MALWARE REMOVAL – BEYOND CONTENT AND CONTEXT SCANNING

Maik Morgenstern, Tom Brosch
AV-Test GmbH, Klewitzstr. 6, 39112 Magdeburg, Germany

Tel +49 391 6075460
Email {mmorgen, tbrosch}@av-test.de

ABSTRACT

Detecting threats is only one of the things anti-malware software needs to be capable of today. Removing malware, often several hundred linked registry keys and files, has become an equally important task. And this is where the trouble begins, because content and context scanning is just not enough to cope with it.

In this paper we will briefly discuss the problems of the usual approaches in removing malware as well as adware and spyware; why and where the programs fail. The programs may miss files, registry keys and values or changes made by the user to the default settings. Or even worse, they will just ignore everything but the detected EXE file, simply because no analysis has been carried out by the vendor yet, hence no dedicated removal routines are known, let alone generic removal routines. To support these points, the results of extensive testing of different technologies will be presented. And nearly all of them will face serious problems.

We will then look into other approaches which might help solve the problem. Supervising the system and bugging the user 100 times per hour is only one of the possible 'solutions'. A sandbox analysis of the malware might be another interesting way to get an idea of what the malware did and what should be removed or changed back. The paper will conclude with a comparison of the different techniques.

THE NEED FOR COMPREHENSIVE MALWARE REMOVAL

While there is consensus that malware removal is an integral part of today's anti-malware software, there are quite a few different opinions on how far those removal procedures should go. The most important point in removing malware is securing the user and his PC system. According to Jason Bruce this involves reversing some of the changes made to the system by the malware [1, p.61]. The first steps necessary are therefore to stop running malware processes and remove the linked executable malware files. In addition to this, references to the executables in the registry, such as Run keys, are often removed too.

This is already an issue to argue about. When the files are removed, the keys can't do any more harm, so they could just be left where they are. While this is certainly true, there are several reasons why registry keys should be completely cleaned anyway. Some of those reasons will also apply to files which are not executables and/or not identified by signature-based scanning, but which still belong to a certain piece of malware. There are several reasons why threats should be completely removed from a system.

Most importantly, a customer is only satisfied when all components are detected and removed properly and all modifications are reverted to their original values [1, p.61].

Secondly, some registry keys alone can be considered malicious or unwanted and should therefore be removed or changed to a default state. Examples include keys that set the Search Assistant or the *Internet Explorer* start page to malicious websites. But the same is true for keys and values which change security settings to unsafe or unwanted states. Most users don't want their task manager being disabled by malware or to have orphaned registry keys on their system.

Thirdly, there are registry keys which don't belong to the previous category but which produce annoying behaviour or will disable some functionality. These include keys that reference removed files, with the result of error messages at the start-up of *Windows*.

Also, the average user often works with more than one security product. Depending on the quality of the products used, problems with false positive issues can occur, when not all components of a threat are removed. Just imagine a product that correctly detects and disables a threat, but leaves behind a few traces in the registry and some non malicious files. A second security product might detect these traces and inform the user about an infection, which he thought had already been fixed by the first product. In the end, the user is left in a state of uncertainty, not knowing whether the first product failed or the second product is giving a false positive or both, or neither. With the increasing distribution of rogue anti-spyware products, this problem seems likely to become even more relevant, because these programs are always looking for something to detect and thus justify their existence, as Schouwenberg pointed out in 2006 [2]. Right now, most of their detections clearly are false positives, but some day they might happily detect all the stuff which is left over from real anti-malware products.

Another point to think about is that malware removal always requires some kind of analysis beforehand. This is, of course, necessary to know what the malware really does on the system but also to avoid false positives, both in detection and removal. So, when a comprehensive analysis is available, why not use that information and perform a full removal? On the other hand, a complete removal indicates a good analysis and the following question arises: when not all traces of a malware are removed, does that mean the anti-malware software vendor performed an improper analysis and just missed that stuff? And who knows what else has been missed?

In the end it all comes down to the first point again. The user expects full removal of all malware components because he wants a secure and fully working system and he wants to be sure about the state his system is in. Hence software is needed which can be trusted and which can be used as a reference when asking about the security of a system.

We'll assume everyone has an understanding of why we think malware removal should be performed to the fullest extent. On this premise, we will take a look in the following sections at the current approach, point out problems and compare it to some more generic approaches, which might help in solving those problems.

CONTENT AND CONTEXT SCANNING AND ITS LIMITATIONS

Malware removal is based on two main approaches, content

and context scanning. The first one originates from anti-virus products, while the second one used to mainly be used in anti-spyware products. Due to the increasing complexity of multi-component malware threats, a combination of both approaches is usually used to remove threats these days. The procedure is simple and only needs a few words of explanation.

Content scanning identifies malicious files with the help of signatures. This works very well on all the primary components which are the actual executable malware files, but is not the first choice when dealing with secondary components such as log or graphics files created by the malware. Additionally, registry entries will not usually be detected via signature scanning either. At this point, context scanning comes into play. When a piece of malware has been identified with content scanning, all the other related components can be detected and removed via simple context rules, which link certain files and registry keys to an identified malicious program [1, p.62].

Of course, these techniques are not free from problems, most of which have been explained in Bruce's paper. These include random filenames, rootkit and anti-removal techniques. Shared components as well as pre-infection settings are additional issues that have to be considered when removing malware and restoring a system. Most of the problems can be fixed easily; in the case of random filenames, for example, an additional content scan can be performed, anti-rootkit techniques can be used to remove rootkits and so on.

The most severe problem, however, is response time. Introduced by Andreas Marx at the 2004 Virus Bulletin Conference [3] the term describes how long it takes for anti-malware vendors to respond to a new threat. Usually it refers to the time frame between the occurrence of a threat and the release of a signature update which detects that threat. In order to create a signature for detection it is only necessary to decide whether a file is malicious or not, which in most cases shouldn't be too hard. Yet response times vary from only one to two hours, to up to several days for different vendors and different threats.

To take this a step further we checked when the dropped files of a sample are detected by an anti-virus product. This could be an indicator of how long it takes until a (final) removal routine becomes available. To get a better insight into the time frame we are talking about, take a look at Figure 1. Corresponding results and conclusions will be presented in the next section.

Before presenting our test results yet another point must be made. The behaviour of modern malware can be affected by many different factors. For example, the origin of the system used for analysis, the date, or the presence of any installed programs. Additionally, when malware loads files from the Internet, these files may change over time, therefore requiring a new analysis. Even when an initial analysis and removal routine is available, there is no guarantee that it is complete and up to date; in this case Figure 1 would become a lot more complicated and the time frame in which the user is unsafe, or at least unsure, might become a lot bigger.

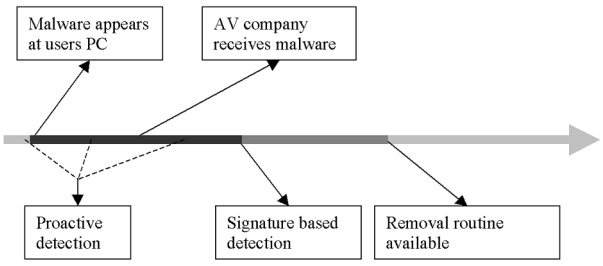

Figure 1: Time frames between detection and removal routines.

Two main issues can be concluded. Analysis of the malware is always necessary to provide detection as well as removal capabilities. Some malware changes its behaviour, so updates to removal routines become necessary as well. Both of these points lead to one big problem: it can take way too long until a satisfying removal routine is available to the user.

Test results

In order to support the assumptions made above, some actual test results will be discussed in this section. First, we will look at the results we've just announced above: statements on the time frame between the detection of a sample and the detection of the dropped files. The second type will include some actual removal results gained from desktop products. Several conclusions will be drawn, that lead to the next sections which will examine different approaches and how they might help close the gap between the occurrence of a new piece of malware and the successful removal of it from a system.

Removal of malware is only necessary when no protection for it was in place before the malware hit the customer's system. Hence, a look at outbreak response times and proactive detection is needed to gain an overview of how relevant the problem of malware removal actually is.

Table 1 presents the average response times of the given products over the last six months. Proactive detection rates are also given. These refer to different test sets and to different ages of the signature updates, ranging from one to three months old. The test sets used here contain worms and backdoors taken from our zoo collection. Although the proactive detection results have been gained in several different tests using different test sets, they still illustrate the general problem.

What can clearly be seen is that all vendors take some time to react to a new threat. It is not possible to determine when a piece of malware first appeared 'in the wild', hence it is very likely that the real response times are even higher. This is also supported by the fact that no vendor can have a zero-second access to a new threat. It has to appear somewhere, before vendors can take care of it. Finally, having an update ready doesn't mean that users will download and install it right away. In the end, there will be a time frame of many hours, sometimes even days, in which a new malware threat can't be detected with a dedicated signature.

At this point, proactive or generic detection would be a solution. However, as the detection rates given in Table 1 suggest, most products wouldn't even catch half of the new malware proactively. Furthermore, proactive detection isn't always available from the start to protect a user's system. Instead it is constantly updated by the vendor, because new malware techniques and families are discovered. Therefore it might be able to detect a malware proactively sooner or later.

Vendor/Product	Average response time range, incl. proactive detections	Proactive detection (based on different tests)
Avira AntiVir	2 to 4 hours	20 to 50%
Alwil Avast	6 to 8 hours	5 to 35%
Grisoft AVG	6 to 8 hours	5 to 35%
BitDefender	2 to 4 hours	25 to 60%
F-Secure	< 2 hours	20 to 50%
Kaspersky	< 2 hours	20 to 50%
McAfee	14 to 16 hours	25 to 45%
Microsoft	38 to 40 hours	5 to 15%
Eset Nod32	4 to 6 hours	30 to 70%
Panda	4 to 6 hours	20 to 50%
Symantec Norton	6 to 8 hours	15 to 50%
Trend Micro	6 to 8 hours	15 to 45%

Table 1: Response times and proactive detection rates, based on tests by AV-Test.org, published in [8, 9].

There is actually a chance that a customer's system will get infected, because on the one hand it takes time to create and distribute signatures and on the other hand proactive detection is not very often able to stop new unknown threats. Then again, this is something which shouldn't be a big surprise when looking at real-life scenarios.

Panda Software, for example, publishes some interesting statistics on its website www.infectedornot.com. About half of the PCs scanned with the company's online scanning service were found to be infected, with not much difference whether anti-virus software was installed or not. The site also provides statistics on the distribution of different malware types found. Adware, hacking tools and trojans are the top three with each accounting for more than 25%.

Clearly, disinfection and malware removal routines are necessary. But how well do they work and how quickly are they available? Two different tests have been carried out to answer these questions. The first one determines the time frame between the detection of the original malware via signature and the detection of the dropped components with additional signatures. The second test presents actual removal rates of different types of malware. These results can be used to conclude how well anti-virus products are able to clean a system after it has been infected by malware.

Figures 2 and 3 are two small extracts of figures showing the response times for dropped components compared to the detection date of the original dropping component. Figure 2 refers to samples which were added to the WildList in February 2007 while Figure 3 refers to adware and spyware samples that appeared throughout 2006 and at the beginning of 2007.

The columns refer to different products used in the test, which are the same as those listed above, but in no specific order. The rows refer to the dropped files that have been considered. Please note, the detection of dropped components differs between different products, hence neighbouring cells don't necessarily refer to the same dropped component. Again, it is not the results of single products that are of interest, but rather the overall picture that should be looked at.

Three different kinds of grey tone have been used. The lightest one refers to dropped samples that have been detected the same day or even before the dropping sample has been detected. The middle tone refers to samples that have been detected in a time frame of one to seven days after the dropping sample has been detected. Finally, the darkest tone shows all the dropped files which have only been detected after a week or even longer.

The WildList table looks a lot like it should. Most products captured all the dropped files on the same day the original samples were detected, or even before. Only on very rare occasions did it take some vendors more than a day to add detection, usually just one to seven days. The same result has been seen in more tests using the additions to the January 2007 and March 2007 WildLists as test samples.

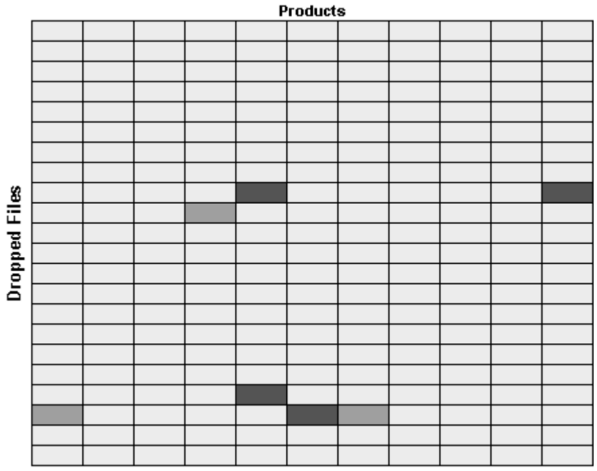

Figure 2: Example detections of dropped files, WildList 02/2007 samples.

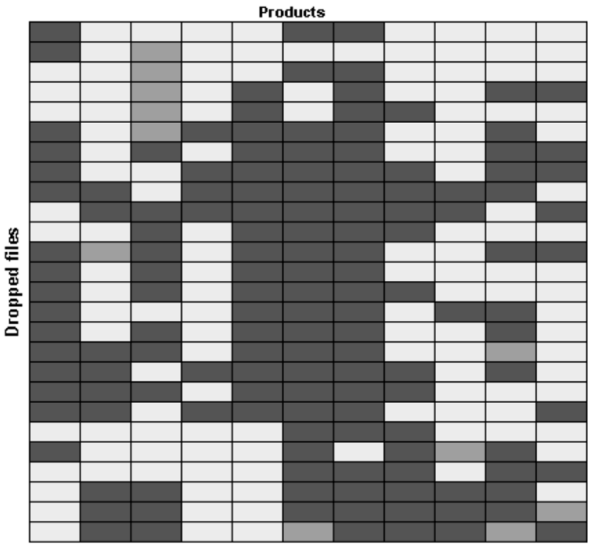

Figure 3: Example detections of dropped files, adware and spyware samples.

Dropped component detected the same day or before the dropping file.

Dropped component detected one to seven days after the dropping file.

Dropped component detected more than seven days after the dropping file.

However, when looking at the adware and spyware results the products' performance was a lot worse. Every product had trouble to some extent in detecting dropped components in a timely manner. Actually, most of the initially missed components were detected at least seven days after the detection of the original dropping sample. There is no easy conclusion to draw from these results. The products obviously do fine on WildList malware in these tests. But why is that so?

One possible explanation is the sheer volume of malware these days. There are so many samples to process, that different priorities are necessary. WildList samples will get the highest priority because this is malware that is known to be a real and dangerous threat. Also, since most of the products reviewed are still mainly anti-virus software with added anti-spyware capabilities, they focus on their original main purpose of removing viruses, worms, etc. rather than adware and spyware.

Another explanation of the results could be due to the complexity of the different threats. Conventional malware, as found on the WildList, usually performs fewer system changes than adware and spyware which often create several hundred registry keys and files. Hence it is easier to provide a full analysis of WildList malware with regard to system changes. There are of course exceptions, such as malware which uses rootkits or other similar techniques which complicate the analysis.

Finally, since adware and spyware often consist of many files which are not malicious *per se*, context scanning may be used instead of content scanning to identify and remove these. So the dropped files don't necessarily have to be identified with a signature. However, we only considered files that were detected by the software sooner or later, which conflicts with this assumption.

Some actual removal results of tests with desktop products might shed some more light on the whole story. Different WildList malware as well as adware and spyware was used to infect a clean *Windows XP SP2* system. The changes made by the malware were recorded and the state of the infected system was saved, so it could always be restored for testing again. This was repeated for all the malware samples in the test. In the next step, the anti-malware products were used to disinfect the system of the respective malware and the changes made to the system by the anti-malware software were noted. The products used in these tests include those mentioned in the tables above. The results were not really surprising considering what we have written above.

System disinfection of WildList malware worked relatively well. The removal of malicious files proved no problem for any of the tested products. However, non malicious files such as log or configuration files were often left on the hard disk. Additionally, changes to the hosts file were either ignored or were only partly fixed by most of the products. The picture doesn't look too bright where the registry is concerned, either. Despite the fact that all products were able to deal with the registry in some way, a lot of simple mistakes were made. Even Run and RunServices keys were overlooked by several products. In the end, no product was able to perform a 100% system disinfection of WildList malware. But it must be noted that all products successfully disabled nearly all malware samples in this test, with only very few exceptions, where the malware stopped the scan process or the scanner as a whole.

When looking at adware and spyware the results were a lot worse. The tested products can be divided into two classes: those that try to remove adware and spyware and those that don't seem to put much effort into that task. Let's start with the latter. While these products try to deal with all infections they detect, they only remove a very small fraction of the total number of changes. For example there were several products that were not able to remove more than 25% of the created files. If you think this sounds bad, just wait for the registry removal results. Out of 3,500 registry keys created by 50 different adware and spyware samples, several products couldn't remove more than 60 keys and some didn't even remove 10. As an example, refer to Table 2 below. The adware shown here is 'Hotbar', which creates quite a lot of files and registry keys on a system. Removal results of four different products have been included. Names of the products are not given, since this example doesn't necessarily reflect their overall performance.

	Files created	**Registry keys created**
AdWare.Hotbar	183	789
	Files removed	**Registry keys removed**
Product A	16	0
Product B	25	0
Product C	26	43
Product D	182	778

Table 2: Removal results of 'AdWare.Hotbar'.

The first three products don't give the impression they were all too serious about removing the threat. However, as you can see, there are products that are able to perform a (nearly) complete removal of such threats. This brings us to the other type of products which are actually doing pretty well at removing adware and spyware. They reach about 60–80% in file removal and about the same rates in removing registry keys on average over all samples. Another conclusion is that those products either do a very good job on adware and spyware, or they nearly completely fail to remove it. That way, an important difference from WildList malware removal has to be noted. While all products could disable nearly all of the WildList malware threats, this is certainly not the case for adware and spyware. The main installation files and some components may be removed, but many other components remain which stay active on the system. Only when nearly all components of the malware were removed was it truly disabled and no longer able to do any more harm.

The results of this section seem to support the findings of the earlier sections. WildList malware was again handled very well by all products. These samples are clearly given priority. However, some shortcomings were already obvious in the case of registry cleaning. Adware and spyware removal showed very different behaviour for the different products. There were some that failed on nearly all samples while on the other hand there were products that did pretty well on nearly all samples. However, no product reached as good a rate as in the case of WildList malware removal. This may again point to the issue of higher complexity in the case of adware and spyware. It also seems that some samples were not yet fully processed at the time of our tests and only detection for the original sample and maybe some of the

related files was in place. Even the better products have a hard time keeping track of all the new malware sometimes.

GENERIC APPROACHES NEEDED

As described in the previous section, vendors sometimes react pretty quickly to new threats and sometimes they take many hours or even days. But often just the original samples and maybe a few dropped files are detected. A full disinfection routine that can clean all those files and registry keys may still be missing. Only after a couple of days, when the malware has been completely analysed, can the user take advantage of the better disinfection routine. But even then not all malware components can always be removed, for several reasons as described above. To close the gap from detection to complete removal of the malware, generic disinfection routines which work directly on the infected system could be a solution.

When looking back through anti-virus history, there has been a similar situation before, not with system disinfection but with file disinfection. Since the number of malware outbreaks rose very quickly, it was ceased to be possible to create file disinfection routines for every single item of malware. Instead of analysing the whole infection routine, generic file disinfection strategies were developed [4, p.474]. In most cases the virus reverts the infected file to a clean state after it has started, so generic approaches emulated the execution of the infected file and let the virus do the disinfection work. Applying this technique to system disinfection is not really possible, since the malware won't revert its changes to the system. But it's possible to monitor it and learn what happens during the infection phase. The idea examined here is to emulate the system infection process in a sandbox and report all system changes the malware makes and then revert all of them to clean the whole system.

Sandbox-based disinfection

Before we continue it is necessary to get some insight into exactly what we mean by sandboxes. We all know sandboxes from our childhood. They were a place where we could do what we wanted without damaging anything, just play around. A computer sandbox is exactly the same. It is a separated area of the computer where applications can run without changing anything on the original system. Or more precisely: a sandbox is a virtual environment where executable files can be test-driven to analyse their behaviour.

The malware runs in a virtual subsystem of the actual operating system. In this cage it has read access to all files and even the registry. Also the storage device, the IO manager, the ROM, the RAM and the user are emulated [4, p.490; 5] When the sample attempts to modify files or registry keys, virtual copies are created to which the malware has read-write access. Furthermore it is possible to detect whether the malware wants to re-flash the BIOS or tries to hide information. After the program terminates a report is produced of the activity of the malware, which could then be used for disinfection.

So the simple disinfection routine would follow these steps:

1. During the first step a signature-based full system scan would be performed. All files found in this step would be added to a list of malicious files. In this step only some files (and maybe registry keys) would be found where signatures were available.
2. In the next step every file from the list of malicious files would be analysed within a sandbox. During this step you would get even more files and registry keys that were created during the system infection.
3. Next, all active components of the malware would be deactivated. This includes terminating malicious processes as well as unloading drivers that come with the malware.
4. In the last step all found files and registry keys would be deleted (if necessary after a reboot). Infected files would be disinfected using generic disinfection routines and system changes made by the malware would be reverted to a default state.

Test results and further concepts

In our test we analysed each malware sample twice. In the first step we ran the file on a real system to find out which system changes were made by the malware. In the second step we used a sandbox analysis and then we compared both results.

During our test we saw that with the simple sandbox approach only a few files and registry keys were found. The spyware 'Admedia', for example, creates 48 files and 178 registry keys and values, but only 24 files and six keys were found by the sandbox. One reason for this poor performance is that one sample does not perform a complete infection process. During the infection other files were dropped and executed to complete the system infection, so our infection analysis has to cope with that.

An improvement on the simple approach is to analyse all dropped components inside the sandbox as well. Additional files that were found during this step were added to the list of malicious files. The analysis process stops when an analysis has been performed for all files on this list. This is followed with steps three and four.

Better results can be achieved with this approach because files and registry keys can be found that were created during a multi-stage infection process. In the case of 'Admedia' the detection of files was raised from 24 to 32 and four more registry entries were found too. But these results are not sufficient anyway. The reasons for this will be discussed in the next section.

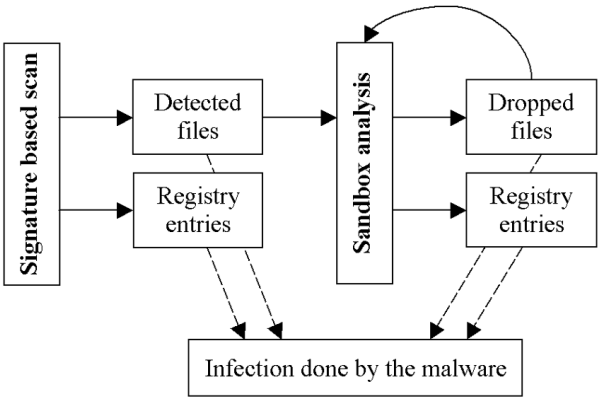

Figure 4: Sandbox-based information retrieval.

Limitations of the approach

There are two classes of problem that arise when performing automated sandbox disinfection. On the one hand there are problems which are closely related to the sandbox so these are the same for all malware samples. Other problems relate to certain malware samples only.

Some sandboxes only emulate the well-documented *Windows* API that can be seen as an abstraction of the native API and is not intended to be called in user applications. But virus writers might use the native API calls to confuse the sandbox. So if the sandbox is not able to emulate this API, the sandbox will not be able to emulate the infection process correctly.

Additionally, malware could try to detect virtual environments and behave differently from how they would on a real system. Therefore they could test unimplemented APIs or services or use other tricks like the Red Pill test presented by Joanna Rutkowska [6].

The second class of problems consists of problems that are related to certain malware so they cannot be solved in general. Some spyware installers require user action to complete the installation process, but in a sandbox only random user actions can be simulated so far. This makes it hard to analyse such installers automatically. Files downloaded from the Internet during the installation process are a problem too. Fileservers can be down or send different files from time to time so you cannot guarantee that you get the same files while analysing the malware as during the initial infection. Sometimes infections are scheduled and not performed right away when executing the sample. These scheduled tasks have to be detected by the sandbox and triggered automatically.

Additionally, it is possible that the malware checks whether the system is already infected or not before doing anything. As described before, in a sandbox the malware has a read-only view of the whole system so it can detect that no infection is needed and can terminate without doing anything. To solve this problem you could emulate a default operating system but then infections that relate to user-specific applications cannot be found. So you would have to emulate the system before the infection. But this is what we actually want to get as a result of disinfection and is therefore impossible to do.

Random filenames are also a problem when no signatures for these files are available, because these files will be named differently during the analysis than they will during the infection. In addition, pre-infection settings are hard to recover since the sandbox cannot determine the pre-infection state. For example, security settings of the web browser or other settings can only be reverted to if they were saved before the infection. Otherwise you have to revert them to the default values.

Not cleaning the infected system correctly is one problem but it is even worse when the disinfection routine triggers an infection process on a clean system. This can be the case when the malware breaks out of the sandbox during the emulation. It is hard to guarantee that there are no security holes in the sandbox. This way malware could exploit the sandbox and break out. More descriptions of such problems can be found in 'Insecurity in Security Software' [7].

Other approaches

Clearly sandbox-based disinfection has a lot of weak points as the test results and further limitations showed. It is pretty clear that at the moment a sandbox-based disinfection can't replace really good removal routines based on a proper analysis performed by the vendor. However, they are a valuable contribution to disinfection techniques and help with closing some gaps which have been pointed out above. The same is true for other approaches, which can assist either current approaches or the sandbox-based approach. We will look briefly into one that supports the latter.

This will basically be some kind of system supervising, which logs all the system changes caused by certain programs and also stores the old settings. This information can then be used to restore a clean system state, when that program is known to be malicious, if for example signature-based detection is in place. Similar functionality already exists in current security software. However, the software either completely locks down the system and forbids certain system changes or asks the user what to do, sometimes bugging him with message after message when installing legitimate software. Eventually, the user gets used to clicking 'Yes' and allowing all the system changes the software asks for, maybe even when it's malware. In contrast, the approach discussed here will log changes, but neither ask the user nor prevent any of these changes. Only when it is certain that the program which performed these changes is malicious, will this information be used for the removal process.

There are of course several problems that have to be dealt with here. The first question is which programs should be supervised? Monitoring each and every piece of software on the system won't work out well for performance reasons, hence some kind of security risk rating needs to be applied. Most of today's anti-malware products are able to rate certain files as suspicious based on heuristics. The same should be applied here, but in a very loose way to make sure nearly all potentially malicious files are caught. On the other hand it is also helpful to have a whitelist containing known clean programs. Additionally, when a supervised program is still not identified by a signature after a few months, a 'time-out' should trigger and that file can then be removed from the monitoring list.

So far it is known which files should be supervised initially, but it is unclear which of the system changes made by these files can be considered malicious. It is quite possible that some programs perform both wanted and unwanted changes to the system and create both clean and malicious components. The step of removing the malware and reverting the system changes could be quite challenging because of this.

With these limitations it is hardly imaginable that this approach could be used for malware removal without using any additional methods. But it can solve a few of the problems which occur with sandbox-based attempts. We talked about pre-infection settings which couldn't be restored either with content and context scanning or with sandbox-based disinfection. This could now easily be handled with the information gained in supervising the malware, since the old settings would have been saved.

Another problem relates to files with random names not detected by a signature. These wouldn't be handled correctly with a sandbox analysis, since the names would most likely be different in reality. With the supervision approach, this wouldn't be a problem, because all the actual changes and created files would have been recorded. The same is true for

files or registry keys that didn't show up in a sandbox analysis, but were created anyhow.

CONCLUSION

It is not hard to rate the current situation with content and context scanning. It cannot be denied that there are a lot of weak points, both in detecting and removing malware. PC systems still get infected a lot because it takes too long until signature updates reach the users and proactive detection is still very variable. Therefore, the problem of malware removal arises. While for some malware categories, many vendors provide removal routines pretty quickly, there are other categories which have to wait a lot longer. Additionally, no generic routines are in place for proactively detected malware. So, even when a piece of malware is detected on a system, there is no guarantee that all related components can or will be removed.

Generic routines that work directly on the user's system would be a solution to that problem. We examined an approach that works on a sandbox-based analysis. While in theory this sounds like a very good idea, real tests proved to be disappointing. The naïve procedure, which only considered one layer of sandbox results, didn't yield any noteworthy results. Only when more layers were reviewed could some meaningful information be gained for removal of the malware. Compared to the current procedures this approach cannot act as a replacement, but rather as a complementary procedure.

Similar statements can be made about the supervision concept. While it is not able to solve all problems on its own, it can be a good addition to other methods. When used in conjunction with the sandbox-based approach some of the issues described above can easily be solved.

And this is the main thought that can be taken from this paper. A combination of several approaches, each solving issues of the others, is probably the best way to go. It remains the case that nothing can beat a proper analysis performed by anti-malware vendors. However, it takes time until they are available and they might become outdated when malware changes its behaviour. Generic approaches such as sandbox-based methods can solve these two problems. But the problem of randomly named files and pre-infection settings is still open. This is where the supervision method can help and take care of these issues.

REFERENCES

[1] Bruce, J. The challenge of detecting and removing installed threats. Proceedings of the 16th Virus Bulletin International Conference. 2006.

[2] Schouwenberg, R. The (correct) detection of light grey software. Proceedings of the 16th Virus Bulletin International Conference. 2006.

[3] Marx, A. Anti-virus outbreak response testing and impact. Proceedings of the 14th Virus Bulletin International Conference. 2004.

[4] Ször, P. The Art of Computer Virus Research and Defense. Addison Wesley. 2005.

[5] Natvig, K. Sandbox technology inside AV scanners. Proceedings of the 11th Virus Bulletin International Conference. 2001.

[6] Rutkowska, J. Red Pill... or how to detect VMM using (almost) one CPU instruction. http://invisiblethings.org/papers/redpill.html. 2004.

[7] Morgenstern, M.; Marx, A. Insecurity in security software. Proceedings of the 15th Virus Bulletin International Conference. 2005.

[8] Knop, D. Die Leibwächter - 17 Virenscanner für Windows XP und Vista. c't 5/2007, pp.142–153.

[9] Larkin, E. Top antivirus performers. PC World, April 2007. http://www.pcworld.com/article/id,130869/article.html.