OCTOBER 2004

virus BULLETIN

The International Publication on Computer Virus Prevention, Recognition and Removal

CONTENTS

ISSN 0956-9979

IN THIS ISSUE

PORK TALK

Pigs are considered to be the fourth most intelligent animals (excluding humans) on the planet. Martin Overton explains how he uses a truffle-hunter of a pig, the SNORT Intrusion Detection System (IDS), to sniff out malware.
page 10

EFISHING FOR MALWARE

From pigs to fish: Peter Ferrie and Frédéric Perriot describe the zoo virus W32/Efish, whose author claimed that its encryption was unbreakable.
page 4

HASH WOES

After last month's excitement in the crypto community following the demonstration of flaws in a whole set of hash functions, Morton Swimmer and Jonathan Poritz clarify the situation and explain its significance.
page 13

vbSpam supplement

This month: anti-spam news & events; the use of trust networks for email filtering; ASRG summary.

"Current AV vendors are the dinosaurs of the security industry ... they will be extinct before the decade is out."

Nick Scales, Secure::42

DEFINITION-BASED AV SOFTWARE IS DEAD

There was a time when definition-based anti-virus (AV) software was effective, but that was a long time ago and before virus writers used the Internet to propagate their little nasties. Why, then, do AV vendors continue to promote the use of this type of product, which provides a solution *after* the virus has attacked?

Current AV vendors are the dinosaurs of the security industry – I believe they will be extinct before the decade is out. We are all waiting for the revolution; a technology that provides the user with real-time protection from all viruses, that is simple to use and transparent. This may sound like science fiction, but such types of solution have been around for some time – one example of which is policy enforcement.

A very simple anti-virus technique, policy enforcement allows a business to set up policy rules for accepting or rejecting code. For example, many organisations apply the simple, but incredibly effective rule, "*we will not accept any active code unless it is from an expected source and the code is signed as coming from a known source*". Many large enterprise users have recognised that this rule is highly effective in stopping the mass infection of hosts. Through the application of a sequence of rules such as this it is possible to provide a policy-based anti-virus solution that not only is as effective as definition-based AV, if not more so, but which is also cheaper and easier to manage.

So, why don't the AV vendors set aside some of their development budgets to produce security software that allows for the easy implementation of these policies? The answer is simple. Policy enforcement software does not require the user to subscribe to a continual update mechanism, the user does not need constant contact with the vendor and the viruses do not need to be named and hyped – meaning the user no longer feels constantly under threat. All of these add up to a much lower revenue stream and profile for the AV vendors.

It is not a coincidence that this non-detection form of AV protection is starting to appear from those who have no vested interest in selling the world $3 billion worth of updates each year. Even *Microsoft*'s *XP* service pack 2 has made great strides in this direction.

This type of AV has some enormous advantages. It does not depend on some of your customers becoming infected first, there is no delay in providing protection, the customers' costs are much lower, the cost of delivery is hugely reduced, a great deal of this technology can be placed in the network, and it is very simple and quick to implement.

If this is all true, why haven't we seen it yet? Because the AV vendors do not want it to appear. They have ignored these technologies because they can see the impact in their revenue streams. Imagine what would happen to the revenues and stock prices of the top three AV vendors if they moved away from a subscription model.

The end of this decade will be a very interesting time. I am sure that current anti-virus technologies will be relegated to what they *should* be – minor utility programs. By 2007 anti-virus will be built into the network and chipsets of the latest computers and devices. Virus protection will not take the form of detection or heuristics.

Perhaps in the future the anti-virus industry of the last 20 years will be likened to the tobacco industry – having created and sustained an 'unnecessary danger' purely for profit. Or will this be seen as a curious phase of computing evolution that is identified with the 20th century? Only time will tell.

Did Nick's opinions on the anti-virus industry ruffle your feathers, or do you agree that time is running out for definition-based AV software? VB would love to hear your views – email editor@virusbtn.com.

NEWS

SYMANTEC BUYS AGAIN

Symantec, the AV company that never seems to stop shopping, has revealed its latest purchase: digital security company *@stake*, *Inc*. A *Symantec* spokesperson said of the acquisition, "By joining forces with [*@stake*] we expand the capacity and capabilities of our consulting organization, which allows us to better secure the applications that our customers develop and deploy." *Symantec* has already acquired three companies this year: management software manufacturer *ON Technology Corp.*, anti-spam and email filtering company *Brightmail* and anti-spam start up *TurnTide*. The latest transaction is expected to complete this month.

INCREASE IN INFECTIONS FOR CHINESE

According to China's Ministry of Public Security, 87.9 per cent of computer users in China are affected by malware. The Ministry revealed the figure as part of the findings of its first national survey on Internet information security and computer viruses. Information for the survey was gathered from more than 7,000 organisations, including government bodies, financial and educational institutions and commercial organisations, as well as over 8,400 computer users. The 87.9 per cent infection rate among the computers of users in China is a two per cent increase from last year.

GOVERNMENTS URGED TO DO MORE TO COMBAT CYBERCRIME

The message at a conference organized by the Council of Europe last month was that governments must do more to deal with Internet criminals. The Council of Europe's 2001 Cybercrime Convention, which aims to speed up international cooperation in investigations and extraditions, has been signed by representatives of 30 countries, but is now law in only eight of those countries. Driving home the point that international cooperation is essential for prosecuting cybercrime, Ulrich Sieber, head of the Max Planck Institute for Foreign and International Criminal Law, said: "Effective prosecution with [only] national remedies is all but impossible in a global space."

CORRECTION

VB regrets that an error slipped through the editorial net in the August 2004 *NetWare* comparative review (see *VB*, August 2004, p.14). Despite appearances both in the table for on-demand scanning results and in the results listed in the text, *Eset*'s *NOD32* did not, in fact, miss any samples in the polymorphic test sets. The figure should have indicated an unblemished 100.00%. *VB* apologises for the misinformation.

Prevalence Table – August 2004

Virus	Type	Incidents	Reports
Win32/Bagle	File	47,626	81.57%
Win32/Zafi	File	2,779	4.76%
Win32/Netsky	File	1,431	2.45%
Win32/Dumaru	File	1,197	2.05%
Win32/Mabutu	File	826	1.41%
Win32/Funlove	File	578	0.99%
Win32/Mydoom	File	535	0.92%
Win32/Sobig	File	425	0.73%
Win32/Klez	File	416	0.71%
Win32/MyWife	File	356	0.61%
Win32/Lovgate	File	347	0.59%
Win32/Mimail	File	250	0.43%
Win32/Bugbear	File	201	0.34%
Win32/Swen	File	182	0.31%
Win95/Spaces	File	139	0.24%
Redlof	Script	110	0.19%
Win32/Fizzer	File	102	0.17%
Win32/Valla	File	100	0.17%
Win32/Parite	File	95	0.16%
Win32/Hybris	File	66	0.11%
Win32/Yaha	File	63	0.11%
Win32/Mota	File	52	0.09%
Win32/Sasser	File	52	0.09%
Win32/BadTrans	File	38	0.07%
Win32/Elkern	File	37	0.06%
Win32/Magistr	File	31	0.05%
Win32/Nachi	File	25	0.04%
Win32/Nimda	File	21	0.04%
Win32/Torvil	File	21	0.04%
Win32/Bobax	File	20	0.03%
Win32/Evaman	File	19	0.03%
Kak	Script	18	0.03%
Others[1]		230	0.39%
Total		58,388	100%

[1]The Prevalence Table includes a total of 230 reports across 44 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

VIRUS ANALYSIS

TO CATCH EFISH

Peter Ferrie and Frédéric Perriot
Symantec Security Response, USA

W32/Efish, a member of the W32/Chiton family, contains in its source code (released as part of *29A* magazine) a reference to the television program *The Six Million Dollar Man*. The virus author wanted to call the virus EfishNC ("efficiency"), and referred to it as "Better, Stronger, Faster" (this virus author is not known for humility – in 1994 (s)he named a virus Hianmyt ["high and mighty"]). While the code is indeed better, stronger and faster than comparable viruses, it does have weaknesses. *Symantec* has not received any wild samples of Efish, although the .A variant was published as early as 2002. This suggests that these viruses have not left zoo collections, despite their aggressive infection strategy.

STINGRAY

The infection cycle of Efish starts with an infected program dropping a standalone, unencrypted virus sample to the *Windows* directory, and directing registry hooks to this file. This standalone file, which exists independently of any host program, then infects hosts in a parasitic way. There is no 'direct' infection from one host file to another.

It is worth mentioning that the standalone virus sample is an extremely tortuous, albeit valid, PE file. The PE structure of the file is an abomination of overlapping headers and tables, crafted for the sake of size optimization, which – surprisingly – loads without problem on 32-bit *Windows* platforms, from *Windows 95* to *Windows 2003*. Needless to say, most tools of the trade from *PEDUMP* to *Soft-ICE* have trouble mapping the file properly, and we expect that some anti-virus products would be confused as well.

UNICORN FISH

As with all other known members of the W32/Chiton family, Efish is fully *Unicode*-aware, and selects dynamically between ANSI and *Unicode* routines, as appropriate. These routines include command-line parsing, local network and IP share enumeration, and file enumeration. In some cases, a single routine is capable of processing either type of character sets, by simply changing an AND mask and using the CharNext() API. This is one of the many code optimizations that result in such a small amount of code (around 4kB) that is capable of so much.

There are a number of interesting optimizations in the code. The one that appears most often (and which is the hardest to follow) is the long series of PUSH instructions prior to a series of API calls. The purpose of this seems to be to avoid the reinitialization between API calls of the volatile registers, such as ECX and EDX. One particular example is the code for dropping the standalone virus file, which contains 23 PUSH instructions followed by seven API calls:

```
PUSH    EAX             ;GlobalFree
PUSH    EBP             ;WriteFile
PUSH    ESP             ;WriteFile
PUSH    EDI             ;WriteFile
PUSH    EBP             ;CreateFileA
PUSH    +02             ;CreateFileA
PUSH    +02             ;CreateFileA
PUSH    EBP             ;CreateFileA
PUSH    EBP             ;CreateFileA
PUSH    40000000        ;CreateFileA
PUSH    EAX             ;CreateFileA
LEA     ECX, DWORD PTR [EAX + 7F]
PUSH    ECX             ;MoveFileA
PUSH    EAX             ;MoveFileA
PUSH    EAX             ;GetFileAttributesA
PUSH    EBP             ;SetFileAttributesA
PUSH    EAX             ;SetFileAttributesA
PUSH    ECX             ;DeleteFileA
PUSH    ECX             ;GetTempFileNameA
PUSH    EBP             ;GetTempFileNameA
PUSH    ESP             ;GetTempFileNameA
PUSH    EAX             ;GetTempFileNameA
PUSH    EDI             ;GetWindowsDirectoryA
PUSH    EAX             ;GetWindowsDirectoryA
XCHG    EBP, EAX
CALL    GetWindowsDirectoryA
LEA     EDI, DWORD PTR [EAX + EBP - 01]
CALL    GetTempFileNameA
CALL    DeleteFileA
...
CALL    SetFileAttributesA
CALL    GetFileAttributesA
...
CALL    MoveFileA
CALL    CreateFileA
```

Figure 1. The code for dropping the standalone virus file.

FISH TANKS

Efish is very aggressive when it comes to finding targets. The target selection is contained in three threads.

The first thread periodically enumerates all drive letters, from A: to Z:, looking for fixed and network drives. The second thread periodically enumerates all network shares on the local network, looking for drive resources. The third thread periodically enumerates all network shares on random IP addresses. In each case the virus examines all files in all subdirectories.

BALEENS

Efish examines all files for their potential to be infected, regardless of their extension. First the virus checks if the file is protected by the System File Checker. While the main

file responsible for the protection (sfc.dll) exists in all *Windows* versions that support SFC, the required function is forwarded in *Windows XP/2003* to a file called sfc_os.dll. The method that Efish uses to retrieve the address of exported APIs does not support export forwarding, so the virus resolves the APIs directly from sfc_os.dll on platforms where this .dll exists.

Unprotected files are then checked against a very strict set of filters, which includes the condition that the file being examined must be a character mode or GUI application for the *Intel 386+* CPU, that the file must have no digital certificates, and that it must have no bytes outside of the image. The latter condition is the virus's infection marker.

Additionally, the file must satisfy the needs of the EntryPoint-Obscuring technique (see below).

PILOTFISH

The EPO method that Efish uses is to replace a function prologue with its own code. This method has previously been used by such viruses as Zhengxi (on the DOS platform) and W95/SK (on *Windows*). The virus searches the first section of the file for function prologue code that creates a stack frame, and epilogue code that destroys it. The virus requires that the prologue and epilogue be at least 32 bytes apart, in order for the decryptor to fit.

While it might appear that only the first such sequence is used, this is not always the case. Sometimes a later sequence may be used, or the EPO routine may fail to find a proper sequence even though one exists in the file. This is most likely a coding bug, but it could have been intentional.

FEABUL ENDJINN

Once such a sequence has been found, Efish saves the first 32 bytes of that code, and replaces them with an oligomorphic decryptor. The useful code of the decryptor is 27 to 32 bytes in length, and it is padded up to 32 bytes with ff bytes (an artifact from the memory reuse). The Efish.A engine comprises only about one eighth of the virus body, yet it combines line-swapping, variable load and store instructions and decryption in either a forwards or backwards direction. According to our calculations, there are 23,616 possible valid decryptors and a few invalid ones!

The engine shared by the .B and .C variants adds register replacement, the optional use of 'do-nothing' instructions in the form of INC and DEC of unused registers, and one-byte instructions CMC, STC, and CLD.

The decryptor decrypts the virus body onto the stack and runs it from there. This requires no changes to the attributes of the section in which the virus body is placed within the file, an effective anti-heuristic attack.

DfishNC

A thorough analysis of the engine reveals a lack of optimization in several code sequences and at least two bugs. The result of the first bug is that the PUSH EDI instruction cannot be produced to transfer control to the virus code. However, the code was optimized and that bug was fixed in the .B variant.

The second bug, present in all three variants, causes Efish to produce non-working decryptors in a few rare cases, leading to corrupted replicants. Detection methods based on emulation of the decryptor to recover the virus body are bound to miss such corrupted samples.

BLOWFISH

The decryption is performed using a translate (XLAT) table, in which each unique byte of the virus code is replaced by a unique random value.

The virus author claimed that it is unbreakable, which is clearly untrue, since it is simply a substitution cipher. As we show in our VB2004 conference paper 'Principles and practise of x-raying', several methods exist to break the Efish encryption, and they work quite quickly in practice.

In the .C variant, the author of Efish refined the encryption method a little by taking into account unused byte values from the virus body and reusing slots in the translate table (switching to what is known as a 'homophonic substitution cipher'). Fortunately, efficient attacks still exist against this cipher and, in particular, against the somewhat simplistic implementation in Efish.C. Once again, we refer readers to our paper on x-raying for a thorough explanation.

Efish places its body into the last section of the file, along with the XLAT key table, however it prepends and appends garbage bytes randomly to both the body and the key table, to disguise its true location, and it randomly alters the order in which they are added to the file. If relocation data exist at the end of the file, then the virus moves the data to a larger offset in the file, and places its body and table in the gap that has been created. If there are no relocation data at the end of the file, the virus body and table are placed here (see Figure 2).

STONEFISH

The convoluted code of the virus makes it easy for analysts to overlook one fundamental feature of Efish: the .A and .B variants are 'slow polymorphic' viruses. This term means

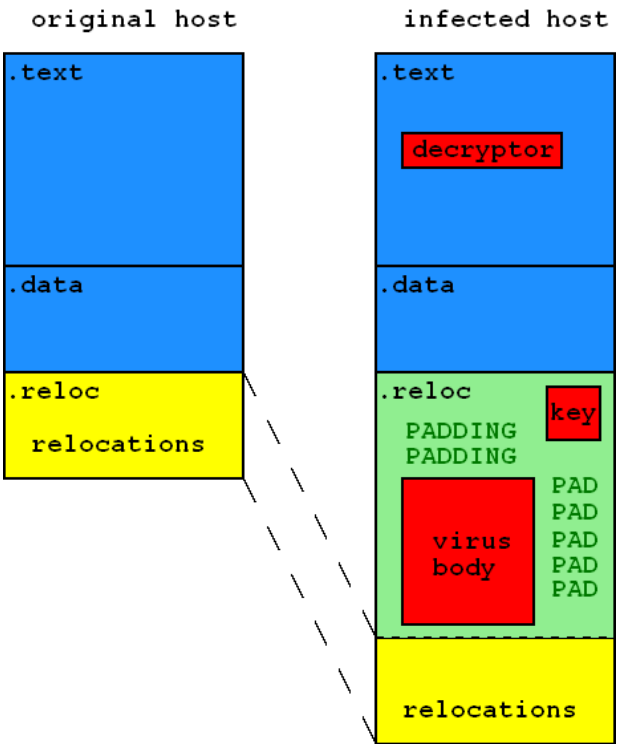


Figure 2.

that the polymorphic decryptor is generated only once in a while, and the same copy is used in the infection of several host programs. In the case of Efish, the decryptor is generated when the standalone sample first runs, before it starts looking for hosts to infect. Additionally, the decryption key, encrypted virus body and layout of the virus segment containing the key, body and random padding, are also generated anew only when the standalone sample starts.

Therefore, all detection methods, whether based on decryptor parsing, emulation, or x-raying of the virus body, must be tested carefully against a range of samples generated from several runs of the virus.

So long, and thanks for all the …

W32/Chiton variant

Type:	Memory-resident parasitic appender/inserter, share crawler, slow polymorph.
Infects:	*Windows* Portable Executable files.
Payload:	None.
Removal:	Delete infected files and restore them from backup. Restore registry.

FEATURE 1

TESTING HEURISTIC DETECTION IN A REAL-WORLD SCENARIO

Andrew Lee
ESET LLC, USA

"*Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write.*"
H.G. Wells (1866–1946).

The results are often published of anti-virus product tests that have no scientific basis, have seriously flawed premises, or demonstrably incorrect methodology. Worse, some tests have all of these faults.

Unfortunately, this is often true of tests conducted by popular computer magazines, whose reviewers seem to think that testing against "a few samples we collected from the Internet and our email" constitutes a valid test of an anti-virus product's detection capabilities. Sadly, it has also been true of some of the tests conducted by recognised AV testing bodies for publication by such magazines – whose interpretation of the results has been wildly misguided.

Reviews in consumer magazines are far more widely read than those in anti-virus industry journals and, in general, their readers are not qualified to understand the (in)significance of the test results. As a consequence, the public's confidence in anti-virus products is at stake.

A classic example, circa 2000, is that of *CNET*'s testing utilising the infamous Rosenthal Utilities (RU). The Rosenthal Utilities generated benign (i.e. non-viral) files, the data part of which contained a portion of a virus. The executable part displayed a message on the screen. It should have been obvious that the detection of any RU-generated file constituted a false positive. However, *CNET* rated the products in the test according to their detection of the RU files. Two years later, *CNET* was not only still making the same mistake [1], but compounding it by altering real viruses such as VBS/Loveletter (i.e. creating new viruses) and trashing products that 'failed' their tests. Eventually, partly due to pressure from the industry [2, 3], *CNET* phased out the use of RU test files.

THE UNDETECTABLES

Particularly misunderstood, and consequently frequently maligned, are the heuristic capabilities of anti-virus products, the testing of which has been woefully inadequate in almost every case this author has seen. This article aims to discuss a scientific basis for real-world testing of heuristics-based anti-virus products, but will begin by describing more general test methodology.

SCIENTIFIC TESTING METHODOLOGY 101

Good testing will aim to prove a hypothesis, will fully define its premises and methodology, and will note any limitations and special considerations.

When comparing products that have different features, it should be stated clearly which features are being tested, and whether the features are common to all products. As far as possible, reviewers should test like against like. Where the testing methodology is likely to have an effect on the product's performance (for instance, testing 'best settings' or default settings), this should be indicated.

Scoring, if any, should reflect only the common features tested – any extra features should either not be scored, or should be scored separately. Any weighting should be clearly defined, and the raw results should be made available. The following are essential to all tests:

- Statistical integrity, using recognised methods.
- Full documentation.
- Presentation of results that arise from scientific analysis of the test data, rather than a subjective view. (It is astonishing how often the final 'score' of a product bears little resemblance to the test data).
- The availability of full sample sets to the product manufacturer and independent bodies (the latter of whom must have the ability to verify any results after the fact).

All too often sample selection is seriously flawed, leading to incorrect test results. The worst mistakes include testing against unreplicated (or non-replicable) files, altered files or renamed files (the latter can be an issue, for instance, if scanning by extension is default, and the test is against default settings).

The size of the sample set is also an issue. Testing products against two or three files will prove almost nothing about the detection capabilities of a product – the smaller the sample size, the greater the likely error. However, there is an interesting paradox here. In virus detection terms, the statistically significant portion of the overall sample set (every virus ever written) comprises less than five per cent of that set. The majority of viruses exist only in the 'zoo' collections of the various AV companies and related labs.

For testing purposes those viruses that appear on the WildList are more significant than those that appear in zoo collections. Furthermore, there are two levels to the WildList: the top list, which constitutes viruses that have been reported by two or more WildList reporters, and the supplemental list, where only one reporter has submitted a sample. A sample of a virus with only one report carries only a little more statistical weight than a zoo virus.

There is little agreement on what constitutes a zoo sample set, and therefore it is likely that most testers will use different test sets (there are frequent cases of tests against "files we found in our email"), which introduces biasing error. It is also quite possible, given the huge number of new samples each month, that there are statistically significant amounts of junk in many zoo sets. Amongst the junk is a large amount of 'grey' material. These may be files that are dropped by a virus (for instance a log file) or files that are used as part of an infection sequence. There may be damaged (non-replicative) files, Trojans, jokes, intended viruses, clean files (which includes the non-viral files dropped by a virus), old files that won't run on modern systems, and any amount of other 'noise' [4, 5].

This, and a lack of agreed definitions about what should be detected or reported, means that testing is increasingly difficult to perform correctly – certainly in terms of the statistical implications.

It is as important to know *why* a product failed a particular test, as it is to know that it failed. If the failure was as a result of flawed test design, it should be possible to prove this from the test documentation. The documentation should be written prior to testing, and should define hypotheses and include the full methodology. Any deviation from the documentation in testing should be noted and justified.

Scoring calculations and formulae should be stated and should be demonstrably linked to the test hypotheses. Any score weighting should be explained fully and justified.

Product failures that fall outside of the stated premises should not be factored into the final scoring. For instance, if a product has a stability or installation problem, and neither the test hypotheses nor premises refer to these, the fault should not be counted as a failure. Where the product cannot be tested because of these problems, it should be noted that the product was excluded on this basis, but no failure rate should be given. Failures in individual polymorphic virus sample sets should show as a percentage of that discrete set (usually >1000 replications per sample), not as a percentage of the overall detection rate, or bias will be introduced.

Scored results should be differentiated according to each hypothesis, and if an averaged score is calculated, the method of its calculation should be stated.

DEFINING A TEST SCENARIO FOR HEURISTIC DETECTION SCANNING

This is not intended to form an exhaustive methodology, nor does it claim to be the best test scenario for testing heuristic product capabilities; rather it will provide a basis upon which a more thorough methodology could be built.

To truly test heuristic detection, the ideal sample set would constitute viruses that are unknown to the product at the time of the test. It is possible to test heuristics against a larger sample set, such as the WildCore, by removing the product's update files (assuming the product works in that way).

A better way would be to 'freeze' a product in time (i.e. not update it for a period), then test it against viruses that emerge in the period between freeze and test. Heuristic capabilities are usually as frequently updated as more conventional detection, so testing against very old (weeks or months) versions of a product will not give a true indication of its capabilities. Ideally, the product on test would be the latest version, without signature updates.

SAMPLE VERIFICATION

The virus sample set must be composed only of verified, replicated samples. The only valid proof that a file is viral is its ability to replicate (ideally to a second generation – but a single replication will usually qualify, with the exception of polymorphic sets).

Using an AV product (or group of products) to determine whether or not something is a virus and perform selection of a sample set is an invalid methodology because neither its error rate, nor its hit rate are statistically determinate. If a test is carried out with un-replicated samples (or assumptions are made about their viability by inference), there will be no scientific basis to the test. Unfortunately there are many cases of tests in which a 'black-box' sample selection is combined with no documentation and no availability of the samples for post-test verification. Especially where sample sets are small, this sort of problem can significantly bias the test results.

PARAMETERS AND HYPOTHESES

Regard should be paid to the default operating mode of the product. For instance, some products have a more aggressive heuristic detection when scanning mail streams (POP3 and SMTP) than in a resident scanner. If this is the case, the most 'real-world' scenario will be to use the correct scan component in its default setting, or a command line version of that scanner with equivalent switches enabled (if available).

Example hypotheses for heuristics testing are as follows:

1. A heuristics-enabled anti-virus scanner will pick up >0% of viruses without the need for updating its signature-based detection.
2. The higher the percentage the better the heuristic analysis can be judged to be.
3. Heuristic analysis detection will not significantly increase the incidence of false positive detection (it may be useful to define this as a percentage, but this is not always necessary).

Example premises:

- Heuristics mitigate a degree of risk of being infected with new viruses.
- A heuristics-enabled scanner will give a better than zero (i.e. better than non-heuristic products) chance of detecting a new or modified virus and protecting against it.

CONTROL TESTS

It is a useful demonstration for each product to be re-tested against the same sample set (e.g. full ItW set) with the full product update at the test date.

The purpose of this is to demonstrate whether, under normal operating and updating conditions, detection would have been available against the sample set.

This test has a key statistical implication. The failure rate of the product against the sample set in its fully updated state is an inference statistic of the overall failure rate against the full ItW test set. The failure rate in the heuristic test is *not* an inference statistic against the entire ItW test set, i.e. it is not correct to say that the failure rate (of detection) against the entire parent set (WildCore) would be the same as against the update frozen heuristic test set (unless the full WildCore test set was being tested entirely heuristically – without any updates), because detection should be available for pre freeze ItW samples. Gaining the correct failure rate inference statistics is critical to the soundness of the test result.

Ideally, a full test of the updated product against the ItW sample core would be carried out, but the full update detection test against the post freeze sample set is a sound inference statistic as long as the sample set is statistically significant (i.e. at least 10 per cent of the size of the parent set – again expected error rates should be calculated).

Ideally, a heuristic approach will maintain a zero per cent false positive (FP) rate as well as a zero per cent false negative (FN) rate. The clean sample test will determine a failure rate.

In this case, the hypothesis does not require a zero per cent FP rate, however, a statistically high FP rate is undesirable, and can be used as a measure of misdirected heuristic aggressiveness.

It should be noted that some test scenarios, for instance, pop3 or SMTP traffic, can have a higher suspicion value

attached to any executable code that is transferred, and false positive rate is unlikely to have direct negative impact on the system stability or customer reaction – but this is no reason not to test for FP. (A false positive in an on-access [memory resident] scanner can have serious implications, particularly for beleaguered corporate support technicians, and it is usual to have a less aggressive heuristic default [if any] set for this reason.)

Ideally, this control set should be an order of magnitude larger than the viral sample set, preferably an order of magnitude larger than the parent set. In the case of any false alert the file alerted against should be recorded, and copies of the file made available to the product manufacturer.

It should also be noted whether the FP was generated against a specifically crafted test file (some testers use denatured or damaged viruses or files deliberately created to look suspicious), or against a 'normal' file that would exist in a standard system. Ideally, failures against such specifically created suspicious files should be noted as a separate statistic, or simply not recorded; after all, we are discussing real-world scenarios.

An ideal clean file control set will take the form of a statistically valid set of verified clean files found in normal end-user systems.

METHODOLOGY

As mentioned briefly above, there are two tests that can be carried out in terms of heuristic capabilities, and it is important to state which is being undertaken. The first measures the product's capability without update over time, and the second measures its capability against specific new malware as it is released.

The first test is likely to show a significant decrease in detection over time, as new types of virus are encountered and, while it is perhaps an interesting test, the results are largely irrelevant, as a correctly installed product (and therefore its heuristics) will be updated – a fact that this test necessarily ignores. It is unlikely, in any event, that evaluation of heuristics on a product that is more than three months out of date reflects the product's actual capabilities.

In the second (and arguably more valuable) test the update freeze point is important, first because each sample must be new for each product, otherwise the heuristics are not being tested, and secondly, because the product should be the latest available to the public before the outbreak of the malware. Most products update automatically every hour, or at least daily, so the test should not be performed using a product that is significantly older than this (say, not older than 12 hours).

STATISTICAL IMPLICATIONS

Testing heuristics will naturally give some degree of bias because the test set is not fixed. If the test is carried out with a different update freeze, or an expanded sample set, bias will be lesser or greater. For this reason, two identical test scenarios using different freeze points and post freeze sample sets will be likely to produce significantly different results.

Assumptions about the overall performance of a product can only truly be made over a long period of time, with repeat tests. This would provide a mean average of performance. A single update – for example, a day later than the original update freeze (to detect the first virus in a new family) – may skew the result significantly if, as in the case of the Netsky and Bagle families, many variants are subsequently released (a modification of a known virus should be easier to detect heuristically than a totally new sample). A wider diversity of families appearing within the post freeze sample set would reduce the detection rate across the whole product range.

It would be nearly impossible to create a totally sound model for heuristic testing, as it would require a full test of every available post update freeze sample set against every possible update freeze point. Having said that, it is possible to create a model that does give a statistically valid result – if sufficient scientific rigour is applied, and the results are expressed in non-absolute terms.

REFERENCES

[1] A *CNET* review (via techrepublic) demonstrating flawed methodology, http://techrepublic.com.com/5102-6270-1043870.html.

[2] In 2000, members of the AV industry, led by Joe Wells, wrote a public letter to *CNET*, denouncing its testing methodology. See http://www.nod32.com/news/joe_wells.htm.

[3] A response to the May 2002 CNET test can be found at http://www.nod32.com/news/cnet_zdnet.htm.

[4] Bontchev, V., 'Analysis and Maintenance of a Clean Virus Library' available online at http://www.virusbtn.com/old/OtherPapers/VirLib/.

[5] Kaminski, J., 'Malware Evolution As A Cause Of Quality Deterioration Of Anti-Virus Solutions', in U.E. Gattiker (Ed.), *EICAR 2004 Conference CD-rom: Best Paper Proceedings*, Copenhagen: EICAR e.V.

[6] Real Time WildList, http://www.wildlist.org/WildList/RTWL.htm.

FEATURE 2

MALWARE IN A PIG PEN – PART 1

Martin Overton
Independent Researcher, UK

PIG TALES

It is often stated that pigs are very intelligent animals – more so than dogs and cats. In fact, pigs are considered to be the fourth most intelligent animals (excluding humans) on the planet [1]. Only chimpanzees, dolphins and elephants (in that order) rate higher in the intelligence stakes.

This article will discuss the use of the SNORT Intrusion Detection System (IDS) with a twist – using it to detect malware by employing various signature creation techniques.

For the uninitiated, SNORT is an IDS that works on *Windows* and UNIX systems and is free (apart from the hardware and manpower costs), very flexible and widely used and respected.

WHY USE AN IDS TO CATCH MALWARE?

Why use an IDS to catch malware? Well, why not? Currently I'm using Bayesian filtering to catch malware too – with great success. I am not saying that virus scanners are useless, or that the use of an IDS is a better method. My reasons for using an IDS to catch malware are as follows:

- Fast-moving threats require quick (and sometimes 'dirty') detection methods.
- Most malware threats that cause problems are network-borne, and therefore an IDS is a suitable tool.
- Many of the signatures I use are created *before* some (if not all) AV companies have detection capabilities for a new breaking threat.
- I am a great believer in defence-in-depth and multi-layered defences against malware. Using an IDS as well as virus-scanning tools offers better overall coverage for a network than merely relying on one or more virus scanners.
- Using an IDS to detect malware propagating across your network means that you have the SOURCE IP address, which will allow faster resolution and clean-up. This is particularly important with mass-mailing worms that forge mail headers as well as fast-spreading/attacking network worms such as Nimda, Slammer, Blaster, etc.

There are other reasons, but in order to list them all this article would need to be several times longer.

FIRST, CATCH YOUR PIG!

You can download, catch and install your pig (SNORT), as well as find more details about its habits, needs, feeding, care instructions and its preferred stabling (most UNIX flavours as well as *Windows*) at http://snort.org/.

Installation on UNIX is very straightforward; even on *Windows* only the additional step of installing PCAP (available at http://winpcap.polito.it/) is required. All other parts, such as reporting, database storage of alerts and management tools, are optional extras.

"These rules are going away. We don't care about virus rules any more."

THIS LITTLE PIGGY CAUGHT MALWARE

When I was first introduced to SNORT I was intrigued to find that, from the early days, SNORT was supplied with a set of virus detection rules (a signature/rule file known as 'virus.rules') which were useful – however, some time ago the following was placed in the virus.rules file: "These rules are going away. We don't care about virus rules any more."

So, many security specialists who used SNORT because it could be used to detect viruses felt somewhat aggrieved that their favourite animal could no longer sniff out new malware. As usual, just as I started to look at using a new tool to detect malware the vendor dropped the very 'feature' that drew me to it in the first place. It seems to be the story of my life!

What could be done to re-educate our pigs? Since no one else (at the time [2]) seemed to be creating rules/signatures for SNORT to detect new malware, I took it upon myself to learn how to create them and make them available to like-minded security professionals.

SNIFFING OUT THE TRUFFLES/MALWARE

So, what can you do to re-train your piggy to sniff out the malware travelling across your networks?

A couple of years ago I decided to try to use SNORT to catch new malware. I installed PCAP and SNORT onto a *Windows* box (I have also installed it on *Linux*) along with the nice front-end known as *IDScenter* from engage security [3], and fired the pig up. Then I installed *ACID* [4], which required a web server (*Microsoft IIS* or *Apache*), *MySQL* [5] and *PHP* [6].

Next came the difficult bit: how to create SNORT signatures, especially for new malware.

EENY, MEENY, MINEY, MO, HOW TO CATCH A MALWARE USING SNORT

Take your freshly caught malware sample, open it in a hex editor (if it is a binary sample) or a text editor (if it is still MIME-encoded in an email). Now examine its entrails and see what they can predict for its future:

"*… I see a teenage nerd, hunched over his/her computer in a room painted entirely in black. Wax-gnarled candle stubs burn fitfully, casting a spectral glow over the proceedings and all the while the teenager is muttering incantations in C/C++, threatening to put some hex on you …*"

A word of caution is needed here: this is not a task for the general end-user population. Only trained and knowledgeable staff who are used to dealing with, working with and handling live malware samples should attempt this – preferably on a dedicated system that is *not* networked, just in case the unthinkable should happen and they should accidentally launch it!

If you cannot justify a dedicated PC, you could use *VMware* instead (remember to disable network support in the virtual machine that you use).

Unless the malware under the knife is polymorphic, pads itself out (with random garbage instructions/code) to fool MD5 hashes, or is encrypted (such as in a password-protected zip file), only a single signature will usually be required to detect it – more will be required if it spreads using other vectors, such as email or peer-to-peer (P2P) networks.

FIRST TAKE YOUR MALWARE ENTRAILS

Let us look at W32/Netsky.p@MM [7] as an example of a typical modern email-based worm that also travels via P2P. How do I create signatures to detect it?

First I find out whether the malware sample is static by hashing (MD5 or SHA1) all the samples I have of it. This may also require me to decode the attachments if they were MIME-encoded when received and perform the same steps on the decoded attachment. If they are zipped, I must unzip them and repeat the steps again on the unzipped, decoded samples.

MIME IF I HAVE A PEEK?

Netsky.p is a static binary image, which means that the MIME-encoded binary image is also static. The next step is to view the sample in a hex editor and select a suitable MIME string to be used to detect the worm when it arrives via email.

A suitable string should be available within the first 30 lines of the MIME-encoded attachment in the email. Try to find a line that is complex, not one that contains mainly 'A's, as otherwise the rule/signature will trigger on perfectly harmless and uninfected email traffic. I usually select at least one full line (72 characters, although sometimes I will use over 100 characters) to ensure that the chances of false positives/negatives are minimised.

Once a likely MIME signature string has been found, this is tested in a simple virus scanner which was created for testing the suitability of SNORT signatures. This scanner – let's call it 'MyScan' – is then run against all captured samples of Netsky.p.

I also test the 'new' signature against earlier members of the same malware family to try to ensure that they will not trigger a false alarm. I also check all the existing signatures in the 'MyScan' database against the new variant – again, this is for false positive testing.

Once the new signature(s) have been tested and any issues ironed out, they are placed into a rule set known as 'malware.rules', which I maintain. These are then made available to other security professionals and researchers, including AVIEN members.

The signature below is one I created and has been very successful. At the end of August 2004 over 3,800 samples have been detected coming to my (personal) mail server from infected hosts.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"W32.NetSky.p@mm - MIME"; content:
"X7soIUEAR4s3r1f/E5UzwK51/f4PdO/+D3UGR/83r+sJ/
g8PhKLw/v9XVf9T"; classtype: misc-activity; rev 1;)
```

Let's break the signature/rule down into its component parts.

The first part is:

```
alert tcp
```

This tells SNORT to send an 'alert' when the signature is matched/triggered – when it sees a 'tcp' packet (you can also test UDP and ICMP traffic too) that contains the signature for the malware.

The next part is:

```
$EXTERNAL_NET any -> $HOME_NET any
```

This specifies that we want SNORT to trigger only when the traffic is coming from an IP address that is *not* one of ours (on any port), but is being sent to one in our IP address range (again, on any port).

$EXTERNAL_NET and $HOME_NET are user-defined variables. You can use the keyword 'any' in place of them to allow the rule to trigger on traffic originating on your network as well as traffic from outside your network address ranges. You can also tie down the detection to specific ports, such as '25' or '110', instead of using the 'any' port keyword.

The next part is:

```
(msg:"W32.NetSky.p@mm - MIME"; content:
```

This tells SNORT to send the alert text "W32.Netsky.p@mm - MIME" to the console, log or database (whatever you use) when the following malware signature (MIME-encoded) is found in the TCP packet:

```
"X7soIUEAR4s3r1f/E5UzwK51/f4PdO/+D3UGR/83r+sJ/
g8PhKLw/v9XVf9T"
```

The final part is:

```
; classtype: misc-activity; rev 1;)
```

This tells SNORT to log the signature match as the 'classtype' of 'misc-activity'. This could be any registered classtype, so you could set the classtype as 'malware' if you prefer. The last part of the signature/rule is the 'rev' statement; this is used to allow revision control so that you can keep track of how many changes you have made to a rule.

BIN HERE BEFORE?

The same steps are followed when rummaging around in a binary sample (EXE, COM, SCR, etc.) looking for a suitable hex signature which can be used to detect the worm as it travels across the network in its P2P (file sharing) mode of operation.

```
alert tcp $EXTERNAL_NET any -> any any
(msg:"W32.NetSky.p@mm - SMB";content:"|4E EB 87 89 77
7E E0 83 B1 94 94 CC E9 F5 97 97 53 95 5C 95 AF C6 40
C5 CA AC 25 8E 47 F1 5D 0B|"; classtype:misc-
activity;rev:1;)
```

A suitable signature is usually at least 32 'hex' characters, each separated by a space. Again, in some cases I will use a longer signature instead.

As you can see, the main difference in the 'content' section of the signature is that hex signatures must also be prefixed and suffixed by the 'l' (broken pipe) character inside the double quotes, whereas MIME signatures are enclosed only in double quotes.

MIME THE BIN

For the more adventurous security professionals out there, you can have multiple 'content' sections within the same rule, and you can even have both binary (HEX) and text (MIME) signatures in the same rule – and lots more besides, but that's another story.

BRINGING HOME THE BACON

The signatures I create are available to all *Virus Bulletin* subscribers (and other selected parties) on my home webserver at: http://arachnid.homeip.net/snort/index.htm. You will need to use the following login credentials (they are case-sensitive):

```
ID = VB2003
Pass = worm!charmer
```

Any assistance is very welcome – either by creating and sharing your own rules or by reporting success or issues with the rules I have created.

This concludes part one of this article as I have covered 'simple' signatures for non-polymorphic or otherwise non-obfuscated malware. The next part of the article will deal with how to write rules/signatures to detect more difficult (obfuscated and encrypted) malware, as well as looking at the use of PCRE (Perl-Compatible Regular Expressions) instead of static binary or MIME strings, in addition to multiple 'content' rules.

For those who are interested, I use my own SNORT rules, as well as running virus scanners, my Worm Charmer, various honeypots, Bayesian filtering, etc. So, you could say that I eat my own pig food.

NOTES AND REFERENCES

[1] Source: http://www.veganpeace.com/AnimalFacts/Pigs.htm.

[2] Now there are several individuals and groups, most notably the maintainers of Bleedingsnort.

[3] http://www.engagesecurity.com/.

[4] http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html.

[5] http://www.mysql.com/.

[6] http://www.php.net/.

[7] First seen 21 March 2004. See http://vil.nai.com/vil/content/v_101119.htm for details.

Photograph of Luther the pig courtesy of Farm Sanctuary, http://www.farmsanctuary.com/.

TECHNICAL FEATURE

HASH WOES

Morton Swimmer and Jonathan A. Poritz
IBM Research GmbH

In a rump session of the August 2004 Crypto conference, where attendees have the chance to give informal (non-refereed) presentations of works in progress, a group of Chinese researchers demonstrated flaws in a whole set of hash functions and the entire crypto community was abuzz. In this article, we will clarify the situation and draw lessons from this incident.

First, we need a little background.

THE NAUGHTY BITS

A hash function h is an algorithm which maps a message (bit string) x of arbitrary length to a digest $h(x)$ of a fixed length – a property we call *compression.* For reasons of practicality, the digest $h(x)$ must be easy to compute for any message x.

One example is the common CRC-32 function. In non-malicious environments, this compression alone can be useful – for example to detect transmission errors on a noisy channel – but in security applications we often use the hash digest as part of an authentication or other cryptographic protocol.

A trivial (but very widespread) kind of authentication using hash functions is the practice, within the open source community, of multiply-posting 'md5 checksums' of software releases, so that a prospective downloader can compare the checksum (actually the hash digest) of the binary which they actually download to the value they have found on various public websites. (Hence a hacker who wishes to Trojanize a software package would have to get his version onto the public servers and *also* get the corresponding hashes to all sites which post these authentication hash values.)

More sophisticated uses of hash functions include various (standardized) digital signature schemes, as well as a cute technique to make what are called 'non-interactive zero-knowledge proofs of knowledge', which are essentially certificates which prove that the issuer has certain knowledge, without revealing all of the details of that knowledge.

CRITERIA

For such security applications, one needs more robust hash functions which have additional properties. We will describe the three most common criteria from [1]. (There are other, more elaborate, criteria we could have used, but such precision is not needed for this discussion.)

For an unkeyed cryptographic hash function h, we generally require one (or all) of the following characteristics:

PRE: *Pre-image resistance* means that, for essentially all given digests y, it is computationally infeasible to find any input x which hashes to that value (so $y=h(x)$). This means we cannot find an inverse function to h that is computationally feasible.

2PRE: *Second-pre-image resistance* means that it is computationally infeasible to find a second input $x' \neq x$ which has the same output $h(x) = h(x')$, given that we know x (and therefore the output $h(x)$).

COLL: *Collision resistance* means that it is computationally infeasible to find distinct inputs $x \neq x'$ that hash to the same outputs, i.e. $h(x) = h(x')$.

Typically, we assume 'computationally infeasible' to mean no better than the brute-force approach, although in formal cryptography this relates to the idea of 'polynomial-time' algorithms (or, rather, algorithms which are *not* polynomial-time).

A candidate hash function h which fails to satisfy PRE or 2PRE is not likely to be useful in security applications: for example, both such failures allow an obvious denial-of-service attack on the simple authentication mentioned above, while failure of PRE usually allows digital signatures using h to be completely forged.

COLL is rather more subtle. If, for example, we have found colliding inputs x and x', and we can get an automated signer to sign $h(x)$, then we can later present the forged x' instead as the signed message; of course, this results merely in a denial-of-service attack (again), unless we have fine control over the collision production.

In fact, there will always be collisions, due to the compression factor of the hash function. What we require of a secure hash is not that there are no collisions, but that it is not possible to generate many collisions with a reasonable amount of computational power.

HISTORY

Modern cryptographic hash history starts with Ron L. Rivest's creation of the MD4 hash algorithm in 1990, which was meant to be an improvement on the slow MD2 algorithm. However, very soon it was suspected that MD4

was not secure enough and in 1992, Rivest supplanted it with MD5, which contained an extra round in the three rounds of the MD4 compression function and changed the functions in some of the rounds. This and other modifications were meant to make the algorithm less symmetric and therefore more secure.

Soon after, MD5 was extended to become HAVEL. Likewise RIPEMD, which appeared a few years later, was also based on MD4.

MD4 BROKEN

Apparently, the suspicions surrounding MD4 were well founded. In 1992, collisions in some parts of the MD4 algorithm were found and by 1995, Hans Dobbertin had found a method of producing meaningful collisions in just a few seconds using the computers of the day [2]. By 1998, Dobbertin had found a way of inverting a reduced strength version of MD4. This means that MD4 completely fails COLL, and its status for PRE and 2PRE is considered essentially broken.

Meanwhile, MD5 was under attack as well. After some successful attacks against the compression function in MD5, Dobbertin extended his MD4 attack to MD5 and was able to produce collisions by modifying the algorithm very slightly.

RIPEMD was also attacked by Dobbertin and collisions were found in a reduced round version of it. In response, the algorithm was modified and RIPEMD-128 (128 bits) and RIPEMD-160 (160 bits) emerged – these are included in the ISO/IEC DIS 10118-3 standard, which was finalized this year.

Meanwhile, in 1993, NIST (National Institute for Standards and Technology) published the Secure Hash Algorithm (SHA, now usually referred to as SHA-0), which was meant to be used with NIST's signature algorithm DSA. Like so many others, it too is based on the now thoroughly broken MD4, so it was little surprise when, in 1995, a modification appeared: SHA-1.

Not only did SHA-1 produce a 160-bit hash code (over the 128 bits of MD4 and MD5), but it included a unique expansion function before the compression, which is attributed with providing greater security. SHA-1, and its sisters, SHA-256, SHA-512 and SHA-384 are now also included in the ISO/IEC DOS 10118-3 standard and are the only hash algorithms allowed by NIST's FIPS 180-2 standard.

THE DOWNFALL OF MD5

After Dobbertin's success with breaking MD4, the focus shifted to breaking MD5 security. A website intending to organize a distributed computer attacking MD5 (as was used in the RSA challenge) was created at http://www.md5crk.com/.

While that effort was focused on a brute-force search for collisions, the more interesting problem of finding a less computationally exhausting attack on the complete algorithm remained elusive. That is, until the presentation given at this year's Crypto conference by Xiaoyun Wang, Dengguo Feng, Xuejia Lai and Honbo Yu [3], all of Chinese research institutes or universities.

Using a multi-processor *IBM pSeries* machine, Wang *et al.* claim to be able to calculate a collision in just over an hour. For the HAVEL-128 extension to MD5, they were able to calculate collisions in 2^6 calculations, as opposed to the brute-force approach requiring 2^{64}. In fact, they claimed that, with their method, collisions can be found by hand calculation for MD4.

Next, they showed two collisions in the original RIPEMD algorithm and finally they mentioned that collisions could be found in the original SHA-0 algorithm in 2^{40} calculations, as well as in the HAVEL-160.

Unfortunately, their presentation (and a paper on the e-print archive site of the International Association for Cryptologic Research [3]) did not provide very much detail about their method. However, they made a convincing argument and it is likely that Wang *et al.* will publish a more detailed paper in the near future so that the crypto community can evaluate it.

Although eclipsed somewhat by the Wang, *et al.* presentation, Antoine Joux (whose ideas the Chinese team used in their work) announced in the same session of the conference the existence of a collision in the original SHA-0 algorithm, effectively lending weight to Wang, *et al.*'s similar statement.

Also in the same session, Israeli cryptographer Eli Biham announced results on the COLL attacks against SHA-1 that he has been waging. So far, collisions have been obtainable in a reduced round version of SHA-1 (40 instead of 80 rounds).

DO WE CARE?

For simple applications of hashes, PRE and 2PRE security is probably sufficient. But before you breathe a sigh of relief, consider this: it is believed that evidence of COLL attacks implies that 2PRE and possibly PRE attacks are also likely to be imminent. This seems to be borne out by our experience with MD4, where first partial, then full COLL attacks preceded useful COLL and then partial PRE attacks. If MD4 were still of interest, there might be a successful PRE attack by now.

But since COLL attacks do not automatically imply PRE and 2PRE attacks, we may be OK for simple uses of hashes. Using MD5 hashes for file integrity or document signing applications will still provide a good level of security until useful collisions can be found.

Applications like SSL may also be secure enough because, while an attacker might try to have a certificate authority sign one version of a certificate and then use its collision later to forge the site's certificate, in practice, signing authorities provide part of the data that then gets signed, which greatly limits the usefulness of a COLL attack in this instance.

However, more complicated cryptographic protocols may rely specifically on a hash function being collision-free. This is apparently the case with ISO/IEC 18033-2 [4], which is still in draft status. These applications will have to look for more secure hash functions.

CONCLUSIONS

MD5 is now considered broken. We can expect folks to be moving away from it in the near future, as the opportunity arises. The same goes for SHA-0, RIPEMD and HAVEL-128.

SHA-1 is now also in doubt, as the crypto community generally believes that COLL attacks are quite possible in the near future, and so also may follow 2PRE and PRE attacks. Thus it seems prudent to replace SHA-1 with SHA-256 or better in security-critical standards, in software, and even in actual digital data which must remain secure for some time.

Furthermore, SHA-1 is hardwired in a whole host of internationally-recognized standards (such as NIST's official digital signature scheme, most of the work of the Trusted Computing Group, *etc.*), and implementations of all such standards must be considered somewhat suspect unless and until each one is examined by experts and concluded even to be safe with a possibly COLL-failing hash function.

Because all of the hash functions we have discussed are based on MD4, it is tempting to lay the blame for this mess on this single fact. However, it is important to note that it has not been proven that COLL-resistant hash functions are possible at all. Also, the entire field of hash functions is at best a black art and is based on the best judgement of a few very talented individuals.

LESSONS

Even a perfect hash function only ever has 'as much security as' half as many bits as its digest size (this notion can be made precise in cryptography). So, in fact, we were *already at risk* before this recent discovery of flaws in MD5 and SHA-1, every time we used such a hash in a protocol with a keyed cryptographic primitive whose key size was more than half the digest size – at risk in the sense that we were getting less security than we thought.

Thus one lesson the current situation should teach us is to be careful about which hash function we use, even simply in terms of its digest size.

We should also take from this incident the fact that we must keep implementations that rely on hash functions flexible. There must be a simple way to replace broken hash function code (or simply increase the size of the digest, when necessary, for a certain number of bits of security, as just noted).

Furthermore – and just as important – there must be a clear migration path to the new hash function. Not only must the hash function be modularized, but the storage available for the hash code must be expandable. It may also be necessary to keep the old, defunct hash codes around and space needs to be allocated for that, too.

Standards bodies should make their designs modular in a similar fashion, so that any future advances in the cryptanalysis of hash functions do not invalidate the entire design, but merely require more powerful hashes to be plugged in.

Given the pervasiveness of MD5 usage, it will be interesting to see how long it takes until we have completely migrated away from MD5. There are still some programs in use today that employ MD2!

BIBLIOGRAPHY

[1] Menezes, Alfred; van Oorschot, Paul C.; Vanstone, Scott A., *Handbook of Applied Cryptography*, 1997, CRC Press.

[2] Hans Dobbertin, 'The Status of MD5 After a Recent Attack', *CryptoBytes* vol. 2, no. 2, 1996, ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf.

[3] Xiaoyun Wang; Dengguo Feng; Xuejia Lai; Honbo Yu, 'Collisions for Hash Functions MD4, MD5, HAVEL-128 and RIPEMD', International Association for Cryptologic Research, 2004, http://eprint.iacr.org/2004/199/.

[4] Working Group 2 of ISO/IEC JTC 1/SC27, 'ISO 18033-2: An Emerging Standard for Public-Key Encryption', 2004, available from http://www.shoup.net/iso/.

LETTERS

SETTING THE RECORD STRAIGHT ON KASPERSKY ANTI-VIRUS PERSONAL 5.0

Having read *Virus Bulletin*'s recent review of *Kaspersky Anti-Virus Personal 5.0* (see *VB,* September 2004, p.16), I was dismayed to learn of your reviewer's concerns about the product's update settings. He comments:

"*Since updates occur automatically only at three-hourly intervals, no update is performed by default when a machine is booted. What is more, there is no provision for an update on boot. If, for example, a user arrives home from holiday and powers up their machine, the virus definitions might be as old as two weeks for up to three hours. Given some of the fast-spreading worms of recent months this would be ample time to infect the machine before the update is triggered.*"

I would draw your attention to several points.

1. Although there is no 'update on boot' option, the product maintains its own task list. If a PC is switched off and misses an update, this is logged in the task list and the next time the PC is connected to the Internet, the update will be performed immediately.
2. Kaspersky Anti-Virus Personal can *easily* be configured to update hourly.
3. Kaspersky provides updated virus definitions *hourly* – far more frequently than other anti-virus vendors. Even using default settings, Kaspersky customers are only ever three hours away from a new virus definition file. This compares very favourably to products with daily updates (where a user could be 23 hours and 59 minutes away from the next update) or those with weekly updates (where a user could be six days, 23 hours and 59 minutes away from the next update).

In light of the above, I'm sure you would have to agree that neither your reviewer nor your readers need have 'concerns' about the updating provisions in *KAV Personal 5.0.*

David Emm
Senior Technology Consultant, Kaspersky Lab, UK

VB RESPONDS

As a reviewer I tend to be concerned by matters which might only warrant a disinterested shrug in other circles. I would be happy to state that any default settings which do not take advantage of all updates available – and KAV is admittedly not the worst offender – are of sufficient concern to warrant a mention. Whether an end-user finds this worrying is their decision to make, based upon the facts I present.

Matt Ham

MACS AWAY!

In his comment in the August 2004 issue of *Virus Bulletin*, David Harley asks: "Are Mac users, especially SOHO and home users, really better at keeping up with patches than their PC-using equivalents?" (see *VB*, August 2004, p.2).

The answer is that Mac users are probably not any better at keeping up with patches – but then OS X takes care of all this for you. As far as I (as a fairly long-term OS X user) am aware, Apple has always made patching a breeze. You sit at your computer, clicking, pointing, drooling, and up pops a box saying: "Security Update – please just click the install button"; minutes later, you have the latest security update installed.

Not only that, but if you do not install the security update, you will periodically be harassed until you have done so. The result: there are not many unpatched OS X machines floating around.

As far as I can tell, recent versions of *Windows XP* provide the same functionality – so why are we not virtually free of *Windows* malware too?

Presumably the reason is a combination of the fairly large number of 'legacy' *Windows 98*, *Windows Me* and *Windows 2000* machines (on which one was forced to search out security patches proactively, and which thus remain unpatched) that are in use, and *Microsoft*'s deliberations about allowing people with pirated copies of *Windows* to install security updates.

One might also consider *Microsoft*'s vulnerability-to-patch cycle to be longer than one might hope, but, as far as I am aware, there are very few pieces of malware that use 'zero-day' exploits.

I am a big believer in the philosophy that no operating system is secure, as long as it has a user in front of it – however, *Microsoft*'s previously clunky update procedures, and the baffling array of different methods for patching your *Linux* and *BSD* machines, make OS X a comparatively hostile environment for malware.

Other comments in the opinion piece suggest to me that David is not particularly well acquainted with the venerable OS X (it is exceptionally difficult to log in as root unless you are really trying to, for example). I would suggest David spends more time with it (as I would suggest to anyone) as it is a very capable, powerful, and user-friendly OS.

Pete Sergeant
Independent writer, UK

Do you have an opinion to air? Send your letters to comments@virusbtn.com.

PRODUCT REVIEW

MKS_VIR 2004

Matt Ham

mks_vir is produced by Polish anti-virus company *MKS*. *MKS* was originally a one-man operation set up by the late Marek Sell. Marek was still very involved with *MKS* until his death earlier this year. As with many smaller anti-virus companies, *MKS* has a solid user base in its home country and is now taking steps to expand beyond those borders.

THE PACKAGE

It comes as no surprise that the product was supplied in two versions. The electronic version was supplied as an English language version and consisted of the application and documentation. This could be installed in Polish if required. The boxed set, meanwhile, was supplied as a Polish language version.

The boxed version arrived with a number of *MKS*-branded goodies: a pen, a coaster, a lanyard, a box of matches and, most surprisingly, an optical mouse. This amount of loot indicates that home-users are being targeted.

Documentation and registration for the boxed version was provided only in Polish. Theoretically, the program can be installed in English or Polish. However, attempting to install from the CD resulted in an error message – in Polish. Although I made the happy discovery that my understanding of Polish is better than I thought, the review was performed on the electronic English language version on *Windows XP*. The documentation indicates that all common *Windows* versions are supported.

An administrator application, mks_administrator, is also available and was supplied on the CD in the boxed version. This requires a *.NET*-capable platform in order to be installed and operational and *.NET* is available on the CD.

DOCUMENTATION AND WEB PRESENCE

Documentation for the English version was supplied as a PDF. This seemed to be a pretty much direct translation of the Polish manual, with some interesting, less than perfect translations. For example, the program is referred to throughout the documentation as a 'packet' rather than a package. Strange phraseology aside, however, the content is good. Better still was the fact that the documentation was rarely needed, since operation of the software proved, by and large, to be intuitive.

As would be expected from a company that has only recently started to make serious inroads abroad, the *MKS* website is much more sizeable in its Polish version than in its English version. The main site is located at http://www.mks.com.pl/, with the English version available at http://english.mks.com.pl/.

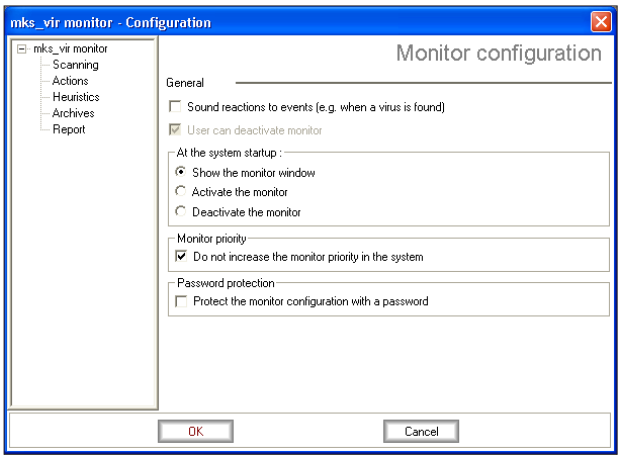
At first glance the site is relatively spartan, with a noticeable lack of a news section. Rather worse is the fact that at least one link was found to be broken due to a typographical error in a URL. Online ordering is not yet available from the website, although this aspect of *MKS*'s activities may become more polished as time progresses. The website allows contact with *MKS* support and download of products and updates – so it is sufficient for most users' needs. The lack of a searchable virus database is its biggest flaw in my eyes.

INSTALLATION AND UPDATE

The application was supplied as a 12MB executable. When launched the first option is a choice of language. After agreeing to the comparatively short licence agreement the next step is to input the licence number. Apparently a holographic sticker should be affixed to any machine upon which the program is installed – but no such object was to be found in the boxed version, let alone the electronic one. Somewhat disturbingly, the boxed version's licence number was out of date when received – not an ideal situation.

From here matters become slightly less strange however, with the choice of user name, installation path and the choice as to whether program configuration should be performed as soon as installation is complete. The default option here is to perform configuration when installation is complete, and was not changed.

With a few file transfers out of the way, the monitor configuration application is launched. Here, the on-access scanner may be activated and various options selected. There are numerous options here, since the application is



essentially the same as that available for adjustment of the on-access service within the program. Configuration is performed through a tree, there being a general root with scanning, actions, heuristics, archives and reports all configurable at this stage.

The first option presented is the choice of whether or not sounds are produced when viruses are detected – an unusual choice for this stage in the proceedings. The monitor may not be disabled by users from within this area, although a check box suggests that this is possible elsewhere.

Next is the more important choice of startup behaviour. The default is for the monitor to display the configuration screen at start up – although the monitor may also be activated or deactivated as startup behaviour. It seems rather odd that the configuration screen is the default behaviour – since this will not be required often and I imagine that its presence will only serve to enrage customers who are in a hurry to use their machine. Monitor system resource usage is limited by default but this limitation may be removed here, and there is an option to password-protect configuration.

Scanning options are comprehensively configurable, with the default being scanning of all executed and accessed files, plus those copied or downloaded. All files are scanned, but this can be changed to a list of extensions (referred to as 'Hosts') and exclusions may also be applied. Unusually, floppy access scanning can be removed independently, and there is an option to scan floppies during system shutdown. Oddly, memory and MBRs are not scanned when the on-access scanner is activated, but this setting can be changed if required. Finally in this area the detection of dialers or spyware and adware can be disabled.

The default action of the product on finding an infection is to ask the user whether to disinfect, delete or rename files. Information is provided as to whether the relevant action has been successful. Backups of disinfected files can be forced. Quarantine settings are also altered here, though the default settings seem suitably sensible.

Heuristics are enabled by default – though this feature is accompanied by a warning that heuristics can trigger false alarms. Heuristic sensitivity may be set as low, standard, high or very high. Standard is the default and recommended setting here.

Archive scanning is disabled by default, which is not entirely surprising: overheads for archive processing can make on-access scanning of such files prohibitive. However, with such settings in place recent worms which arrive as archives will be able to be transferred without interference and it is perhaps for this reason that *mks_vir* allows archives that have been copied and downloaded to be targeted specifically for archive scanning. With regard to the overheads, users may be informed if a particularly large file is about to be processed. The depth of recursion within archives may also be limited.

Finally in this setup area the report branch offers some predictable controls. The size of logs is limited by default – though most users will be hard-pressed to fill a 128kB log file with infection reports. This is also the area where infection reports may be inspected.

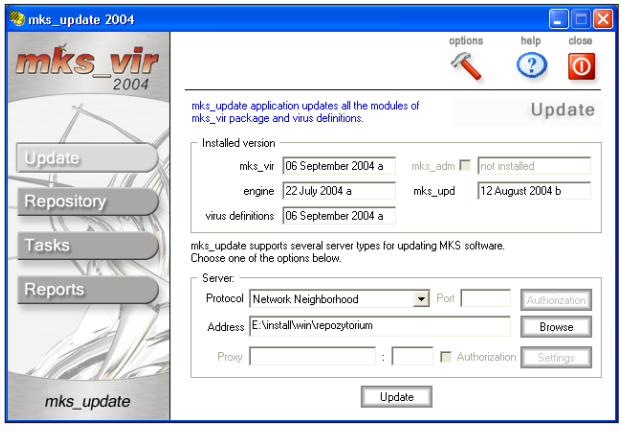
With the on-access scanner configured, the update settings are next to be selected. Unusually, the default here is to use a local repository directory rather than the *MKS* updates server. It is also possible to specify the location of this repository from this dialog. With this configuration complete, the login details for the scheduler must be set and then the installation process is complete.

This is one of the longer installation processes I have seen, since the on-access scanner is fully configured before installation is complete. The good news is that a fairly knowledgable user will likely end up with a more suitable configuration as a result. On the down side, a less knowledgable user will end up with a massive headache and, one hopes, change nothing. Both scenarios could be improved slightly by a few changes to the defaults offered.

Updating was not quite as fast as had been hoped. The operation was successful, but despite using a locally situated file repository the process of file transfer was somewhat less than lightning fast. This may be explained by there being some obvious file integrity checking going on – both for the update application and the downloaded files themselves.

FEATURES

Following the installation of *mks_vir* numerous programs are available from the start menu. Of these, the anti-virus monitor, mks_update and quarantine have already been mentioned. Mail scanner, component registration, uninstallation and a provision to create a DOS version disk for *mks_vir* are also included. This leaves only the



on-demand anti-virus scanner component for a more detailed examination.

The main screen when the scanner is launched shows information on the applications that are currently active and on version dates for various components. If all is well in these areas, a green smiling face or a blue information tab is displayed, while any causes for concern result in a red warning text. From here, the on-access scanner, mail scanner, or updater applications may be launched via icons.

General options are also altered in this screen, again from an icon at the top of the screen. Although they look slightly different from those in the on-access settings area the settings here are virtually identical, with only slight variations in layout and in those options not applicable to on-demand scans. Once again, the sound effect option takes pride of place as the first setting to be configured.

Additional options involve the scanning of process memory, this being noted as slow, and of DBR areas. One nice feature is that if the scanner has problems with access to network resources a separate user account can be supplied to which it will automatically log in. Under what circumstances this access will return to the user permissions is not indicated. Presumably there are some potential security implications to this ability to raise privileges.

By default, suspicious files are sent to *MKS*, which should aid with the fine-tuning of heuristics. The problems encountered with system restore and some infections are also catered for – an option within *mks_vir* allows the feature to be turned off before scanning.

One notable omission is that there is no option to configure a list of extensions to be scanned on demand – all files are scanned. This is at odds with the on-access scanner, where one must assume that all-files scanning may be undesirable for reasons of overheads. Since exclusions can be specified on demand, it is easy enough to decide not to scan specific files – which is, in fact, usually safer than deciding which files should be scanned and attempting to remember all the files that should be targeted.

With configuration covered all that remains is the creation and execution of tasks. All tasks inherit the general configuration options – so the creation of a target is all that is required when a task is created afresh. Generic tasks exist already which cover the usual areas – files, drives and directories may be chosen for the purpose of user-created tasks. The ability to adjust scanning parameters individually for each new task would be a good improvement here.

SCANNING

Scanning was performed on the *Virus Bulletin* test sets, and the full scanning results will be available in the November 2004 edition of *Virus Bulletin* as part of the *Windows 2003 Advanced Server* comparative review.

Preliminary results were certainly satisfactory on infected files, although four false positives were noted in the clean test sets using the default settings. The files alerted on were all from rather ancient anti-virus utilities which contain unencrypted strings from those viruses they were designed to detect. Such false positives tend to be easily dealt with.

The same scan was performed with the most paranoid settings of heuristics and both the result and the files that were falsely detected were identical.

CONCLUSION

One point of interest in *mks_vir* is the extent to which the on-access functionality seems to be favoured with more control options and is selected for configuration during the installation procedure. This indication that on-access scans are considered more important for day-to-day protection is unusual. The default settings for the monitor are perhaps an area subject to improvements but the general principle is sound.

Having both Polish and English versions of the package available for review demonstrated many contrasts in the way *MKS* appears in its different incarnations. It was soon obvious that the English version is not as long-standing as the Polish version, with documentation and translations showing signs of imperfection. There were a number of problems with the boxed version on an English platform. That said, such problems are among the simpler to deal with – time being all that is required for developers to overcome teething troubles and translation issues.

Initial impressions of the underlying functionality of the product were much more satisfactory. Few problems were encountered – and those that were encountered, such as false positives, are the sort of problems to be expected of products in their first few runs against our test sets. With a full detection review imminent, it will be interesting to observe *mks_vir*, both in the short term and even more so over the long term.

Technical Details

Product: *mks_vir 2004.*

Test environment: Identical 1.6 GHz Intel Pentium machines with 512 MB RAM, 20 GB dual hard disks, DVD/CD-ROM and 3.5-inch floppy drive running *Windows XP Professional*.

Developer: *MKS Sp. z o.o.*, ul. Fortuny 9, 01-339 Warsaw, Poland; tel +22 331 33 80; email office@mks.com.pl; web http://english.mks.com.pl/.

END NOTES & NEWS

SecurIT Summit takes place 18–20 October 2004 in Montreux, Switzerland. Streamlined conference sessions will offer a variety of case study presentations, panel debates, interactive 'think-tanks' and workshops. See http://www.securit-summit.com/.

RSA Europe takes place 3–5 November 2004 in Barcelona, Spain. More information, including track sessions and speaker details are available from http://www.rsaconference.com/.

The 31st Annual Computer Security Conference and Expo will take place 8–10 November 2004 in Washington, D.C., USA. 14 tracks will cover topics including wireless, management, forensics, attacks and countermeasures, compliance and privacy and advanced technology. For details see http://www.gocsi.com/.

The ISACA Network Security Conference will be held 15–17 November 2004 in Budapest, Hungary. Presentations will discuss the technologies, and the best practices in designing, deploying, operating and auditing them. See http://www.isaca.org/.

The seventh Association of Anti-Virus Asia Researchers International conference (AVAR2004) will be held 25–26 November 2004 at the Sheraton Grande Tokyo Bay hotel in Tokyo, Japan. For details see http://www.aavar.org/.

Infosec USA will be held 7–9 December 2004 in New York, NY, USA. For details see http://www.infosecurityevent.com/.

Computer & Internet Crime 2005 will take place 24–25 January 2005 in London, UK. The conference and exhibition are dedicated solely to the problem of cyber crime and the associated threat to business, government and government agencies, public services and individuals. For more details and online registration see http://www.cic-exhibition.com/.

The 14th annual RSA Conference will be held 14–19 February 2005 at the Moscone Center in San Francisco, CA, USA. For more information see http://www.rsaconference.com/.

The E-crime and Computer Evidence conference ECCE 2005 takes place at the Columbus Hotel in Monaco from 29–30 March 2005. A reduced daily registration rate of 150 Euro per delegate applies until 21 November 2004. For more details see http://www.ecce-conference.com/.

The first Information Security Practice and Experience Conference (ISPEC 2005) will be held 11–14 April 2005 in Singapore. ISPEC is intended to bring together researchers and practitioners to provide a confluence of new information security technologies, their applications and their integration with IT systems in various vertical sectors. Papers for the conference may be submitted until 1 November 2004. For more information see http://ispec2005.i2r.a-star.edu.sg/.

The 14th EICAR conference will take place from 30 April to 3 May 2005 either in Malta or in Edinburgh. Authors are invited to submit non-academic papers, academic papers and poster presentations for the conference. The deadline for submissions are as follows: non-academic papers 26 November 2004; academic papers 14 January 2005; poster presentations 18 February 2005. For full details of the conference (including imminent announcement of the location) see http://conference.eicar.org/.

The sixth National Information Security Conference (NISC 6) will be held 18–20 May 2005 at the St Andrews Bay Golf Resort and Spa, Scotland. For details of the agenda (which includes a complimentary round of golf at the close of the conference) or to register online, see http://www.nisc.org.uk/.

The third International Workshop on Security in Information Systems, WOSIS-2005, takes place 24–25 May 2005 in Miami, USA. The workshop will gather academics, researchers, practitioners and students in the field of security in information systems and will present new developments, lessons learned from real world cases, and provide opportunities for exchange of ideas and discussion on specific areas. See http://www.iceis.org for details.

NetSec 2005 will be held 13–15 June 2005 in Scottsdale AZ, USA. The program covers a broad array of topics, including awareness, privacy, policies, wireless security, VPNs, remote access, Internet security and more. See http://www.gocsi.com/events/netsec.jhtml.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery: £195 (US$310)

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 www.virusbtn.com

vbSpam supplement

CONTENTS

NEWS & EVENTS

SYMANTEC TO BLOCK PHISHING

Anti-virus and anti-spam vendor *Symantec* has announced a new service to help identify and block phishing scams.

The *Online Fraud Management Solution*, which is aimed primarily at financial services companies, will use probes and decoy email accounts to collect, analyse and identify new phishing scams. When new scams are identified, *Symantec* will create filters to block the messages. The filters will be deployed automatically to consumers who use *Norton AntiSpam* and *Norton Internet Security*.

In addition, *Symantec* says that it will notify any *Online Fraud Management* customer whose name has been used fraudulently in the scam so that it can work together with law enforcement agencies to track down the perpetrators of the scam.

IETF REJECTS SENDER ID

The Internet Engineering Task Force (IETF) has rejected *Microsoft*'s preferred anti-spam specification, Sender ID, due to a possible intellectual property rights conflict.

Members of the IETF's working group MARID (MTA Authorization Records in DNS) reached a rough consensus last month that *Microsoft*'s Sender ID specification should not be made a mandatory part of the eventual standard.

The sticking point is that *Microsoft*'s has stated that it has applied for patents which could affect Sender ID, but has failed to reveal any details. MARID co-chair Andrew Newton posted to the group: "The working group has at least rough consensus that the patent claims should not be ignored," saying that, since the patent applications were unavailable and members of the group were unable to determine exactly what the claims might be, MARID should not undertake work in this area. However, Newton added that the group would not rule out the possibility of work with Sender ID, should the status of *Microsoft*'s patent claim or its associated licence change in the future.

Both the Debian operating system project and the Apache server project have also stated that they will not implement Sender ID, reasoning that its current licensing terms are incompatible with open-source licences.

THE BOUNTY HUNTERS ARE HERE

In a report to Congress, the US Federal Trade Commission (FTC) has said that the US government must be prepared to be generous with its rewards if it decides to encourage 'bounty hunters' to track down email spammers.

According to the FTC, rewards of up to $250,000 will be required to encourage people to snitch on friends or acquaintances who send out bulk unsolicited email. The FTC also said that the cash will have to come from the federal budget, rather than settlements collected from spammers.

But the FTC is not convinced of the effectiveness of such a bounty scheme: "The commission does not believe that the vast majority of consumers who are now forwarding 300,000 pieces of spam daily to the FTC spam database are likely to be a good source for such information," the FTC said in a report to Congress.

EVENTS

INBOX East takes place 17–19 November 2004 in Atlanta, GA, USA. The event will feature over 50 sessions across five tracks: systems, solutions, security and privacy, marketing and 'The Big Picture'. See http://www.inboxevent.com/.

Following the success of the First Conference on Email and Anti-Spam (CEAS 2004), the second conference will be held in summer 2005 (date and venue yet to be announced). A low-volume mailing list has been set up for CEAS conference-related announcements – sign up by sending a message with the body "subscribe ceas-announce" to majordomo@lists.stanford.edu.

FEATURE

TRUST NETWORKS FOR EMAIL FILTERING

Jennifer Golbeck
University of Maryland, USA

The fact that spam has become a ubiquitous problem has led to much research which concentrates on the development of tools to identify spam and prevent it from reaching the user's mailbox. Less effort has been devoted to a question that is related to spam, namely: how to distinguish important messages from unimportant ones. The TrustMail project uses reputation ratings in social networks to score emails and allow users to sort messages based on this score.

BACKGROUND AND INTRODUCTION

Whitelist filters are one of the methods used for identifying legitimate emails and filtering out spam. In these systems, users create a list of approved addresses from which they will accept messages. Messages received from whitelisted sources are delivered to the user's inbox, while others are filtered into a low-priority folder. These systems do not guarantee that all of the filtered messages will be spam, but the whitelist makes the inbox more user-friendly by showing only those messages that are definitely *not* spam.

However, there are two major problems associated with whitelisting. Firstly, there is a burden placed on the user to maintain a whitelist, and secondly, a number of valid emails from non-whitelisted senders will almost certainly be filtered into the low-priority mailbox. If that box contains a lot of spam, the valid messages will be difficult to find.

Other approaches have used social networks for message filtering. Researchers Boykin and Roychowdhury [1] created a social network from the messages received by a user and, using the structural properties of social networks, identified the messages as 'spam', 'valid', or 'unknown'. Their method was able to classify about 50 per cent of a user's email into the spam or valid categories, leaving 50 per cent to be filtered by other techniques.

The TrustMail approach takes some of the basic premises of whitelisting and social network-based filtering and extends them. In this system, users assign a 'reputation' or 'trust' score to people they know. Email messages can then be scored, based on the reputation rating of the sender. When an email is received, an algorithm is applied to the social network that makes a recommendation about the reputation of the sender. The score appears next to the message.

This works like a whitelist, in that users can assign high reputation ratings to the people who would normally appear on a whitelist. This preserves the benefit of making the inbox more usable by making 'good' messages prominent. The added benefit is that, by using the social network, scores will appear next to messages from people with whom the user has never had contact. Since scores are inferred rather than taken directly from a list, fewer valid messages will be filtered into a low-priority mail folder. Furthermore, the initial set of ratings allocated by the user creates points of connection into the network, increasing the number of people for whom ratings can be calculated.

This mechanism is not intended to replace spam filters or any other filtering system. We see our system of reputation scores as being compatible with whitelists, spam detectors, and other social network-based filters, and expect that a combination of all of these strategies would provide the maximum benefit to the user.

TRUSTMAIL: A PROTOTYPE

TrustMail is a prototype email client that adds reputation ratings to messages. Essentially, it is a message scoring system.

While TrustMail will give low scores to spam, it is unlike spam filters that focus on identifying bad messages. Its true benefit is that relevant and potentially important messages can be highlighted, even if the user does not know the sender.

Consider the case of two research groups working on a project together. The professors that head each group know one another, and each of the professors knows his own students. However, neither is familiar with the students from the other group. If, as part of the project, a student sends an email to the other group's professor, how will the professor know that the message is from someone worth paying attention to? Since the name is unfamiliar, the message cannot be distinguished from other not-so-important mail in the inbox. This is exactly the type of situation upon which TrustMail improves.

The system searches along paths in a social network that is augmented with reputation ratings. Using the values in the network, we make a calculation to recommend how much the recipient should trust the sender. Thus, in the above scenario, the professors need only to rate their own students and the other professor. Since the reputation algorithm looks for *paths* in the network (not just direct connections), there will be a path from the professor of one research group to the students of the other group through the direct professor-to-professor link. Thus, even though the student and professor have never met or exchanged correspondence, the student's message achieves a high rating because of the intermediate relationship.

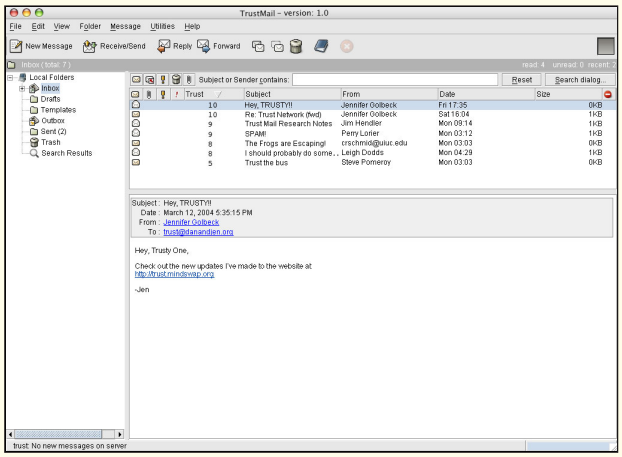


Figure 1. The TrustMail Interface. More information and a download are available at http://trust.mindswap.org.

CREATING AND MAKING CALCULATIONS

The only requirement for this system is that a network be created where individuals assert their reputation ratings for one another. Many of the large email providers will be able to take advantage of their own population of users as a starting place. Service providers like *Microsoft*, *Yahoo!* and *AOL* have access to large numbers of users. *Google* would have an even more explicit set of data available if it were to integrate *Gmail* and the *Orkut* social networking website. To make an Internet-wide reputation network that will allow ratings to be made for any individual in any system is slightly more complicated. Our prototypes use a semantic web-based approach with publicly accessible social network data that is distributed around the web.

The difference between our system and other social network approaches to email filtering is that we rely on the entire network. As such, we can provide information about people with whom the user has never had any contact.

A relatively simple method is used to infer how much one person should trust another person in the network. If Alice receives a message from Bob, Alice will check to see if she has rated Bob. If Alice does not have a rating for Bob, she asks all of the people to whom she is connected to give her their ratings of Bob. Alice then averages the values they return, giving more weight to the ratings from highly trusted neighbours than for lower trusted neighbours. Each of the neighbours repeats this process to get the values that they pass to Alice.

An interesting point to note here is that calculations are made from the specific perspective of the recipient, not universally. This means that Alice's calculated rating of Bob could be high, while someone else's rating could be low, depending on their ratings of people in the network. Our experiments show that this personalization increases the accuracy of the ratings by taking variations of opinion into account.

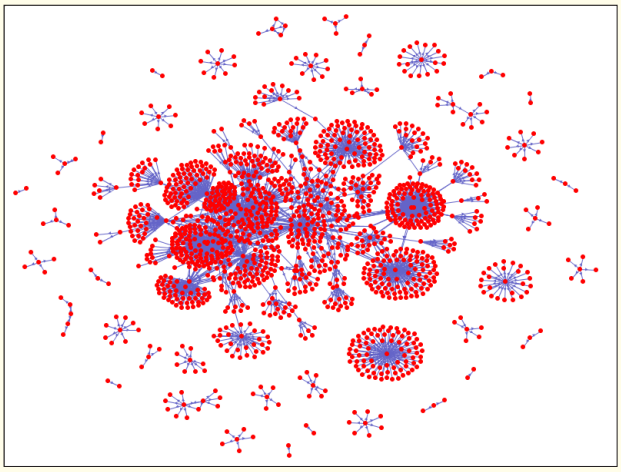


Figure 2. The reputation network developed as part of the semantic web trust project at http://trust.mindswap.org. This network was used in the development of our prototypes.

CONCLUSIONS

The real benefit of this method is that, with a highly accurate metric, valid emails from unknown senders can receive accurate scores because of the connections within the social network. Thus, the system complements spam filters by helping to identify good messages that might otherwise be indistinguishable from unwanted messages.

The ratings alongside messages replicate the way reputations work in social settings, where strangers will often introduce themselves with references to mutual acquaintances. In the context of mail, TrustMail lowers the cost of sharing trust and reputation judgments across widely dispersed and rarely interacting groups of people. It does so by gathering machine-readable encoded assertions about people and their trustworthiness, reasoning about those assertions, and then presenting those augmented assertions in an end-user friendly way.

The author thanks James Hendler of the University of Maryland, College Park, who serves as advisor on this work and is a co-author of many related research papers.

REFERENCES

[1] Boykin and Roychowdhury, 'Sorting e-mail friends from foes: Identifying networks of mutual friends helps filter out spam,' *Nature Science Updates*, 16 February 2004.

SUMMARY

ASRG SUMMARY: SEPTEMBER 2004

Helen Martin

Despite a smaller amount of traffic on the ASRG list this month, there was plenty of debate. The main topics of discussion were address verification and SPF.

Markus Stumpf kicked off a discussion on valid address lists. Previously, the argument against address verifiers has been that spammers would be able to use those services to clean their lists or gather lists of valid recipients. However, Markus queried, "Is 'don't tell the spammers whether an address exists' still true for today's situation?" He argued that, "with all the spam networks of thousands of hosts, spammers don't seem to care about hammering dictionary attacks against SMTP servers. Giving them a chance to find out which addresses exist would not really change much for the existing recipients or the targets of dictionary spams, but it would save us all a big portion of mail to non-existent users, a lot of bounces stuck in the queue and a lot of bounces to faked sender addresses."

Jeff Silverman felt that the address verifiers should be transparent to the sender – the query should be made by the recipient: "If I send you a mail, your mail client should query the address verifier to see if [my address] is a valid address. If so, [it should] deliver it to you, otherwise [it should] drop it silently. The sender doesn't know if the message was delivered or not." It would be possible, but difficult for a spammer to fake a valid address, he said, because they would not know what the valid addresses were.

Markus Stumpf posted a link to a *BBC News* story which reported that *CipherTrust* had found that 34 per cent more spam than legitimate email is passing through SPF checks (http://news.bbc.co.uk/1/hi/technology/3631350.stm). Richard Rognlie was the first of many to express his disappointment that the journalist had failed to understand that SPF is not, in itself, designed to stop spam, but to prevent forgery. Peter Bowyer concurred saying, "*CipherTrust* correctly reported that spammers are publishing SPF records. What they didn't emphasise enough is that this means that SPF is *working as intended*. In order to pass SPF checks, the spammer has to be using a registered domain over which they have DNS control – which is several steps towards accountability. The press unfortunately picked up on the wrong part of the message, and sensationalised it way out of context."

There followed a discussion about whether it was the journalists or the PR representatives of *CipherTrust* that should be held responsible for the inappropriate emphasis of the story – Anne P. Mitchell pointed out that, while the company's press release did contain all the facts, it was worded in such a way that it could lead the relatively ill-informed to conclude that SPF had 'failed' in 'stopping spam'. [*Hmm, PR departments sensationalising stories for publicity, where have we seen that before …? Ed*].

Eric S. Raymond said that if SPF deployment does nothing other than stop the thousand bogus bounces he receives each day as a result of spammers 'joe-jobbing' his domain, it is a winner. Meanwhile, Fridrik Skulason said that, if universally adopted, SPF will kill off the current generation of computer worms. Furthermore, he said that if SPF were combined with ISPs blocking port 25 by default, the blacklisting of ISPs that allow spammers to set up one throwaway domain after another, harsher legal actions against those using compromised machines for spam and Spamhaus-type blacklisting of spammers with their own dedicated spam servers, "you would see a *very* significant drop in spam."

Daniel Feenberg asked, "What advantage does SPF have over merely informing MAPS/Spamhaus, etc. of its dialup ranges?" Damon Sauer responded, saying that the advantage here is that the failure of SPF does not give you a 127.0.0.1 – you are in control of how to handle failures and successes. "SPF does nothing to identify spam – *only* to validate the sending domain to an IP address," he said. "This is only one piece of the puzzle, but a necessary one."

Mark C. Langston described his own reputation-based anti-spam solution, GOSSiP, which observes and rates behaviour, shares ratings with others, obtains ratings from others, and observes the behaviour of those sharing ratings. "Unlike commercial solutions," he said, "it's free (as in speech). It's also free as in beer – no money changes hands, so there's no motivation to bias results. Even if results are biased, the checks and balances built into the project allow those trying to cheat to be detected quickly and marginalized."

The solution incorporates a working Postfix policy agent, written in C, that communicates with a GOSSiP node via SSL. Mark has yet to add a working GOSSiP node, without the peer communication code. A working feedback agent forwards the SpamAssassin spam rating automatically to a GOSSiP node, also via SSL.

Mark claims to have quickly built up a database of several tens of thousands of unique identities, with a history of spam/ham behaviour, and a reputation score based on a sigmoidal function. His system also aggregates identities when SPF is advertised for the domain part of the ID, allowing for a single reputation across all senders associated with the SPF record. With GOSSiP close to its first release, Mark requested active programming contributions –more on the project can be found at http://sufficiently-advanced.net/.