nahata	Worm	0.09%
Sdbot	Worm	0.08%
Areses/Scano	Worm	0.07%
Fleming	Worm	0.07%
Autorun	Worm	0.06%
Lineage/Magania	Trojan	0.06%
Brontok/Rontokbro	Worm	0.05%
Alman	Worm	0.05%
Parite	Worm	0.05%
Others[1]		0.54%
Total		100.00%

[1]Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

FEATURE 1

ALGORITHMS FOR GROUPING SIMILAR SAMPLES IN MALWARE ANALYSIS

Víctor M. Álvarez
PandaLabs, Spain

Malware researchers often need to group large sets of files according to their binary similarity. For instance, they are frequently faced with huge collections of files that must be analysed to determine whether or not they are malware. Such collections generally contain files which are very similar, but which do not match exactly, while many other files bear no relation to the rest. In such a situation, grouping the files according to their binary similarity can save a lot of time and effort. In this article we will discuss some algorithms that can be used for this purpose.

MEASURING FILE SIMILARITY

The first thing we need to do before we can implement an algorithm for grouping similar files is to define a way to measure the grade of similarity between two given files. From now on, we will refer to this grade of similarity as the *distance* between the two files. The more similar the files, the smaller the distance between them and vice-versa.

A good way to measure the distance between two files is to calculate the length of their longest common subsequence.

Let A be a sequence of symbols of length m, then a subsequence of A is another sequence, A', of length $n \leq m$ that can be obtained by removing zero or more symbols from A. For example, *abce*, *bcde*, *bad*, and *ade* are all subsequences of *abacde*. Obviously, the longest common subsequence (LCS) of A and B is the longest subsequence of A that is also a sub-sequence of B.

Although the length of the longest common subsequence (LLCS) is theoretically a good measure of file similarity, it has a major drawback in practice: its time complexity. Several algorithms for solving the LCS and LLCS problems have been proposed by different authors, including Hirschberg [1], Hunt and Szymanski [2], Kuo and Cross [3] and some others, but all of them run in quadratic time. Taking into account the large number of file comparisons needed to cluster a large set of files, it is obvious that a linear time algorithm for calculating file distances is preferred, even at the expense of losing accuracy in the measurement.

That's when delta algorithms come into play. The purpose of delta algorithms is to receive two files and generate a set of instructions that can be used to convert one of the files into the other. In other words, given the reference file A and the target file B, a delta algorithm is composed of two functions Δg and Δp such that:

$$\Delta g\ (A,B) = C \ \text{ and } \ \Delta p\ (A,C) = B$$

The function Δg generates a set of instructions C that can be used in conjunction with A to obtain B using Δp. This kind of algorithm is widely used for performing incremental data backups, distributing software updates, and many more situations where changes must be made to existing data and it is more efficient to send or store just the difference between versions than the whole new set of data.

Delta algorithms are strongly related to the LCS problem. The relationship becomes more evident when considering that the LCS and the string-to-string correction problem (STSC) are dual (i.e. the two problems are complementary, a solution to one of them determines a solution to the other).

The STSC problem consists of finding the minimum number of insertions and deletions necessary to convert a given sequence of symbols into another. In fact, some delta algorithms rely on solving the LCS/STSC problem to find the shortest possible delta file, but many of them don't pretend to find the optimal set of instructions, they just try to find one that is good enough, sacrificing solution quality in favour of execution speed.

The algorithm we will present here to calculate the distance between two files is inspired by xdelta [4]. Both the reference file F_1 and the target file F_2 are scanned simultaneously with an offset of W bytes. The scanning of F_2 starts when the first W bytes of F_1 have been scanned (in our tests we set W to 64 KB with good results).

On each increment of the current position of F_1 four bytes are read from the file, passed through a hash function H, and the result is used as an index on the hash table T where the current position of F_1 is stored. The hash table does not chain hash collisions; when a collision occurs the previous value is overwritten.

When scanning F_2 four bytes are also read on each iteration. The hash table is used to find out if there is a match in F_1 for these four bytes. If a match is found, the current position of F_2 advances until the end of the matching block – this would generate a copy instruction in the output of the xdelta algorithm. If there is no match the current position of F_2 is incremented. This would generate an insert instruction. The distance d between the two files

is based on the number of instructions c that would be generated, and the size of both files s_1 and s_2, following the expression:

$$d = \frac{|\, s_1 - s_2 \,| + 2c}{s_1 + s_2}$$

It must be said that this distance is not symmetrical (i.e. $d(A,B) \neq d(B,A)$), although it tends to be more symmetrical when the files are more similar. For files with very dissimilar sizes, but which are still similar in some way (consider, for example, the case of comparing a file with a truncated version of itself), this asymmetry could lead to very different results depending on the order of comparison. To avoid this discrepancy, we decided that the biggest file would always be the reference file F_1, and the smallest one the target file F_2. This is because the delta algorithm generates fewer instructions when trying to convert a large file into a smaller one than the opposite way round.

Here is the algorithm for calculating the number of instructions c in more detail:

(1) while F_1 [i] = F_2 [i] do i ← i + 1
(2) if (i = s_2) return 0
(3) for i ≤ j < i + W do
    x ← four bytes of F_1 at offset j
    T[H(x)] ← j
    j ← j + 1
(4) c ← c + 1
    x ← four bytes of F_2 at offset i
    k ← T[H(x)]
    if (k is null)
        x ← four bytes of F_1 at offset j
        T[H(x)] ← j
        j ← j + 1
        i ← i + 1
    else while (F_1 [k] = F_2 [i]) do
        x ← four bytes of F_1 at offset j
        T[H(x)] ← j
        j ← j + 1
        i ← i + 1
        k ← k + 1
(5) repeat (4) until end of F_2
(6) return c

GROUPING SIMILAR FILES

Once we have defined a metric to measure file similarity, the next part of our problem is to find a method for grouping similar files together. This is when clustering algorithms become helpful. In general terms, clustering algorithms are aimed at partitioning a set of objects into subsets according to their proximity, as defined by some distance measure.

An incredible number of clustering algorithms have been developed over the years for different purposes and employing a variety of techniques. Terms like 'K-means', 'fuzzy C-means' and 'hierarchical clustering' populate hundreds of academic and research papers across a broad spectrum of fields, and there are plenty of choices for a clustering algorithm. However, the characteristics of the problem we wanted to solve meant that it was not difficult to choose an appropriate algorithm for our needs.

From the very beginning it was obvious that algorithms like K-means, which rely on finding a centroid for each cluster, were not suitable in this case. A centroid is nothing more than a central point for a cluster or group of objects, which is not necessarily one of the objects. Depending on the nature of the objects we want to cluster, establishing the centroid for a cluster could be an easy or a very difficult task. Calculating the centroid of n points in a Euclidean space is straightforward, but establishing a centroid for a group of files is a whole new problem on its own. K-medoids, another clustering algorithm very similar to K-means, does not have this problem because the central point for each cluster (or medoid) is not external to the objects, but one of the objects itself. However, both algorithms need to know in advance the number of clusters into which the data will be partitioned – a requirement that makes these algorithms unsuitable for our purposes.

On the other hand, hierarchical clustering algorithms seemed very appropriate for the task. There are different variants of hierarchical clustering algorithms, but we will concentrate on agglomerative single linkage clustering. This algorithm starts by placing each object (a file in our case) in a separate cluster. At each stage of the algorithm the two closest clusters are merged together until a certain number of clusters is reached, or until the distance between the two closest clusters exceeds a predefined value. The distance d between two clusters C_n and C_m is defined as the minimum distance between any object from C_n and any object from C_m.

In a more formal way:

$$d(C_n,C_m) = \min \{\, d(x,y) \mid x \in C_n,\, y \in C_m \}$$

In order to see the algorithm in a more detailed way let's assume that we have N files F_n ($1 \leq n \leq N$) to cluster, C

denotes a cluster, and d_f is the minimum distance allowed between any two clusters of the final result, then:

(1) Construct matrix D of dimensions $N \times N$ such that:

$$D_{nm} = d(C_n,C_m) = d(F_n,F_m) \quad n \leq N, m < n$$

(as $d(F_n,F_m) = d(F_m,F_n)$, D is symmetrical and only the lower half is used)

(2) Find clusters C_x and C_y such that:

$$d(C_x,C_y) = \min \{ d(C_n,C_m) \mid n \leq N, m < n \}$$

(3) Merge C_x and C_y into a single cluster $C_x \cup C_y$ and reconstruct D by removing the two rows and columns that correspond to C_x and C_y and adding a new row and a new column for the newly created $C_x \cup C_y$. Now $N \leftarrow N-1$ and the distance from any existing cluster C_p to the new cluster $C_x \cup C_y$ is given by:

$$d(C_p, C_x \cup C_y) = \min \{ d(C_p,C_x), d(C_p,C_y) \}$$

(4) Repeat from step (2) while there is more than one cluster and the distance between the two closest clusters does not exceed d_f.

Successive steps of the algorithm applied to a set of five files are shown in the tables below. The minimum distance in the first matrix is between clusters C_2 and C_4. The corresponding columns and rows (in red) are deleted from the first matrix and a new row and column (in green) are added to construct the second. The algorithm continues until the distance between the two closest clusters is greater than $d_f = 0.25$.

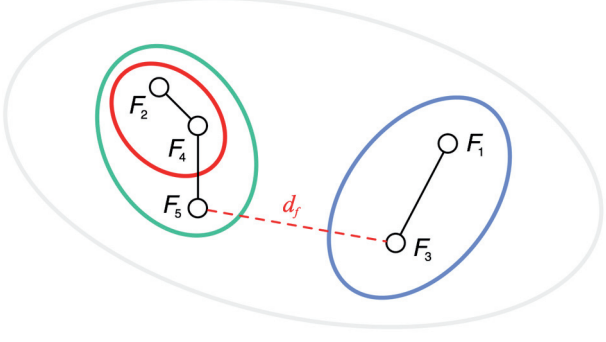
	C_1	C_2	C_3	C_4	C_5
$C_1 = \{F_1\}$	0				
$C_2 = \{F_2\}$	0.35	0			
$C_3 = \{F_3\}$	0.19	0.32	0		
$C_4 = \{F_4\}$	0.29	0.12	0.31	0	
$C_5 = \{F_5\}$	0.34	0.18	0.28	0.33	0

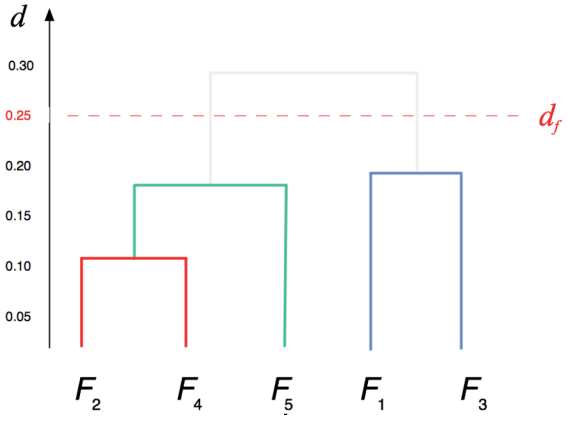
	C_1	C_3	C_5	$C_{2\cup4}$
$C_1 = \{F_1\}$	0			
$C_3 = \{F_3\}$	0.19	0		
$C_5 = \{F_5\}$	0.34	0.28	0	
$C_{2\cup4} = \{F_2,F_4\}$	0.29	0.31	0.18	0

	C_1	C_3	$C_{2\cup4\cup5}$
$C_1 = \{F_1\}$	0		
$C_3 = \{F_3\}$	0.19	0	
$C_{2\cup4\cup5} = \{F_2,F_4,F_5\}$	0.29	0.28	0

	$C_{2\cup4\cup5}$	$C_{1\cup3}$
$C_{2\cup4\cup5} = \{F_2,F_4,F_5\}$	0	
$C_{1\cup3} = \{F_1,F_3\}$	0.28	0

The result of this clustering algorithm is a hierarchical representation of similarity between the files, which is usually called a dendrogram. In a dendrogram each leaf of the tree is an element of the set being clustered, and each node in the middle of the tree represents an association between two objects, one cluster and one object, or two clusters. The picture below, showing the dendrogram corresponding to our previous example with five files, speaks for itself.





Notice that as the algorithm is interrupted when the distance between the two closest clusters is greater than d_f, a set of sub-trees is obtained instead of a single tree. All elements that are leaves of the same sub-tree are also members of the same cluster in the final result of the algorithm.

OPTIMIZATION

File clustering is a costly process. The number of file comparisons that must be done to fill the initial distance matrix for the clustering algorithm is $N \times (N - 1) / 2$. As the number of files grows linearly, the number of comparisons grows quadratically. The file comparison algorithm is not very costly in itself, it only depends linearly on the size of the files being compared, but it requires the content of both files to be read completely, incurring a large number of I/O disk operations with the obvious performance degradation that such operations impose. For this reason it is necessary to avoid unwanted file comparisons as much as possible.

A good heuristic is to avoid comparing files whose size is quite different. As a rule of thumb it can be said that a pair of files with sizes s_1 and s_2 should be compared only if they satisfy:

$$\frac{|s_1 - s_2|}{s_1 + s_2} < r$$

Here, r is a certain adjustable ratio that can be raised or lowered depending on the number of files, the characteristics of the hardware, and other factors. If the sizes do not satisfy the expression above, they are not compared and the distance between them is considered to be infinite.

Another possible optimization consists of aborting the distance computation when it is about to exceed a maximum value. As mentioned above, the clustering algorithm stops when the minimum distance between all existing clusters is greater than a given value we called d_f. Therefore, distances greater than d_f do not influence the final result – their values are irrelevant as long as they are greater than d_f. But the distance between two clusters is by definition the distance of their nearest elements, which means that it is also the distance between some pair of files in the set. When calculating such distances, the process can be interrupted at the very moment it exceeds d_f. This saves a considerable amount of file read operations, improving the general performance of the algorithm.

CONCLUSIONS

This article is based on well-known and studied algorithms; there is nothing genuinely new in what has been described here. However, I hope it may have served to show that mixing together existing algorithms and techniques in a creative way can sometimes help to improve the tools we have available to make our job easier.

I also hope this article may be relevant beyond the anti-virus industry, because grouping files according to their binary similarity has a broader range of applications than just sample categorization in malware analysis.

REFERENCES

[1] Hirschberg, D.S. A linear space algorithm for computing maximal common subsequences. Communications of the ACM, volume 18, no. 6 (1975) pp.341–343.

[2] Hunt, J.; Szymanski, T.G. A fast algorithm for computing longest common subsequences. Communications of the ACM, volume 20, no. 5 (1977) pp.350–353.

[3] Kuo, S.; Cross, G.R. An improved algorithm to find the length of the longest common subsequence of two strings, ACM SIGIR Forum, volume 23, no. 3–4 (1989) pp.89–99.

[4] MacDonald, J.P. Versioned file archiving, compression, and distribution. University of California at Berkeley (1999). See http://citeseer.ist.psu.edu/macdonald99versioned.html.

[5] Burns, R.C.; Long, D.D.E. A linear time, constant space differencing algorithm. Proceedings of the International Performance, Computing and Communications Conference (IPCCC), IEEE, (1997).

[6] Hunt, J.J.; Vo, K-P.; Tichy, W.F. Delta algorithms: an empirical analysis. ACM Transactions on Software Engineering and Methodology, volume 7, no. 2 (1998) pp.192–214.

[7] Jain, A.K.; Topchy, A.; Law, M.H.C.; Buhmann, J.M. Landscape of clustering algorithms. Proceedings of the 17th International Conference on Pattern Recognition, 2004, volume 1, issue 23–26 (2004) pp.260–263.

[8] Jain, A.K.; Murty, N.M.; Flynn, P.J. Data clustering: a review. ACM Computing Surveys, volume 31, issue 3 (1999) pp.264–323.

FEATURE 2

METAMORPHIC AUTHORSHIP RECOGNITION USING MARKOV MODELS

Mohamed R. Chouchane, Andrew Walenstein, Arun Lakhotia
University of Louisiana at Lafayette, USA

Automated code morphing techniques can make malware recognition difficult [1]. Morphed malware can be detected by recognizing invariant runtime behaviours, or by first normalizing the programs to remove the variations introduced by the morphing engine [2]. While effective, these methods are computationally expensive to apply to every object scanned.

We propose a fast method that can be used to decide whether a binary might be a variant of a known item of metamorphic malware. The key idea is to treat the morphing engine as an author, and then use its morphing characteristics to decide whether a suspect program is a variant of the original, or 'Eve' version.

INTRODUCTION

Since the time of the Morris worm it has been suggested that it may be possible to trace a program to its authors by noting features of the program that are likely to fit the output profile of what those authors may generate [3]. But morphed programs are the output of a certain kind of 'author' – the morphing engine. This observation leads us to ask: is it possible to recognize morphed malware by virtue of recognizing its author in the form of the morphing engine? We propose a method for recognizing metamorphic malware – which may generate successive transformations of itself – using mechanisms similar to those used in determining the authorship of ordinary text.

Our approach starts with a model of the metamorphic engine as a probabilistic generator of text. We employ a simple model for closed-world probabilistic instruction-substituting, metamorphic malware.

With the model in hand, some method of selecting authorship features is needed, and a method is required for matching the expected features to those found in a program whose authorship is in question. We propose to use instruction frequency vectors, or IFVs, as the program abstractions to compare. An *a priori* computable transition matrix is used to construct predicted IFVs for different generations of the metamorphic malware. This transition matrix is computed using a model of the metamorphic engine. The predicted IFVs can then be compared to the IFV of a suspect program to determine whether it is a descendant of the Eve. Markov theory [4] is used to formalize the frequency vector comparison approach.

PROBABILISTIC AUTHOR MODEL

We are interested in recognizing variants of closed-world probabilistic instruction-substituting metamorphic malware. This class of malware carries a metamorphic engine that uses a fixed, finite set of transformation rules, each of which maps an instruction (the left-hand side) to a set of possibly larger code segments (the right-hand sides).

An example of this type of rule set appears in Figure 1. In the example there are two rules, which for comparison purposes in the analysis below have identical left-hand sides. When the left-hand sides are found, the rule is fired with the probability of 0.2. There are two right-hand sides which are selected at probabilities 0.3 and 0.7, respectively.

P	left	right1	P1	right2	P2
0.2	`mov reg, imm` →	`mov reg, imm` `add reg, imm`	0.3	`mov reg, imm` `sub reg, imm`	0.7

Figure 1: Probabilistic instruction substitution.

These rules are typically carried as data in the malware code. They are used by the engine, perhaps along with other transformations, to substitute occurrences of the left-hand sides of the rules probabilistically with one of their corresponding right-hand sides. The probabilities are assumed to be fixed and exactly learnable from the description of the engine. While this is a simple model of metamorphic malware, it captures the essential elements of probabilistic substitution.

Metamorphic malware gives rise to variations through the generation of descendants. Let M denote some metamorphic engine and (M, x) a malicious program using M to transform its own code. The set of all programs (M, y) into which (M, x) can possibly be transformed at the end of a run of M are called the 'first-generation descendants of (M, x)'. More generally, for positive integer n, an $(n + 1)$-generation descendant of (M, x) is a first-generation descendant of an nth-generation descendant of (M, x). The descendants of a given (M, x) are often called variants of each other.

PREDICTING FEATURE FREQUENCY VECTORS

A set of features of a given program must be used to determine whether it has been authored by a given

morphing engine. Ideally, the set of features selected is such that the idiosyncrasies of the authoring program – its specific generation characteristics – can be detected. In this paper we use instruction forms, which are assembly-level instructions in an abstract form. Depending upon the implementation chosen, these could use merely the operations themselves (i.e. without parameters or prefixes), or employ instruction 'templates' that keep parameter counts and indexing modes but ignore all other details. As will be shown, matching involves comparing the frequencies of such features against the frequencies we can expect from the metamorphic malware.

Let P denote a program and n the number of distinct instructions occurring in P. The instruction frequency vector of P, denoted $IFV(P)$, is the n-tuple of pairs, each of which consists of an assembly operation and the frequency (or count) of that operation in P. For example, consider a code segment s whose operations (ignoring all parameters and prefixes in this case) follow the sequence: mov, add, push, mov, mov, add, pop. Then:

$IFV(s)$ = [(mov, 3), (add, 2), (push, 1), (pop, 1)]

For any sequence of generations of malware created by a probabilistic morpher, the IFVs will evolve in a predictable way. Our model of probabilistic morphing does not consider any previous transformations when selecting a transformation rule to apply (i.e. it has no 'memory'). It can therefore be treated as a first-order Markov process, and the sequence of descent is a Markov chain. Each application of a transformation rule by the engine serves to transform the IFV for the program. Figure 2 illustrates an example of such a transformation using instruction 'templates' as the instruction forms being used. A new generation is created when the metamorphic malware applies one or more such rules probabilistically on one of its variants. Producing such a new generation is, in our Markov model, taken to be a state transition. The IFV of the original variant (our 'Eve') is the initial state.

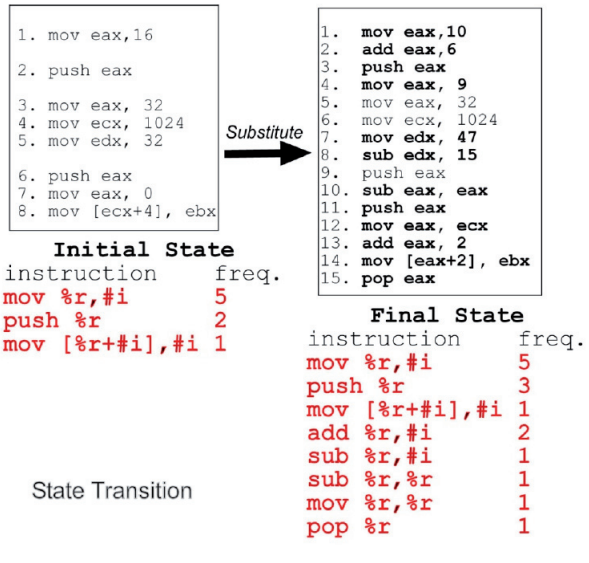


Figure 2: IFV transition induced by metamorphic transformation.

Utilizing Markov theory has several advantages. It provides clear formalization of the computations needed to predict the evolution of IFVs as the metamorphic engine produces new generations. Moreover, Markov theory has identified certain interesting classes of chains and ways of using a chain's transition matrix to infer useful information about the process it represents. Two of these results suggest clearly how and when the IFV transition matrix (whose computation is discussed further below) can be used to assist in the detection of descendants of a given malware variant.

Distribution prediction using the successive powers of the transition matrix

Typically, Markov chains are started in a state determined by a probability distribution on the set of states, called a probability vector. Let u denote a probability vector which holds the initial probabilities of a malware's IFV. Then T^n is the nth power of T, and $T_{ij}^{(n)}$ is the i,j-th entry of T^n. The powers of T are known to give interesting information about the evolution of these IFVs from one malware generation to the next: for any positive integer n, $T_{ij}^{(n)}$ gives the probability that the chain, starting in state s_i, will be in state s_j after n steps. More generally, if we let $u^{(n)} = uT^n$, then the probability that an nth-generation malware descendant has IFV_i after n transitions is the ith component of uT^n.

Convergence towards a stationary state distribution

For every transition matrix T of a Markov chain with a finite space, there exists at least one stationary distribution, i.e. a row vector s satisfying $s = sT$. Furthermore, if T is irreducible and aperiodic, then it has a unique, *a priori* computable stationary distribution given by $\lim T^n = 1s$, where 1 is a column vector all of whose entries equal 1.

Hence, for a piece of malware whose starting probability distribution on the set of IFVs happens to be a stationary distribution for its engine's IFV transition matrix, the corresponding states of the elements of every generation of descendants will be distributed as indicated by s.

COMPUTING THE IFV TRANSITION MATRIX

Let $I = \{I_1, I_2, ..., I_M\}$ denote the set of valid instruction forms for the considered computing platform. Let T denote a finite set of k productions of the form:

$$l_i \rightarrow \{(Pr_{ij}, r_{ij}) : 1 \leq j \leq i_{max})\}$$

where $l_i \in I$, $r_{ij} \in I^+$, I^+ is the set of all non-empty strings of elements of I, i_{max} is the number of right-hand sides indexed by i, and Pr_{ij} is the probability of use of the sequence of instructions r_{ij} to substitute an occurrence of l_i in the program being transformed. In order to allow the engine to choose whether or not to transform an occurrence of l_i, we require that exactly one of the r_{ij} be identical to l_i.

We also require that the identity $\sum_{j=1}^{i_{max}} Pr_{ij} = 1$ holds for each production and that two different productions do not have identical left-hand sides.

Using the expression of the engine's productions, we can compute:

1. The probability that an arbitrary instruction i will be transformed by the engine into an arbitrary number x_j of some instruction j.
2. The probability that x_i instances of an arbitrary instruction i will be transformed by the engine into an arbitrary number x_j of some instruction j.
3. The probability that a code segment with an IFV v will be transformed by the engine into an arbitrary number x_j of some instruction j.
4. The probability that a code segment with an IFV v will be transformed by the engine into a code segment with an IFV v'.

The computation of the probabilities of step 2 requires individual substitutions performed by an engine on a variant to be mutually independent. Computation of each of these probabilities is hence NP-complete in general, since they each require a solution for an instance of the SUBSETSUM problem.

SUBSETSUM takes as input a set of integers and asks whether there is a subset of those whose elements sum to a target number. An approximate solution to SUBSETSUM may hence be needed here to compute these probabilities approximately.

Successful completion of step 4 yields the IFV transition matrix of the malware using the probabilistic instruction-substituting engine represented by the matrix. Each element of the matrix represents a state transition from one IFV to another.

DECISION PROCEDURE

A malware detector can use the IFV transition matrix to implement a fast decision procedure for determining whether a given program is a descendant of the Eve. Let D_0 denote the initial distribution vector of IFVs for the malware. Hence, if we wish to recognize the descendants of a malware variant Eve, then we set all of the components of D_0 to 0 except for that representing the IFV of the Eve. For any positive integer n, compute the vector $D^n = D_0T$, where T is the engine's induced IFV transition matrix. Deciding membership in nth-generation descendants of the Eve would consist of the following steps:

1. Disassemble the suspect program
2. Abstract the resulting assembly program into the sequence s of instruction forms
3. Extract the IFV of s
4. If (D^n[IFV (s)] $\neq 0$)
   then s is an nth generation descendant of Eve
   else s is not an nth generation descendant of Eve

The expression D^n[IFV (s)] represents the component of D^n that corresponds to IFV (s). Since our goal is to decide membership approximately in the various generations of descendants of a given Eve, we only need to compute and use T to determine the IFV distribution vectors D^n, up to a chosen constant value n.

OPTIMIZATIONS AND APPROXIMATIONS

It is possible for malware to have no upper bound on growth, and so in theory there may be no fixed upper bound for storing counts within the IFV. Hence, it is expected to be necessary to impose an upper bound on the size of this set while limiting, as much as possible, the deterioration of the predictive and classification power of the transition matrix. Moreover, without due care in modelling the features, the transition matrix may become too costly to construct or store. Thus, in practice, some optimizations and approximations will frequently be desirable so that the size is reduced.

Possible optimizations and approximations include, but are not limited to: (1) reducing the size of the instruction set by abstracting an assembly language instruction to its opcode mnemonic or by ignoring register names and variable values; (2) imposing an upper bound on the possible frequency of each individual instruction; (3) imposing an upper bound on the value held in any of an IFV's components; (4) abstracting the range of possible frequencies of each instruction to an imprecise scale (e.g. 'low', 'medium', and 'high'); and (5) removing from the i nstruction set those instructions which are as likely to

appear in a malicious program as they are to appear in a benign one.

One optimization that may be used in practice to reduce the size of an engine's induced transition matrix is first to pick a threshold t on individual opcode frequencies. If the frequency of an opcode in a code segment P is below the threshold, then the component of the IFV (P) that corresponds to that opcode is set to 0, else it is set to 1. Then, only a feasible count of instruction forms are considered relevant. If, say, only 10 are selected, then the IFV could be encoded using a vector of 10 binary numbers.

CONCLUSION AND FURTHER WORK

This paper brings to bear results in authorship determination of (human) documents on the problem of determining variants of a piece of metamorphic malware. After all, a metamorphic variant is a machine-authored (more accurately, machine-*transformed*) program.

The proposed approach suffers several limitations. The basic IFV transition matrix may grow too large for us to use in practice, and abstracting the matrix to optimize its size may cause a loss in the predictive power of the matrix. Furthermore, the probabilities of use may not be available. This may be remedied by estimating them from a corpus of descendants of a given malware Eve. Unfortunately, estimated probabilities may not be precise, resulting in a bias in the predictive power of the transition matrix.

We are currently investigating the possibility of extending the approach to general probabilistic metamorphism where arbitrary probabilistic transformations are employed by a metamorphic engine. It may be possible to capture the evolution of the instruction distribution of this type of malware as it transforms its own code.

REFERENCES

[1] Ször, P. The Art of Computer Virus Research and Defense. Symantec Press, Addison Wesley Professional, 1st ed., 2005.

[2] Walenstein, A.; Mathur, R.; Chouchane, M.R.; Lakhotia, A. Constructing malware normalizers using term rewriting. Journal in Computer Virology. 2008.

[3] Spafford, E.H.; Weeber, S.A. Software forensics: Tracking code to its authors. Computers & Security, vol. 12, pp. 585–595, December 1993.

[4] Meyn, S.P.; Tweedie, R.L. Markov Chains and Stochastic Stability. London: Springer-Verlag, 1993.

OPINION

BLENDED MALWARE DEFENCE

Morton Swimmer
John Jay College of Criminal Justice/CUNY, USA

Anti-malware vendors talk a lot about 'blended threats' and their solution is always 'defence in depth', which besides being a great way of selling more products is basically the right direction. For many reasons, our systems still contain vulnerabilities and are likely always to do so until the economics of system design and implementation change dramatically.

Now that operating system level vulnerabilities are better under control, more and more vulnerabilities are being found in the application level. Our best defence against the exploitation of these vulnerabilities is to use reactive technology such as anti-virus, anti-spyware, intrusion detection and prevention systems (IDS and IPS), firewalls, etc., but the delay these incur in detecting the attacks is unacceptable.

The problem should be well known by now: the time required to get the sample to the vendor, then through analysis and finally to distribute the detection updates to the clients, is still much longer than it takes potentially for the malware itself to spread. Although malware detection technologies typically use tunable heuristics, the problem of false positives makes it difficult to bring proactive detection to market, despite the number of startups trying to play in this field. It would be an advantage to have a more systematic and immediate way of creating these signatures and then be able to deploy them to where they are needed most as quickly as possible. The cure must spread faster than the disease, as we used to say when working on the *IBM Digital Immune System*.

In this article, we see how the convergence of various security technologies can help us accomplish this goal. This is achieved by utilizing the strengths of various sensors and being able to generate semantically relevant signals from them. This is a 'blended response' to a 'blended threat'.

IMMUNOLOGY

We are seeking a model for dealing with a complex and multi-level threat. Because of its powerful metaphor, the biological immune system has inspired many defence systems, not least in the field of intrusion detection and virus detection. In particular, the Self/Non-Self (SNS) detection mechanism used by the mammalian immune system is a highly compelling model.

Unfortunately, in practice, the mammalian immune system analogy – in particular SNS – has not worked particularly well when applied to computer security. The SNS model relies heavily on the ability to differentiate between self and foreign proteins and the ability to establish a memory of past infections. However, biology has many orders of magnitude more diversity and complexity than computer systems, which tend to obviate many of the problems such a system may have including the occasional false positive or false negative (which may have catastrophic effects on one individual, but not on the entire species). The false positive rate of such a system is much higher than is acceptable in the computer world.

DANGER MODEL

An alternative model, called the Danger model, has been proposed by Dr Polly Matzinger (see http://www.ncbi.nlm.nih.gov/pubmed/8011301) and it departs in one significant way from classical immunology in that it does not rely on SNS to find the foreign body. Instead, it relies on danger signals from injured cells in order for the antigen-presenting cells (APC) to activate the T-cells and thereby the appropriate B-cells that eliminate the antigen. This model is not accepted in the medical community yet, and it may never be, but we don't necessarily need the model to be validated in the medical field to find it useful in ours.

For us, the real lesson of the Danger model is that co-stimulation through a signal that identifies the threat as dangerous is required to confirm an attack. We want to combine a well-defined danger signal with some other well-defined signal, such as an SNS signal, and possibly others, before issuing an alert. The resulting composite alert will then be used to stimulate other components of the defence network.

We can also contemplate diluting this model slightly, simply by requiring two or more independent signals – so long as both signals indicate an attack – rather than strictly requiring evidence of clear and present danger in one of the signals.

Because the Danger model gives us a higher confidence level in our observation of the attack, we can now derive signatures automatically from the running system. Analogously to the cloning of the appropriate T-cells, the signatures are then spread from the originator to neighbouring systems, thus spreading the detection out from the origin.

The sensors that use these signatures can dismiss unused ones over time. This can happen, for example, because the type of sensor in question never sees that sort of traffic. Keeping old signatures around too long may degrade sensor performance or cause false positives. Collectively, however,

the entire network must be able to maintain a complete set. This is close to the biological model, where fewer and fewer T-cells are available to detect a long gone threat, but always remain in minute quantities and quickly replicate if stimulated by a recurrence of that threat. This is all very nice in theory, but how could an implementation look?

A SKETCH OF AN IMPLEMENTATION

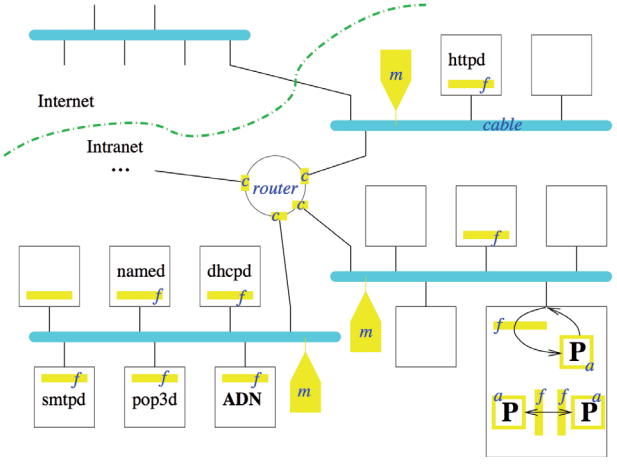
It would be foolish to throw out existing technologies and all the intelligence that went into them, but some things need to change to build an approximation of this architecture.

First of all, despite the trend towards security suite products, internally these are comprised of separate components, each of which is expert at doing a certain thing well in a certain context. There are also many products that should factor into a complete malware security solution but which are perhaps not yet mainstream, such as the *Firefox* extension NoScript.

On my Mac, I use a personal firewall product for monitoring incoming and outgoing connections, an anti-virus product to determine if a file is infected, and a script monitor for my *Firefox* browser. I feel this is less than ideal, but it has worked for me so far. Of course, these products don't talk to each other to build the bigger picture of a potential attack or even instigate countermeasures automatically.

However, using an anti-virus solution designed for detecting *Windows* malware on a Mac doesn't make that much sense and may result in false positives (to its credit, the one I'm using hasn't produced a false positive so far). The problem is that taking any tool out of the context it was designed for is just not a good idea. Trying to detect DOS/*Windows* malware on a Mac may incur false positives merely because there are fewer Mac files in a vendor's false positive set or because the instruction set of the G4 processor produces unexpected code characteristics. An IDS system like SNORT may detect suspected attacks against a database where there is no database on the subnet.

On the other hand, a real instance of *Windows* malware on a Mac system or a database exploit on a subnet without a database, is still suspicious and needs to be reported. However, the report should indicate the futility of the attack because in our model this influences how we react to it. Furthermore, modern anti-virus products are using heuristics for malware detection, which is not the same as signature-based detection. To avoid false positives, the heuristics are tuned to be extremely conservative, but in our danger model early warning heuristics can be useful, so



long as the type of detection is made explicit. The context is important.

Behaviour-based (BB) security tools, in particular in the anti-malware field, have been on the rise recently because they promise to remedy the problem of detection lag time long associated with knowledge-based (KB) tools. However, they are very different in nature and in our model we treat them as complementary to KB tools.

BB monitors are capable, at least theoretically, of producing signatures that KB scanners can consume in our danger theory model. But before they can be allowed to do that, we need to address the problem of false positives. The unfortunate fact is that BB monitors are intrinsically prone to false positives, so the goal is to reduce the number of false positives to nearly negligible levels.

FORMALISM

In the past, attempts have been made to use event correlation on the data from intrusion detection sensors to produce a signal of higher quality through aggregation. Despite incremental improvements, no one would trust the output of correlation to be false positive free. The trouble is that inputs to these systems are not independent of each other and correlating the events cannot produce a better signal if the input signals are effectively reporting the same thing. This is where the missing context comes back to bite us.

Furthermore, the information in the signal is usually imprecise in that the event is reported using a vendor-specific code or text, and while correlation can adapt to the format of the input signal, it is much harder to attach a precise meaning to some arbitrary text or code.

What is needed is more formalism in event reporting. The output of sensors, be they IDS, AV, honeypots, etc., needs to be expressed in a way that is formally comparable to other

signals. Note that there is no need for all vendors to agree on a single language, but in whatever form they decide to express their signals, the output must not only be parsable, but also comparable.

Over the last couple of years a model for expressing ideas in a comparable form has matured in the form of OWL-DL (a subset of OWL-Full), which is an ontology language for description logics. Both the event itself and its context can be captured this way, elevating what could have been a piece of data with semi-well-known characteristics to a true piece of information that can be used in a reasoning system. Once more than one signal has been found from truly and provably independent sources, the correlator can determine if there is sufficient merit to raise an alert. With greater confidence in the quality of the resulting signal, automatic response in the form of signature capture and dissemination can be achieved.

Lastly, no single vendor is capable of creating a turn-key system based on these guidelines. It is not feasible to cover all available platforms, neither is it really necessary for one vendor to do so. The key is to make the architecture open, but secure. The modes of communication must be documented and freely available for any willing vendor to participate, though the system must still be kept secure from subversion. That is certainly a tough problem but not an insurmountable one as there are various successful models one can emulate.

CONCLUSIONS

Now that the anti-malware industry has matured, the feeling is that it has lost sight of its mission of protecting the community. I sorely miss the big picture when looking at the offerings of the various vendors. Certainly, the start-ups with their (sometimes) new ideas can only focus on their individual solutions. The smaller vendors provide us often with very focused products, which is good, but only if they interoperate. The large vendors are the ones who talk loudest of 'blended threats' and 'defence in depth', but cannot (or in one case will not) cover enough of the IT infrastructure to deliver on their own.

It would be of great benefit to the IT community at large if a more complete solution could emerge soon, as my feeling for the last year or so is that we are finally losing the war against malware.

Morton Swimmer will present an extended version of this paper at VB2008 in Ottawa this October. VB2008 takes place 1–3 October 2008 in Ottawa, Canada. The full programme, with abstracts for each paper, as well as online registration, can be found at http://www.virusbtn.com/conference/vb2008/programme/.

PRODUCT REVIEW

EEYE DIGITAL SECURITY BLINK PROFESSIONAL 4.0

John Hawes

Founded ten years ago and based in Orange County, California, *eEye Digital Security* first made its name as a vulnerability research company, providing security advisories on flaws found by its teams investigating a wide selection of software and offering businesses a range of security auditing services. From this grew the company's current range of security offerings, which include several packages focused on protecting network-facing servers from the vulnerabilities presented by flaws in software and configuration, managing policy enforcement and incident reporting across corporate networks, as well as monitoring network traffic for potentially dangerous activity.

The company's vulnerability alerting service continues to offer privileged detail and early warnings on upcoming dangers, as well as a forum for administrators to debate the latest flaws and the hottest techniques for locking down systems and networks. The company boasts more than half of the US Fortune 100 companies amongst its clients, and its early research successes include spotting and alerting on the IIS flaw, which soon after allowed the Code Red worm to spread across the world's web servers.

The *Blink* desktop offering first appeared about four years ago, and has grown from a simple HIPS product into a full endpoint suite, combining the standard ingredients of anti-malware and firewall with proactive defence in the form of intrusion prevention and vulnerability management. The suite is available in a full-featured 'personal edition' for home users, and the professional edition, which offers greater flexibility of configuration and can be combined with a centralized management and reporting system.

Version 3.0 of the product, using anti-malware technology provided by the *Norman* engine, received its first VB100 award in June last year in some style. The latest version (4.0) is due for release shortly, featuring the redesigned interface introduced in version 3.5, additional *Windows Vista* support and a number of improvements under the hood.

WEB PRESENCE, INFORMATION AND SUPPORT

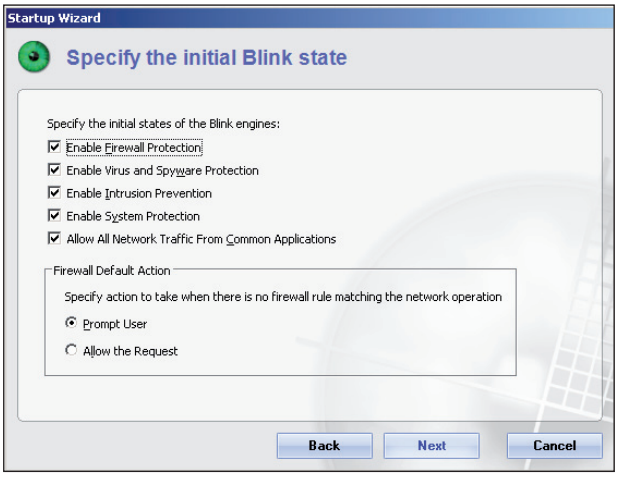
eEye's main web presence is at www.eeye.com, a site dominated by product marketing with in-depth coverage of the firm's various offerings. All products are available as time-limited trial editions, with the personal edition of *Blink* currently free for home-user purposes while offering

the same level of protection as the professional suite, and all are backed up by a wealth of information about them and the security problems they address. The site also carries the usual items of company and product news, as well as links to a number of favourable reviews and test performances.

On the more technical side of things, a research sub-site is the home of the company's vulnerability information, most of which seems to be available only to subscribers to the company's 'Preview' services. This offering is available at several levels of detail, the higher of which include personalized network security scanning, advice and insider information on the latest undisclosed vulnerabilities, as well as the standard alerting, in-depth analysis and newsletters on significant software security issues. The area also includes a selection of security research tools available for download.

Technical support for the products is similarly available at a range of subscription levels, with the most basic providing access to email-based support via an online form. A knowledgebase of common issues is available to all, however, and provides brief and often highly technical details on a range of common issues, focusing on the server range of products and the management suite. In fact, all the searches I carried out specifying *Blink* as a filter returned information on issues associated with deploying *Blink* across the network (generally solvable by setting *Windows* networking controls correctly). Behind the customer login area resides access to further documentation and guidance, including the user manuals which are also accessible directly from within the product, more on which later.

Having spent long enough looking at the information available online, it was time to get my hands on the product and see whether it would stand up to the impressive boasts made about it in the wealth of marketing material.



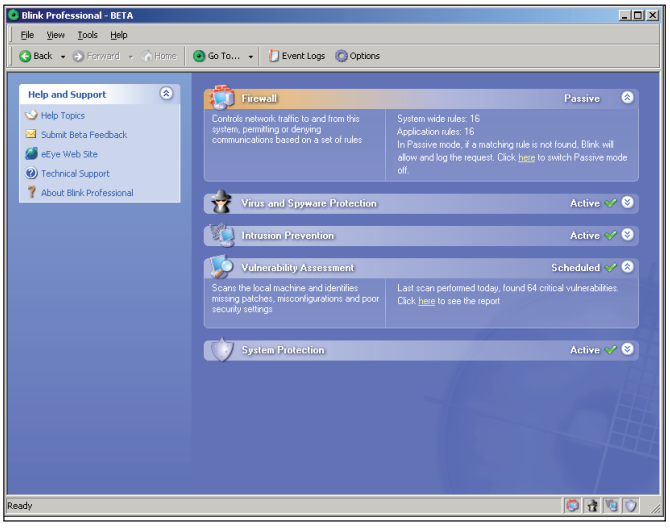
INSTALLATION AND CONFIGURATION

Initial installation of the product is a pretty standard process. The installer for the latest beta build of version 4.0 of the product comes in at a very reasonable 45 MB and runs through its business pretty rapidly, with the usual installation location options and EULA to be got through, as well as an unusually long activation key. On one system, the installer complained about a freeware browser sandboxing utility I had installed, insisting it be removed before the installation could continue, but there were no other hitches.

At the end of the process a dialog provides some information on the product's default settings and status – this begins with the firewall in rather minimal protective status, set to allow anything that is not specifically blocked by a rule. This gives something of a clue as to how the product operates – this is no simple set-and-forget tool for the average unskilled user, and although the default set of functions do provide a basic level of protection against the majority of attacks, the beauty here is in the depth of control available. A huge range of optional extras are available to achieve maximum lockdown, while the product's initial state is to apply only those thought suitable for all situations. Tuning the product to meet the individual requirements of the user requires considerable understanding of the problems being faced and the means provided by the product to mitigate them.

The interface provided to access this vast configuration is simple and reasonably appealing, being modelled along similar lines to built-in *Windows* tools such as the 'Security Center' or other system configuration applications, with menus of options on the left and details in the main panel. This gives it a straightforward and no-nonsense feel, achieving a sense of simplicity and authority without the unfriendly starkness which often comes along with more business-oriented products. This again reflects the product's ethos, not bending to the whims of the inexperienced user with lots of twinkly cartoon graphics.

Navigating the system is pretty untaxing. There are five main categories, of which at least three are pretty obvious – the firewall, anti-malware and vulnerability scanning components. The other two, labelled 'Intrusion Prevention' and 'System Protection', seem to overlap somewhat and it is not immediately obvious what each covers, but looking inside soon clears things up. The system protection area covers guarding of registry and applications, while everything else, including anti-phishing measures, is included under intrusion prevention. With most of these now fairly standard in security suites, I opted to start off with the most novel, the vulnerability scanner.



SYSTEM HARDENING FUNCTIONS

With the product installed, there are several steps required before the host system is fully secured to *Blink*'s satisfaction. The initial interface shows several items to be lacking the comforting green tick that signifies that they are fully active. The most interesting and unusual of these is the vulnerability scanner. This requires an initial run to find any problems with the current setup of the system, and the setting up of a schedule to look out for any further flaws.

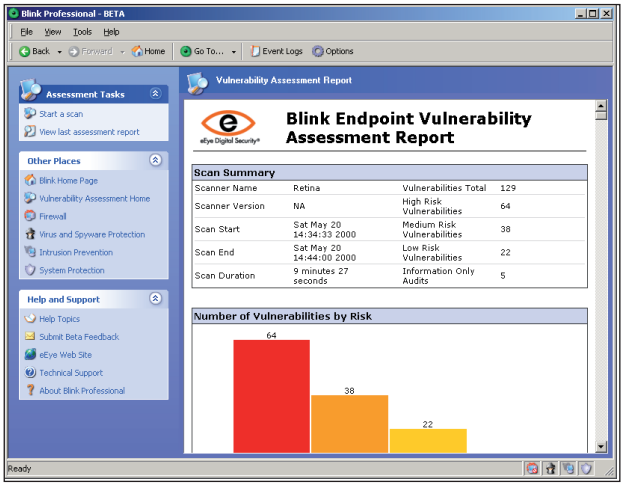
Running the vulnerability scan is a pretty simple process. The module has few options, simply the ability to schedule scans or run them manually, and a report viewer to analyse the results. The scan itself was pretty fast, taking no more than a minute or two even on crowded and low-powered systems. In test systems in the sealed *VB* lab, a large number of problems were easily identified thanks to the lack of access to recent updates from *Microsoft*. To emulate a real user more closely, I fired up a well-used and by now rather wheezy old laptop, which had languished powered down under a bed for several months. With the product installed and updated, the vulnerability scanner found an even wider range of issues – the majority of which were easily resolved by letting the *Microsoft* updater carry out its slow and tedious business of downloading and installing missing patches. However, for the remaining issues it seemed that considerably more work would be required to satisfy *Blink*'s stringent requirements.

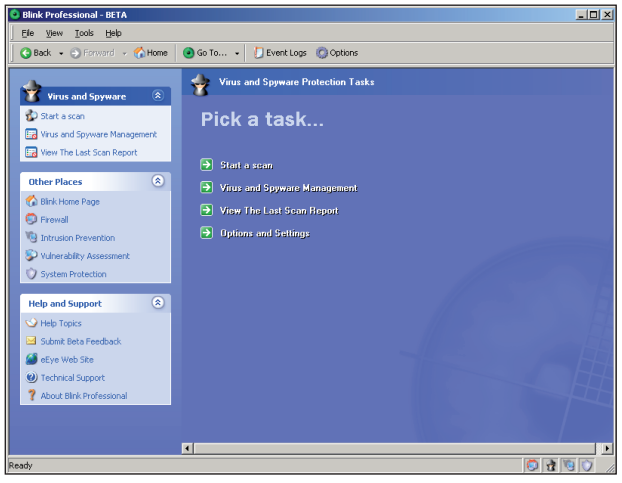
Several of the remaining issues concerned various pieces of software installed on the system, ranging from several *Adobe* and *Mozilla* products to more surprising ones such as *WinRar*. While some had their own updaters, several required manual update or even reinstallation. Among the most serious problems found was a 'zero-day' vulnerability in some *Microsoft* software which, as the report pointed out, was as yet unpatched; instead a workaround was suggested, with a link helpfully provided to advice from US-CERT on applying it. One item remaining on the 'high risk' list was a problem with anonymous registry access, a slack setting which could be closed down with a few tweaks in the registry.

Browsing further down the lengthy report, a slew of entries detailed potential weaknesses in my system. These included a lack of fully trackable logging, unsafe caching of usernames, passwords and page file contents, as well as various issues with unnecessary services, drive sharing and allowing unaccredited users to perform various activities. The autorun default, a spreading vector of a lot of recent unpleasant worms, was also highlighted, and even the fact that users could insert USB key drives and use them to move data off the machine was mentioned as a potential means for unwanted data extraction.

Each entry was accompanied by details of how to correct or mitigate the problem, usually in the form of instructions for doctoring registry keys, changing settings using Control Panel tools, or links to more involved instructions in appropriate places, predominantly *Microsoft Knowledge Base* articles. Each entry was also accompanied by links to alerts and advisories on the subject, from the likes of *Secunia* and *iDefense* as well as *eEye*'s own vulnerability pages, *Microsoft* bulletins and articles and other alerts from the software developers involved in any given flaw, with CVE numbers included where appropriate.

The depth of detail provided was remarkable, and the range of areas covered, from potential remote exploits and sources of data extraction to problems with fully accountable logging and physical access points for abusive users, was quite staggering. The sheer scale of the issue





of locking down a system could easily be overwhelming, particularly for the less technically minded user, but for a network admin wanting to ensure all the systems in his charge are as secure as possible, and with the power to automate most of the tasks involved, this is surely an invaluable tool.

Vulnerabilities in software are a huge vector for malware, particularly in the ever-growing area of web threats which are rapidly increasing in complexity, subtlety and scale, with more and more legitimate sites playing unwitting host to attacks. Most of these attacks make use of long-patched flaws, probing systems for holes to sneak malware onto new victims, and the importance of keeping a system fully patched is greater than ever. Since this task is also more complex than ever, having details of all the potential dangers in a single report, along with information on remediation, and having it regenerated rapidly on a regular basis to keep up with the latest developments, is an enormous advantage.

The only feature I could think of that would be a useful addition would be an option to disregard some of the entries, as either unfixable in a given situation or not applicable under a corporate policy, but given the attention to detail it seems more than likely that such functionality is already available to admins using the separate management tools. As it was, it was tempting to try to eliminate each and every one of the issues flagged up, if only to see what would happen when a scan found nothing to complain about – surely some kind of fanfare or shiny virtual gold medal would be an appropriate reward for such diligence.

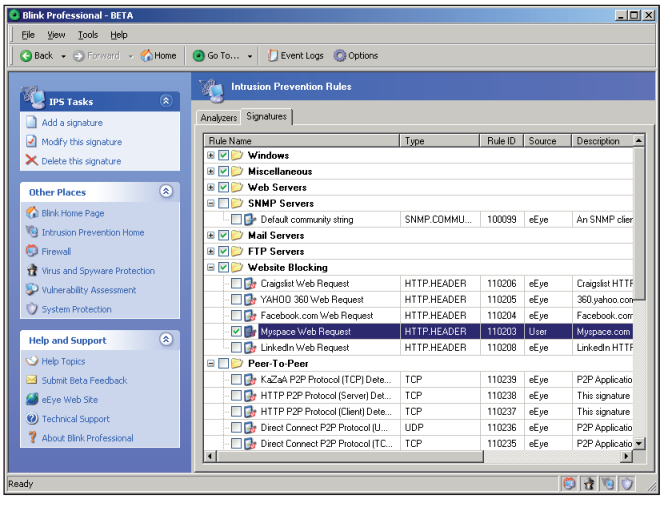
Sadly time was too pressing to go to such great lengths, and I left my test machines with a few minor issues remaining unfixed to look into the more common security measures provided by the suite.

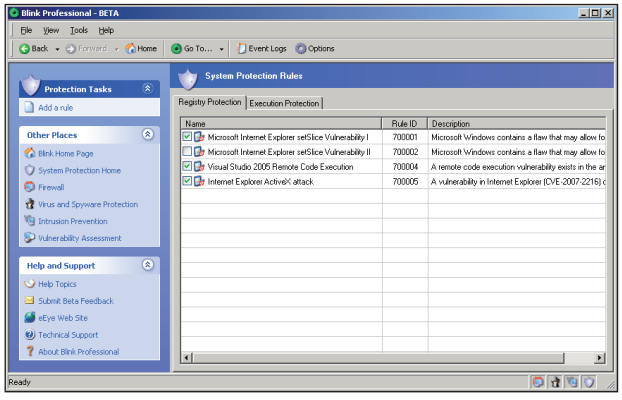
SYSTEM PROTECTION FUNCTIONS

Of course, once the system is fully patched and configured to the product's liking, the vulnerability scanner becomes a core part of the ongoing protection offered. A scheduled scan will highlight new patches as and when needed, including updating the status of those nasty as-yet-unpatched flaws. New configuration tips are also added as researchers spot new vectors and new potential issues with the standard setup of a *Windows* system. Beyond this rather special functionality, however, the product also offers a full set of the more usual protection features provided by most other security suites on the market.

At the core of the standard anti-malware protection provided is the *Norman* engine with its strong 'sandbox' heuristics. Running it over the *VB* test sets showed a high level of detection, which was improved still further after upping the heuristic settings. The interface to the engine and all the file-hooking and other integration is developed by *eEye*, and operating the scanner and adjusting the on-access settings proved a pleasingly simple business, with defaults seeming well chosen and appropriate. Any on-demand scans required were also available from the context menu. On its own this seemed something of an improvement on *Norman*'s own interface to the same detection technology, which I have frequently found rather complex and fiddly when adapting it to the specific needs of VB100 testing.

Scanning speeds and on-access overheads closely mirrored past test results for *Norman* and *Blink*, implying that little extra burden was being placed on the systems by the range of added extras. The *Norman* engine has a long and illustrious past in VB100 comparative testing, and with a few recent problems caused by a batch of polymorphic items now behind it, it looks set to continue to do well. It also regularly achieves decent scores in other independent





tests, making the 'Advanced' grade in the most recent *AV-Comparatives* test and scoring 'Satisfactory' or better in all but the speed category in *AV-Test*'s latest set of results. In our own speed measurements, both *Norman* and *Blink* products appear in the middle of the field, somewhat behind some of the zippiest products but never imposing the sort of overheads seen in the weightier ones. Using the product on a range of systems I never observed any intrusive slowdown, although when running the updater on a particularly aged and underpowered machine whilst trying to carry out several other tasks, things did become a little slow to respond for a few minutes as drive lights flickered and crackled with effort.

Moving on to the intrusion prevention filters, these again seem to focus to a large extent on vulnerability monitoring, watching numerous protocols for suspicious data which could indicate an attempted attack. The large set of categories comes fully stocked with long lists of known bad behaviours, and a separate tab presents a lengthy list of signatures for known exploits. The majority are active by default, but some are provided for those who have more specific needs, which include a website-blocking section populated with common social networking sites.

The process of adding more rules and signatures is via a simple and straightforward wizard, which in all these modules advises the user to be sure they know what they are doing before setting up a rule which could impinge on important system operations. With the default settings already pretty thorough, exploit signatures can be extended by adding pattern strings of one's own design, providing the user with a level of control over what comes through to the machine usually only available to network admins. The

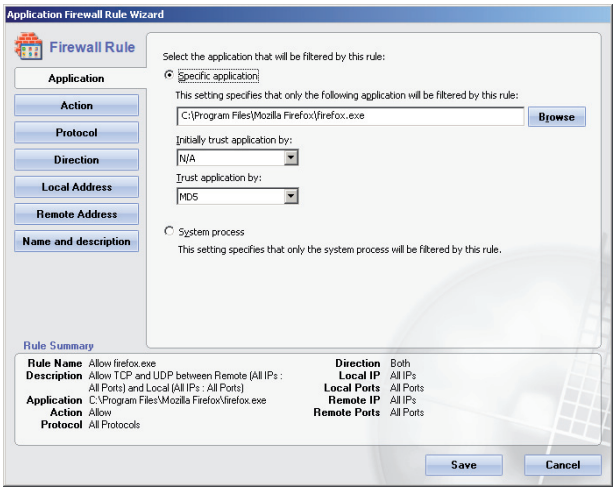
phishing controls, listed under 'Identity Theft Rules', cover a range of common tricks found on phishing web pages, including hidden or spoofed URLs and links, and again can be extended to the user's content.

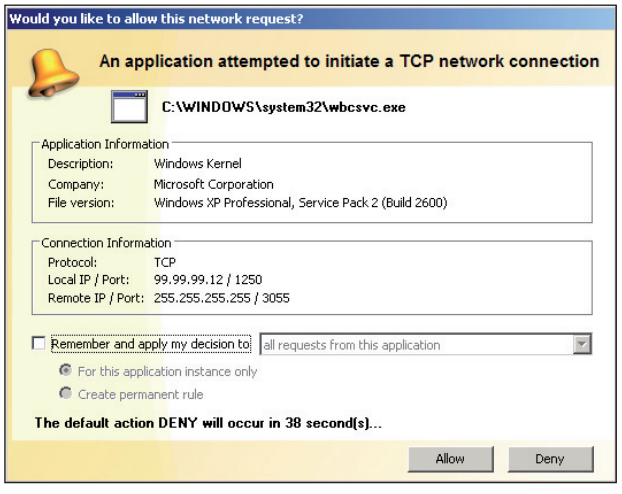
The system protection setup operates in a similar manner, this time with far fewer built-in rules but with the same straightforward system to allow the user to generate their own. Setting controls on specific applications, ensuring doctored versions cannot be run, or even allowing them only to be run by a specific parent process, is a pretty straightforward task achieved in a few clicks, and a similar system prevents (or allows) access to specific areas of the registry.

The firewall also uses the same system, giving a pleasing consistency across the product. The various options, with a handful of default system-wide rules and more for specific applications, are presented clearly and legibly with a good level of plain-language description to assist the less technical user. Its initial rather passive setup does require a few extra steps to ensure a decent level of protection, but this can be done with a couple of clicks of check-boxes, and it seemed to operate well once fully up and running.

Most of these rules function in a quiet and unflashy way, not bombarding the user with a deluge of hyperbolic warnings about blocked activities and simply logging unwanted events, if desired. Even the on-access malware scanner produced small, simple popups with the minimum of fuss.

The settings can be programmed to provide a training popup, filled with detail and options, when an unknown application attempts a restricted activity. In my tests, these managed to block the handful of malicious items that managed to get past the signatures and heuristics of the anti-malware engine, as they attempted to leak data from the system, contact base to download further nasties,





doctor important registry entries or perform other malicious activities. The popups default to a deny action if left for 45 seconds.

My only quibble with the whole setup is that the descriptions of the rules are often considerably longer than the display space available. Double-clicking the title bar boundaries shrinks the area even further rather than expanding it to the required width, which means that it takes some fiddly stretching of boxes and dragging of sliders to read the full detail of any given rule or setting. That this detail is available at all is impressive, however.

HELP AND GUIDANCE

The provision of clear and useful information, a pattern repeated across the product, caters more than adequately for the complexity of configuration available. While this is not a simple set-and-forget system, and may appear daunting to many inexperienced users at the desktop level, the product provides plenty of information for those willing to put a little effort into deciding for themselves how to set things up.

Beyond the basic information provided alongside each individual rule, vulnerability alert or malware warning, a superbly detailed manual is provided, alongside an equally well thought out help system. Unlike many help pages, which often do little more than list the available buttons and what they do, this is properly task-oriented, detailing the steps required to achieve a given objective. The manual PDF runs to some 99 pages, providing even more step-by-step information on how the various features should be operated, including detailed instructions for defining new rules. All are written in lucid language with a minimum of jargon, and are clearly aimed at putting the exceptional power of the product within the reach of the humbler user.

CONCLUSIONS

With such an in-depth product to look at in a very short time, it has not been possible to do more than skim the surface of *Blink*'s capabilities. I have focused predominantly on the vulnerability scanner as it is a rare if not unique component in a security suite, but the rest of the functions (apart from the straightforward anti-malware scanner) are also unusual in the sheer depth of configuration available. In the right hands, this product can do far more than provide solid security from malicious code and attacks; it can implement a complete usage policy, managing many aspects of how a system and its user operate, including controlling access to unwanted software and web resources, maintaining hygiene standards and accountability through logging.

Of course, those hands need to know what they are doing, but as I have come to see through longer exposure to the product and its support systems, they do not necessarily need to be those of an expert. Enough background information and links to further resources are provided at almost every level of the product to allow an informed and committed novice not only to implement a solid security regime on their system, but also to learn a considerable amount about it along the way. The home-user version, offering the same full range of tools and options, can be put to use fairly simply using more or less the default settings to provide a very decent level of security, but with a little effort, and some trust in the assistance provided, can allow anyone to take control of their computer and take a little responsibility for their own online safety.

Of course, I can understand how this could be rather too much to bear for many home users, and they may be better off investing in something more cuddly, but for those willing to put in the effort the rewards should be well worth it. In a more professional setting, for those requiring absolute control to enforce a detailed and demanding security policy, *Blink* can provide a superb breadth of power to do just that, in a single well-designed and solid package.

Technical details

eEye Digital Security Blink Professional 4.0 was variously tested on:

AMD K7, 500 MHz, 512 MB RAM, running *Microsoft Windows XP Professional SP2* and *Windows 2000 Professional SP4.*

Intel Pentium 4 1.6 GHz, 512 MB RAM, running *Microsoft Windows XP Professional SP2* and *Windows 2000 Professional SP4.*

AMD Athlon64 3800+ dual core, 1 GB RAM, running *Microsoft Windows XP Professional SP2* and *Windows Vista SP1* (32-bit).

AMD Duron 1 GHz laptop, 256 MB RAM, running *Microsoft Windows XP Professional SP2.*

END NOTES & NEWS

The 2nd International CARO Workshop will be held 1–2 May 2008 in Hoofddorp, the Netherlands. The focus of this year's workshop will be on the technical aspects and problems caused by packers, decryptors and obfuscators in the broadest sense. For details see http://www.datasecurity-event.com/.

EICAR 2008 will be held 3–6 May 2008 in Laval, France. See http://www.eicar.org/conference/ for the full details.

The 5th Information Security Expo takes place 14–16 May 2008 in Tokyo, Japan. For more details see http://www.ist-expo.jp/en/.

The 9th National Information Security Conference (NISC) will be held 21–23 May 2008 in St Andrews, Scotland. For full details and registration information see http://www.nisc.org.uk/.

Hacker Halted USA 2008 takes place 1–4 June 2008 in Myrtle Beach, SC, USA. The conference aims to raise international awareness towards increased education and ethics in information security. Hacker Halted USA delegates qualify for free admission to the Techno Security Conference which runs concurrently. For more details see http://www.hackerhalted.com/.

The 20th annual FIRST conference will be held 22–27 June 2008 in Vancouver, Canada. The five-day event comprises two days of tutorials and three days of technical sessions where a range of topics of relevance to teams in the global response community will be discussed. For more details see http://www.first.org/conference/.

The SecureAmsterdam conference on emerging threats takes place 15 July 2008 in Amsterdam, the Netherlands. For details see https://www.isc2.org/cgi-bin/events/information.cgi?event=66.

The 17th USENIX Security Symposium will take place 28 July to 1 August 2008 in San Jose, CA, USA. A two-day training programme will be followed by a 2.5-day technical programme, which will include refereed papers, invited talks, posters, work-in-progress reports, panel discussions, and birds-of-a-feather sessions. For details see http://www.usenix.org/events/sec08/cfp/.

Black Hat USA 2008 takes place 2–7 August 2008 in Las Vegas, NV, USA. Featuring 40 hands-on training courses and 80 Briefings presentations. This year's Briefings tracks include many updated topics alongside the old favourites including zero-day attacks/defences, bots, application security, deep knowledge and turbo talks. Online registration is now open. For details see http://www.blackhat.com/.

VB2008 will take place 1–3 October 2008 in Ottawa, Canada. Presentations will cover subjects including: sample sharing, anti-malware testing, automated analysis, rootkits, spam and botnet tracking techniques, corporate policy, business risk and more. Review the programme and register online at http://www.virusbtn.com/conference/vb2008.

Black Hat Japan 2008 takes place 7–10 October 2008 in Tokyo, Japan. For full details see http://www.blackhat.com/.

The third APWG eCrime Researchers Summit will be held 15–16 October 2008 in Atlanta, GA, USA. eCrime '08 will bring together academic researchers, security practitioners, and law enforcement to discuss all aspects of electronic crime and ways to combat it. For more information see http://www.antiphishing.org/ecrimeresearch/.

The SecureLondon Workshop on Computer Forensics will be held 21 October 2008 in London, UK. For further information see https://www.isc2.org/cgi-bin/events/information.cgi?event=58.

RSA Europe 2008 will take place 27–29 October 2008 in London, UK. For full details see http://www.rsaconference.com/2008/Europe/.

CSI 2008 takes place 15–21 November 2008 in National Harbor, MD, USA. A call for papers is now open. Online registration will be available from June. See http://www.csiannual.com/.

AVAR 2008 will be held 10–12 December 2008 in New Delhi, India. A call for papers has been issued, with a submission deadline of 15 July. For more details see http://www.aavar.org/avar2008/.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:
Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England
Tel: +44 (0)1235 555139 Fax: +44 (0)1235 531889
Email: editorial@virusbtn.com Web: http://www.virusbtn.com/

vb Spam supplement

CONTENTS

NEWS & EVENTS

NEW HOME FOR THE SPAMMERS' COMPENDIUM

For more than five years, John Graham-Cumming has tracked the tricks used by spammers in the bodies of their messages and recorded the details of those tricks in a collection known as The Spammers' Compendium. As the Compendium has grown it has proved a useful resource for spam-fighters, enabling patterns in trickery to be identified and innovations to be spotted. At the end of March, however, John announced his retirement from the anti-spam industry and *VB* is very pleased to reveal that John has handed over the hosting and maintenance of the Spammers' Compendium to *Virus Bulletin*.

The new home of the Spammers' Compendium is at http://www.virusbtn.com/tsc. As previously, entries are made in The Spammers' Compendium when new tricks have been identified in spam seen in the wild by volunteer contributors. Submitters of new tricks will be credited in The Spammers' Compendium for their contributions. Please send contributions to tsc@virusbtn.com.

EVENTS

The 13th general meeting of the Messaging Anti-Abuse Working Group (MAAWG) will be held in Heidelberg, Germany, 10–12 June 2008. The meeting is open to MAAWG members only. The 14th general meeting (also members only) will take place 22–24 September 2008 in Harbour Beach, FL, USA. See http://www.maawg.org/.

CEAS 2008 will take place 21–22 August 2008 in Mountain View, CA, USA. CEAS is soliciting non-spam email for use in its 2008 spam challenge. Non-sensitive legitimate email can be donated at http://ceas.klika.eu/ceas/. For more information about the event see http://www.ceas.cc/2008/.

FEATURE

DELIVERING RELIABLE PROTECTION AGAINST PHISHING WEBSITES

Sorin Mustaca
Avira, Germany

Phishing, spam and malware have become major problems for users of the Internet and for online businesses. Whether delivered as email attachments or via URLs contained in emails, the AV industry is doing its best to protect customers against these threats by gathering and analysing emails with dangerous attachments and by blocking malicious URLs.

Any user who buys a complete security product can expect to receive both local and online protection. Online protection is provided by those products or modules that deal with information coming from outside the system or network on which they are operating. Usually these are the firewall, anti-spam, anti-phishing, URL-filtering and parental control modules. Great importance is currently placed on URL filters, which must be able to prevent the user from accessing phishing and malware-serving sites.

It might seem a trivial task to identify malicious URLs, pack them in a file and send them via updates to the customer so that the URL filter can block them, but the reality is a little more complex.

It all starts with spam traps, in which hundreds of thousands of spam, phishing and malware emails are

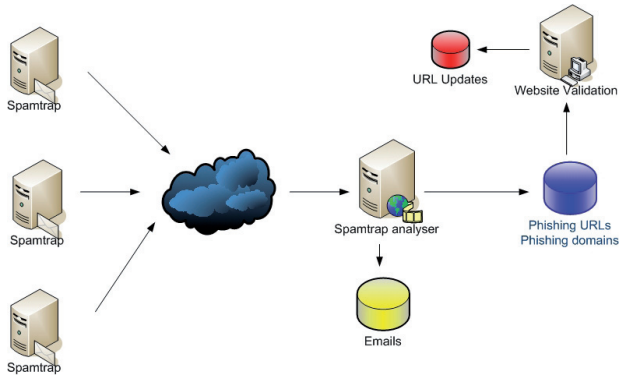


Figure 1: System's architecture.

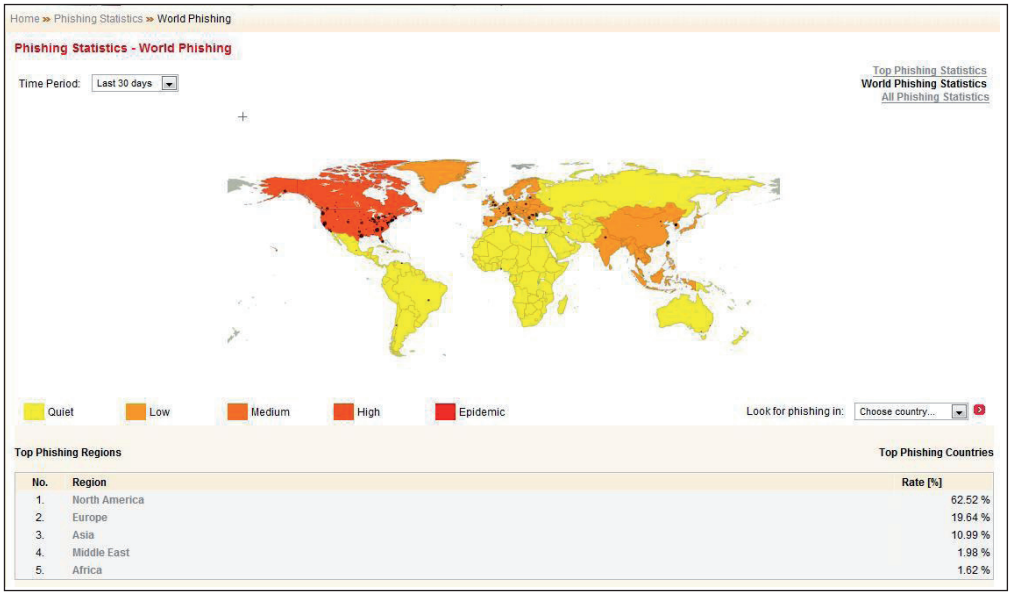


Figure 2: World phishing statistics [2].

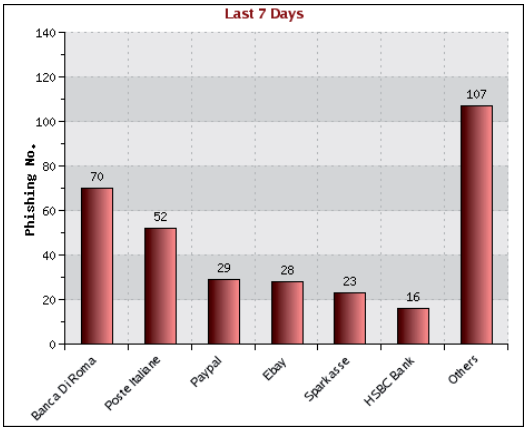


Figure 3a: Targeted brands for 7 days [3].

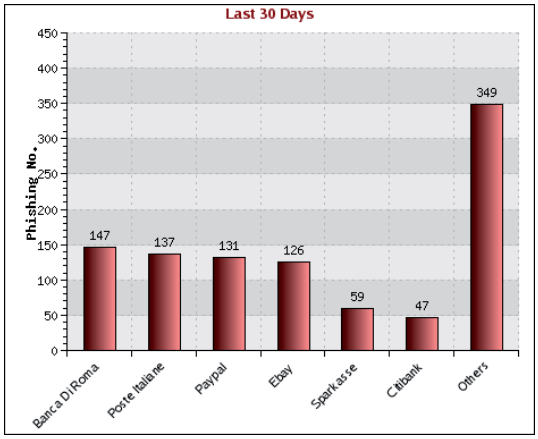


Figure 3b: Targeted brands for 30 days [4].

gathered each day. An automated system gathers the emails and splits them into spamming and phishing categories (Figure 1). The emails categorised as phishing are sent to a URL analysis system. This system must check for false positives (i.e. check that the URLs contained in these emails really are malicious), check that the website to which each link points is live and online, and that the website is a phishing site rather than an automatic redirect (this is not as easy as it might seem). The system must also inform several online entities about each malicious URL and prepare a product update for the customers.

This paper will explain how such an automated system was created and how its results are used.

THE ENTRY POINT

As mentioned, emails are collected in spam traps – mail boxes that have been set up for the sole purpose of collecting spam and which no-one uses for genuine incoming or outgoing email. By using these spam traps we can be certain that the email collected is spam – there is no real person behind the inbox to say 'I did opt to receive an email from company X, but not from company Y'. Removing the human factor gives us the most secure way of being able to say that a message is unsolicited.

We gather emails from spam traps hosted by mail servers all over the world, giving us an almost global overview of the spam activity in any 24 hours around the planet (see Figure 2). Interestingly, even though we receive emails from many areas around the world, and we sometimes see outbreaks in German, Italian, Spanish, Romanian and others, the vast majority of phishing emails are in English.

With a finite number of domains seen in the phishing emails, we can also produce statistics about which brands have been targeted and for how long (see Figures 3a and 3b).

Our anti-spam product can differentiate between the targeted phishing domains and extract the URLs from the

emails, so we store this information for further analysis if the content of the website to which the URL points matches the content extracted from the email. This is just another measure to check for false positives and website availability.

FILTERING URLS

'To be' or 'not to be'?

When we first started to develop this anti-phishing system, we created a simple Perl script that launched an external program to test the URLs. If the return value of the program was 0, the website was live; if it was 1, the website was not valid.

However, we soon realized that even though many of the websites were no longer online, the ISPs hosting them were not always returning a simple '404 - Page not found' error, but instead a page containing some form of explanation such as:

- 'website not found any more, contact the webmaster' (the page was simply deleted)
- 'website is available for renting'
- 'website is no longer valid because it contained a dangerous page' (contravening the EULA results in automatic deletion)

Alternatively, the URL would be redirected to another website (often the home page of the ISP).

Filtering these special cases would have been a lot easier had all the ISPs used English. The messages were in various languages depending on the ISP's country of origin, which meant that not all of them could be parsed.

A good idea for handling these pages would be to train a Bayesian filter with the words commonly found in such pages in order to be able to classify them automatically in the future. The filter could be trained with the HTML pages without interpreting them. This would mean that we would have to train with plain HTML and JavaScript code, teaching the classifier to 'learn' the techniques this way. This classifier would suffer from the same problems as suffered by all Bayesian filters: trained only with one type of input it will tend to detect more of that input than anything else. This project is currently being investigated.

Fortunately, after analysing some of the substituted web pages, we figured out that there are several common keywords, many of which are international. We are able to filter about 60% of these pages using the keywords.

```
C:\>wget www.google.com/dddhefiheihe
--20:49:05--  http://www.google.com/dddhefiheihe
          => `dddhefiheihe'
Resolving www.google.com... 209.85.135.103, 209.85.135.104, 209.85.135.147,
Connecting to www.google.com|209.85.135.103|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
20:49:05 ERROR 404: Not Found.
```

Figure 4: Returns '404 Page not found' code – OK.

```
C:\>wget www.110mb.com/ehdfjdhjdfh
--20:49:17--  http://www.110mb.com/ehdfjdhjdfh
          => `ehdfjdhjdfh'
Resolving www.110mb.com... 195.242.99.84
Connecting to www.110mb.com|195.242.99.84|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.110mb.com/404.php [following]
--20:49:17--  http://www.110mb.com/404.php
          => `404.php.1'
Connecting to www.110mb.com|195.242.99.84|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7.862 (7.7K) [text/html]

100%[====================================>] 7.862         --.--K/s

20:49:17 (134.87 MB/s) - `404.php.1' saved [7862/7862]
```

Figure 5: Returns either a '302 Found' and/or '200 OK' code and a page with some text – not OK.

Follow the clicker

Even though it is a rather uncommon practice to track each user who clicks on a link, we have seen phishing attacks which were probably intended to be a form of spear phishing (targeted phishing attacks). Each time we notice a URL that has a rather long and randomized parameter at the end, we cut it and we block the entire path up to that parameter. This way, we make sure that all possible combinations of the URL will be blocked.

For example, a long URL like this:

```
http://s.om.e...d.o.m.a.i.n.net/path/anfang.asp?id=0
07845698351867681038316409831031764793454231155537345
0305078216
```

is truncated to this:

```
http://s.om.e...d.o.m.a.i.n.net/path/anfang.asp
```

and the entire path is blocked to make sure that access is denied to any possible URL combination.

THE GREY AREA: PHISHING AND MALWARE WEBSITES

The system described does not have any AV scanning capabilities, so there are routines in place that filter from the outset any URL whose target is obviously a binary file, which usually proves to be a piece of malware (dropper, trojan, etc.).

Most of the phishing websites we see are 'classic' phishing sites (i.e. they imitate the site of a well-known brand and try to steal credentials), but occasionally they also attempt to download a piece of malware in the background. The websites that attempt

to do this are in a 'grey' area that crosses over between malware and phishing. I have seen only two methods used for downloading the malware: via client-side code (JavaScript) or server-interpreted code like PHP or ASP. A link to such a website looks suspicious from the start:

```
http://www.google.com/pagead/iclk?sa=l&ai=trailhead&
num=69803&adurl=http://some-phishing-website/
download.php
```

There are many possible variations where a background action starts the download:

```
http://www.google.com/pagead/iclk?sa=l&ai=trailhead&
num=69803&adurl=http://www.some-phishing-website.com
```

Since the analysis system deals only with phishing and not with malware, the only thing that can be done here is to follow the final target and if the content received is binary, discard it, thus protecting the user from a potentially dangerous download.

```
Result: HTTP/1.1 200 OK
Connection: close
Date: Sun, 03 Feb 2008 22:48:29 GMT
Accept-Ranges: bytes
ETag: "86820f-a200-47a510d4"
Server: Apache/1.3.37 (Unix) mod_ssl/2.8.28 OpenSSL/
0.9.7a PHP/4.4.7 mod_perl/1.29 FrontPage/5.0.2.2510
Content-Length: 41472
Content-Type: application/octet-stream
Last-Modified: Sun, 03 Feb 2008 00:54:44 GMT
Client-Date: Sun, 03 Feb 2008 22:43:07 GMT
```

SEARCH ENGINE REDIRECTS

The use of the *Google PageAds* as seen in the above example is another technique used by phishers. In general, using a search engine to redirect to a website must be seen as a suspicious action:

```
http://google.com/url?sa=p&pref=ig&pval=2&q=
http://www.phishing-site.com
http://rds.yahoo.com/_ylt=
http://www.phishing-site.com
http://aolsearch.aol.com/aol/
redir?clickeditemurn=
http://www.phishing-website.com
```

(Note: the above URLs are simplified. Additional parameters have been removed for the sake of simplicity.)

REFRESHING

More and more phishing websites are making use of botnets to redirect browsers from one URL to another without the user noticing.

This technique can be achieved using the HTTP Refresh or JavaScript code:

```
<script language=javascript>
top.location="http://www.phishing-website.com";
</script>
```

The same effect can be obtained with window.location.

Another technique is to use plain HTTP code to refresh the website to another location after an interval:

```
<head>
<meta http-equiv="refresh" content="0;
url=http://www.phishing-website.com" />
</head>
```

The situation becomes interesting when there is a redirect chain through the botnet. The maximum length of redirect chain we detected was four hosts.

There is a danger that these websites will form a loop, either on purpose or by mistake. In this case the parsing module of the system would enter into an endless loop and would have to be interrupted manually. To avoid this we added a maximum recursion limit of 25 redirects.

Another technique seen in the wild is to use a *rotating refresh*. This uses the same technique as the simple HTML refresh, but mixed with JavaScript code in order to self-generate the HTML document. Such a technique of making the website really dynamic could be called 'polymorphic phishing' if we borrowed the terminology from malware.

Figure 6 shows some rotating refresh code. Simple analysis of this code shows that every five seconds a new page containing a refresh URL is being generated. The page is refreshed after three seconds, which is way too often.

All the intermediary websites used to reach the final phishing website are saved in our database, regardless of

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
<html>
<head>
<title>PayPal - welcome </title>

</head>
<body>
<script language="JavaScript">
day = new Date();
hr = day.getSeconds();
if ((hr >= 0) && (hr <= 6)) {
randomURL = "http://jray.us/includes/sites/login.html";    }
if ((hr > 6) && (hr <= 12)) {randomURL = "http://vvv.vv/calendar/tools/paypal-login/login.html";     }
if ((hr > 12) && (hr <= 18)) {randomURL = "http://ccc.cc/modules/Uso_Frecuente/paypal-login/login.html";     }
if ((hr > 18) && (hr <= 24)) {randomURL = "http://xxx.xx/includes/sites/login.html";     }
if ((hr > 24) && (hr <= 30)) {randomURL = "http://xxx.xx/includes/sites/login.html";     }
if ((hr > 30) && (hr <= 36)) {randomURL = "http://yyy.yy/calendar/tools/paypal-login/login.html";     }
if ((hr > 36) && (hr <= 42)) {randomURL = "http://xxx.xx/includes/sites/login.html";     }
if ((hr > 42) && (hr <= 48)) {randomURL = "http://zzz.zz/modules/Uso_Frecuente/paypal-login/login.html";     }
if ((hr > 48) && (hr <= 54)) {randomURL = "http://xxx.xx/includes/sites/login.html";     }
if ((hr > 54) && (hr <= 60)) {randomURL = "http://aaa.aa/includes/sites/login.html";     }
document.write("<meta http-equiv='refresh' content='3;url=" + randomURL + "'>");
</script>
</body>
</html>
```

Figure 6: Rotating refresh.

the method used. This way we make sure that nothing gets changed in the redirect chain, up to the final website.

FLASHING

In June 2006 we saw an entire phishing website written in *Flash*. A 250 Kb *Flash* file called login.swf was referred to by a simple website like this:

```
<object classid="clsid:D27CDB6E-AE6D-11cf-
96B8-444553540000" codebase="http://download.
macromedia.com/pub/shockwave/cabs/flash/swflash.
cab#version=4,0,2,0" id="login" height="1280"
width="979">
  <param name="movie" value="login.swf">
  <param name="bgcolor" value="#FFFFFF">
  <param name="quality" value="high">
  <param name="allowscriptaccess" value="samedomain">
  <embed type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/
getflashplayer" name="login" src="login_files/
login.swf" bgcolor="#FFFFFF" quality="high"
swliveconnect="true" allowscriptaccess="samedomain"
height="1280" width="979">
</object>
```

The only way to detect such a website is by parsing the object and analysing the original URL. Of course, this technique is very error prone.

FRAMING

The last and by far the most commonly used technique is to use HTML frames as the entry point in the phishing website. As many as possible are used and in as complicated a way (nested) as possible. But frames can be parsed relatively easily, and this is why we seldom see techniques just using plain frames. They are used together with all the above techniques in order to make parsing as complicated as possible. Also, it seems that the phishers have taken into consideration browsers which do not support frames. Some websites we've seen have used JavaScript code to handle this kind of browser.

The solution against this technique is to act as a browser and dive into the frame structure. Of course, this makes everything a lot more complicated because it is not trivial to implement an HTML and JavaScript interpreter in Perl.

CONCLUSIONS

All of the above techniques and various combinations of them have been seen in real phishing websites. Creating a validation mechanism for these URLs is not an easy task. When a URL is found, the system has to check if the target website is a real phishing site, an automatic response because the website has been taken down or a false positive. This proves that the fraudsters are no longer script kiddies but knowledgeable developers keen to make a lot of money.

The URL analysis system described in this article is currently maintained semi-manually. The phishing URLs are gathered by a fully automated system, but the analysis of the hyperlinks cannot be fully automated. As in the case of malware analysis, human input is a vital factor, whether that is performing a manual check to see if the system's decision is correct or upgrading the automatic detection logic. Of course, in the long term, only the latter option is viable, since the unique URLs arrive in their hundreds per month.

The purpose of this system is to determine if the phishing URLs are valid, so that the invalid ones can be discarded before they reach the end users' filtering mechanisms. This way we can minimize the size of the product updates. Unfortunately, in recent weeks the number of phishing URLs has increased to such a level that it is no longer possible to check every URL at the entry point if we are to deliver the updates in a timely manner. Currently, only basic tests are performed, mostly to prevent the blocking of the ISPs that substitute the '404' error with other pages.

REFERENCES

[1] http://www.avira.com/en/threats/section/worldphishing/top/30/index.html.

[2] http://www.avira.com/en/threats/section/phishing/top/7/index.html.

[3] http://www.avira.com/en/threats/section/phishing/top/30/index.html.

[4] http://www.faqs.org/rfcs/rfc2616.html.