The InMAS Approach

Markus Engelberth Felix C. Freiling Jan Göbel*
Christian Gorecki Thorsten Holz† Ralf Hund
Philipp Trinius* Carsten Willems

Laboratory for Dependable Distributed Systems
University of Mannheim

Abstract

The *Internet Malware Analysis System* (InMAS) is a modular platform for distributed, large-scale monitoring of malware on the Internet. InMAS integrates diverse tools for malware collection (using honeypots) and malware analysis (mainly using dynamic analysis). All collected information is aggregated and accessible through an intuitive and easy-to-use web interface. In this paper, we provide an overview of the structure of InMAS and the various tools it integrates. We also introduce the web frontend that displays all information on different levels of abstraction, from a coarse-grained overview down to highly detailed information on demand.

1 Introduction and Motivation

The Internet has become a vital part of today's critical infrastructures, not only in Germany. A failure, even if it is only a partial failure, of the Internet can have dramatic consequences on economy and society [4]. Security incidents like the spread of Code Read in 2001 or Conficker in 2009 have shown that malicious software (*malware*) can have a significant influence on today's Internet. It is therefore important to anticipate such incidents, detect them once they occur, and finally react in a robust and effective way. Anticipation and detection are the main benefits of an *early-warning system*.

An early-warning system should satisfy two properties: On the one hand, it should give an overview over the current state of the network similar to a global view of the system. On the other hand, it should detect anomalous network behavior and classify its cause as exactly as possible. Especially interesting from our point of view are network anomalies caused by distributed and wide-spread attacks onto the network infrastructure, which are today mostly caused by malware. Nowadays, malware is such a severe threat because it has the ability to

*contact authors, e-mail: {goebel,trinius}@informatik.uni-mannheim.de
†Thorsten Holz is now affiliated with the Secure Systems Lab, Technical University Vienna.

spread autonomously by infecting vulnerable computers on the Internet. Examples include worms, Trojan horses, and bots. An early-warning system must be able to give an overview over the current zoo of spreading malware together with the level of danger associated with them.

In this paper, we provide an overview of the *Internet Malware Analysis System* (InMAS), a modular platform for distributed, large-scale monitoring of malware on the Internet. Historically, InMAS evolved from two tools that still make up the core of the system: (1) the honeypot tool *Nepenthes* [1], and (2) the dynamic malware analysis tool *CWSandbox* [20]. To cover other propagation vectors of modern malware (such as email and drive-by download attacks) and to support the analysis of collected binaries, the system has been extended by new sensor types, new analysis techniques, and new visualization approaches, on which we report in this paper.

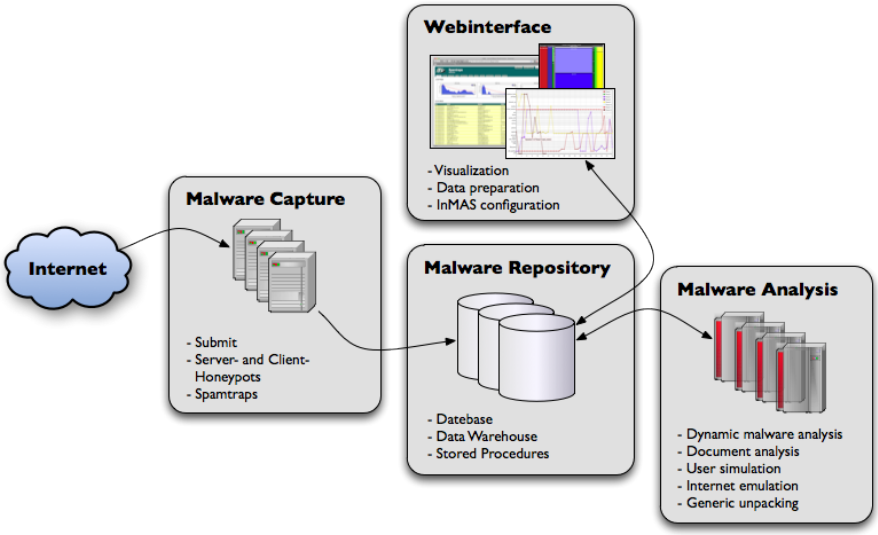


Figure 1: Schematic overview of InMAS.

In a nutshell, InMAS is centered around a single database (*Malware Repository*) that stores all of the collected and analyzed data (see Figure 1). The Malware Repository is origin and destination for all InMAS activities, i.e., all analysis tools query this database to either retrieve or store information about malware binaries. Through this concept of sharing data, all sensors, analyzers, and visualization tools are decoupled, increasing the modularity of the system.

In addition to the already mentioned sensors and analyzers, another building block of InMAS is a web interface for administrators and analysts. The web interface shows the results of all performed analyses and allows administrators to configure InMAS, e.g., by adding new sensors to the system or improving the classification rules of some analysis tools.

While the basic structure of InMAS has been described elsewhere [6], this paper focusses on two particular aspects of InMAS that have not been presented earlier: First, we give an overview of the structure of InMAS and the diverse capture (Section 2) and analysis (Section 3) tools it integrates. Second, we introduce the philosophy behind the web frontend and provide examples of the look-and-feel of InMAS to the user (Section 4). We conclude in Section 5.

2 Malware Capture in InMAS

In order to cover various propagation vectors of malware, InMAS has the capability to support different types of sensors. In this section, we present four sensor types for detecting and collecting malicious software. Due to space reasons, we focus on technical aspects of the sensors. Note that InMAS also supports several techniques for data privacy protection such as pseudonymization and anonymization of the collected data.

2.1 Low-interaction Server-Honeypots

Server honeypots are honeypots that provide services to hosts on the Internet, i.e., this type of honeypot passively waits for attackers to connect and exploit the offered services. According to the interaction level of the honeypot, these service are either real or emulated. InMAS supports both low-interaction and high-interaction honeypots (we focus on the latter in the next section).

The current setup of InMAS supports the two low-interaction honeypots Nepenthes [1, 19] and Amun [8], that both emulate common vulnerabilities of *Windows* and *Unix* services. As soon as a client connects to one of the honeypots, all actions are recorded and sent to the InMAS database server. Thus, all information beginning with the first connection to the final exploit is stored at a central location for analysis. Beside the detection of attacks, the honeypots also try to interpret incoming shellcode and extract information about download locations of malware. Downloaded files are sent to malware analysis system of InMAS for dynamic, behavior-based analysis.

2.2 High-interaction Server Honeypots

Low-interaction honeypots, as the ones mentioned above, are only capable of detecting attacks using known vulnerabilities. Additionally, the known vulnerabilities need to be implemented in the honeypot before any detection can take place. As a result, the detection of zero-day exploits is hard, as it requires the current service emulation to support this new exploit and the shellcode must be known as well. The detection of attacks using unknown vulnerabilities is impossible. Although current low-interaction honeypot solutions are modular, the implementation of new vulnerabilities modules and the detection of new shelllcode needs to be done manually.

To circumvent this drawback InMAS supports the high-interaction honeypot HoneyBow [22]. Instead of service emulation, HoneyBow uses a real system with real services. In order to detect attacks against these services, some special monitoring software is installed as well. Therefore, HoneyBow can detect zero-day exploits and attacks against unknown vulnerabilities. Again, the collected files are sent to the malware analysis component of InMAS.

2.3 Spamtraps

Another common attack vector is email spam. With the increasing security mechanisms in networks, malware has shifted to other attack methods than only scanning for vulnerable systems. Almost all of the major botnets, for example *Storm Worm* [16, 12] or *Waledac* [17], use email spam for propagation. In order to detect these kinds of attacks, InMAS allows the integration of *spamtraps*. Spamtraps are email accounts without a productive purpose, i.e., no regular email traffic is expected. Therefore, any received email on one of these accounts is considered as spam. Besides single email accounts, it is also possible to register domains and collect all incoming emails targeting this domain.

Each incoming email is automatically processed by InMAS. Each embedded URL or attachment is extracted and stored separately. Executable attachments are passed to the Malware Repository to generate an analysis report, and URLs are passed to client-honeypots for further investigations. Spamtraps are on the one hand sensors to measure the amount of spam in a network and hence, a good indicator for propagation campaigns of botnets. On the other hand, they serve as an input vector to client-honeypots, which we describe in the next section.

2.4 Client Honeypots

Client honeypots are used to study attacks against client applications, such as web browsers and document readers. The above mentioned spamtraps form the major input vector to the client honeypots used within InMAS. In contrast to server honeypots, client honeypots do not passively wait for attackers to come by, but actively crawl the Internet for malicious content.

The current setup of InMAS supports the integration of *CaptureHPC* [10], a high-interaction client honeypot. CaptureHPC is used for the detection of attacks against a web browser. For this purpose, URLs found in email spam are passed to the server instance of CaptureHPC, which distributes them to individual virtualized clients running for example Internet Explorer to visit the websites. Special monitoring software, called CaptureClient, on each of the clients monitors any changes to the system. With the help of an exclusion list, i.e., rules of allowed changes, CaptureHPC determines if a website is to be considered malicious or not. The results are stored in the InMAS database. As attacks against client applications, especially web browsers, have increased in the recent years, the integration of client honeypots as sensors to InMAS was a major step towards getting a better overview of current attacks on the Internet.

3 Malware Analysis in InMAS

An early-warning system has to deal with a huge number of collected malware samples every day. Therefore, it is mandatory that InMAS uses tools and techniques to analyze new malware samples with no (or only limited) user interaction. Nearly all analyses within InMAS are fully automated. By combining both *static* and *dynamic* analysis techniques, we try to extract as much information about collected malware samples as possible.

Beside antivirus scanners and CWSandbox, we integrate several custom-built tools, like *MalOffice*, *TrumanBox*, and *SimUser*. These analysis tools and the corresponding interfaces are discussed in detail in the following sections.

3.1 Dynamic Analyses using CWSandbox

Dynamic analysis of malware either aims at the *code* or at the *behavior* of the particular sample. For the latter approach several tools have been developed in recent years [2, 15, 20]. Those execute a malware sample in a controlled environment and record all system-level changes such as modifications of the filesystem or the registry. As a result, an analysis report is created, which provides detailed information about all observed activities performed by the analyzed sample.

InMAS utilizes CWSandbox [20] for dynamic, behavior-based analysis. All collected samples are executed within the sandbox system for around 2.5 minutes. Most of the malware-related analysis techniques are based on the XML encoded behavior report generated by CWSandbox.

3.2 MalOffice

Malicious documents, i.e., documents that contain malicious payload (for example a keylogger) became a serious threat recently. This attack vector is often used in targeted attacks against governments, military, and similar high profile targets [14]. *MalOffice* is a combined approach of static and dynamic analysis to detect malicious data files, e.g., Word or PDF documents that exploit a vulnerability in the according application [7]. The scanning technique allows us to find obvious malicious artifacts in files, such as embedded PE32 files or known exploit code. In the dynamic approach, the particular document is loaded in its associated application, which in turn is monitored for *suspicious behavior*. Since some exploits only trigger on particular application versions, we use many instances running different versions of these host applications in parallel.

The monitoring is carried out in several CWSandbox systems that allow us to observe suspicious actions like the creation of files, spawning of processes or outgoing network connections that happen when opening a document. For every document there are several virtual machines containing exactly one version of a certain application. The resulting analysis reports are checked against predefined policies in order to reach a verdict.

The overall result for a document is calculated from several single analysis outcomes. A numeric value is assigned to every single outcome: *probably benign* (0), *suspicious* (0.5), and *malicious* (1). The overall result is just the average value of the single results. If a single result states that a document is malicious, the overall result is malicious (1) as well.

3.3 SimUser

Some malware samples trigger only under certain circumstances, which depend on user behavior. The typical example are keyloggers monitoring keystrokes and sending the harvested information to an attacker periodically [11]. Executing a keylogger binary in a sandbox environment will not reveal much malicious behavior, as there will not be any of the mentioned activity in the absence of user activity. In order to study this kind of malware, we need some sort of *user simulation* to trigger needed events.

We have developed a tool called *SimUser* to simulate the behavior of a victim. The core of SimUser is based on *AutoIt*, a scripting language designed for automating the Windows GUI and general scripting [3]. We use AutoIt to simulate arbitrary user behavior and implemented SimUser as a frontend to enable efficient generation of user profiles. SimUser itself uses the concept of *user behavior templates* that encapsulate a concise user task, e.g., opening a website and entering a username/password combination in the form fields to log in. These templates can be combined in an arbitrary way to generate a profile that simulates user behavior according to specific needs.

The resulting user profiles can be used within the dynamic analysis environment to simulate the behavior of a user and to trigger conditional malicious activity by that. Therefore, CWSandbox can be instrumented to execute a SimUser script while it is analyzing a malware sample.

3.4 TrumanBox

One of the most important requirements during dynamic malware analysis is connectivity to the Internet, since most of the malicious software is meant to send data to other systems or receive commands, for instance by contacting an attacker-controlled server. Hence, providing full Internet access during the dynamic malware analysis process is important in order to obtain good and extensive reports. However, there are scenarios where it is not acceptable to grant Internet access to a malicious sample, e.g., when the sample tries to send out loads of spam or aims to spread itself to other machines. Restricting connectivity to the Internet during malware analysis often causes results of lower quality as many functions are not triggered and hence can not be reported.

As a solution, we have developed *TrumanBox* [9], a system that can be deployed as a router or bridge in an existing network infrastructure without any need for reconfiguration. Using TrumanBox, we cannot only restrict access to the Internet, but also transparently redirect connections to the TrumanBox itself. The redirected connections are handled by generic services providing

functions and an environment similar to the service expected by the malware. This way, we obtain extensive results without affecting third-party systems. Furthermore, we can even analyze malware trying to connect to a server that is already offline, since connections are transparently handled by TrumanBox instead of timing out with a "host unreachable" error.

During our tests we have proved this expectation. Using TrumanBox, we have been able to trigger activities of malware samples which tried to contact servers that have been taken offline a long time ago. These activities could not have been reported with a regular Internet connection. Furthermore, we can utilize TrumanBox to collect all spam emails during the analysis of spambots. Thus, no spam is sent to third party users, but we can nevertheles collect mails.

3.5 Generic Unpacking

Modern malware typically employs advanced obfuscation techniques to avoid detection by intrusion detection systems and antivirus engines. Additionally, obfuscation also serves the purpose of hampering human analysis of the binary. In order to neutralize signature-based detections, many nefarious programs vary their obfuscation scheme by, for example, randomly changing the encryption key. Signature-based detection is thereby rendered impossible as only a very tiny stub of the program remains constant among all versions. Such behavior has been frequently seen in the wild [16].

To apply obfuscations, a so-called *packer* takes the readily-compiled malware, transforms the image code and data, and then prepends a decoding stub to the executable file. When the program is executed on a victim's computer, the stub decrypts the actual malware in memory at first and then jumps to the original entry point.

To account for the rising occurrence of obfuscated malware, we have extended the deobfuscations capabilities of InMAS and have also integrated an out-of-the-box generic unpacker into the system. In this context, a diverse set of generic unpackers has been presented by the academic community in the recent past [13, 21]. After having evaluated all of the publicly available solutions in this field, we chose *Ether* [5], because it is powerful, publicly available, well documented, and can be easily automated. However, as with every other generic unpacker mentioned above, Ether suffers from a severe lack of performance, which means that the unpacking of an exemplary sample takes several minutes on average. For this reason, we cannot run Ether for every submitted binary; unpacking must be explicitly requested in the webinterface.

To bridge this gap, we have also implemented basic unpacking features into CWSandbox. Whenever a sample terminates, our component dumps the memory contents to disk and tries to rebuild a valid executable file. The dumper also reconstructs the *import address table* (IAT), i.e., the references of the program to external libraries in the system. This information is typically also obfuscated by the packer and reconstructing the imports is hence a mandatory step for reverse engineering. By looking at the imports, an analyst can watch the in-

teraction of a program with its environment and thus gain in-depth knowledge about the sample's behavior.

The dumper component of CWSandbox does not impact performance and can be used for each analyzed binary. The output is an unpacked executable that is enabled to further human analysis. The drawbacks of this solution in comparison to Ether are, that we cannot determine the original entry point of the unpacked program and that the data sections have already been modified due to the execution of the program.

4 InMAS Backend

Since potentially every Nepenthes and Amun installation can send the collected samples to our system, we effectively have an InMAS sensor system that is distributed all across the world. We monitor attacks on our honeypot systems and spamtraps and log a lot of attack-related data, like the IP address of an attacker, the port number of the attacked service, or even the SMTP server used to send spam. In addition we gather the malware which is involved or distributed by these attacks. Furthermore, we receive a considerable amount of malware by our online analysis service *cwsandbox.org*, which is also a part of InMAS. This global mesh of sensors along with the wide variety of analysis and appropriate aggregation of attack and malware data allows the derivation of a substantiated status report. The statistic analysis of this data helps to detect changes in service usage and new vulnerabilities in services of client side applications, e.g., web browsers or office applications, and to detect network anomalies. This data is very important for the operator of an early-warning system, security professionals, and vendors using this real world data to generate signatures for antivirus scanners or intrusion detection systems. We now introduce the web interface, which is the central interface for users of the system.

4.1 Webinterface

Representing data within the context of an early-warning system it is important to provide an abstract overview of relevant events supporting easy and immediate recognition of anomalies, without restricting professionals in further investigations. This requirement is considered throughout the whole web interface implementation. Users can set up their own queries to the recorded data and analysis results, or just control the automated generated statistics and metrics. We use several visualization techniques to attract the attention of the analysts to the main results and assist to recognize current attacks in an early state. At the same time, the webinterface provides detailed information about the monitored attacks and the malware analysis results.

The webinterface of InMAS provides access to all the stored information that is collected with the different sensor types and analyzers. Each supported sensor type disposes its own web interface. These single purpose interfaces are accessible through buttons at the top of the main webinterface, see Figure 2

for an example. We use this subdivision to enable the partially utilization of our system, if some components of it are not needed. In conjunction with the implemented user and role management, it is also possible to enable and disable selected modules – or even single analyses results of a module – for a set of users or a single user. Thus, the webinterface can be easily customized and thereby optimized for the individual access rights and requirements of an analyst.

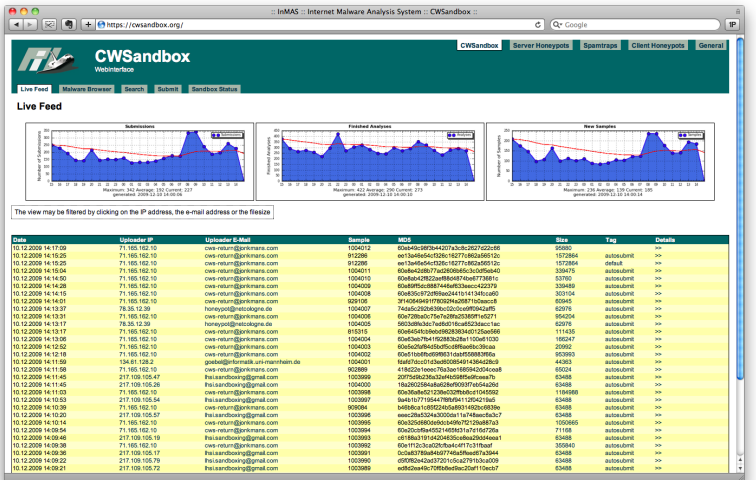


Figure 2: CWSandbox live feed section

The *Live Feed* pages offer a quick overview to the analyst. Depending on the individual subsystem, e.g., CWSandbox, server honeypots, spamtraps, or client honeypots, it reveals the latest submissions, attacks, accessed webservers, or received spam messages. Figure 2 shows the incoming submissions to the CWSandbox interface. The Live Feed shows only the general information of these submissions. All analysis-related information, like the antivirus scanner result and the behavior report, are accessible over a link within each entry. Figure 3 shows the details subpage for one CWSandbox submission.

As mentioned before, the webinterface also provides several analysis results to the analysts. Figure 4a shows the *analysis tool* section for the server honeypots. In this part of the InMAS webinterface, it is possible to have an in-depth look at all recorded attacks provided by the registered honeypots. Stored information are aggregated and displayed using visual methods like pie or bar charts. The chart in Figure 4a for example shows the number of attacks and downloads of malware for each day of the week, aggregated over a period of approximately two weeks. These analysis tool section exist for all InMAS subsystems.

Beside the configurable analysis tool sections, InMAS also provides an *analysis* section, where several statistics are summarized, Figure 4b shows this sec-

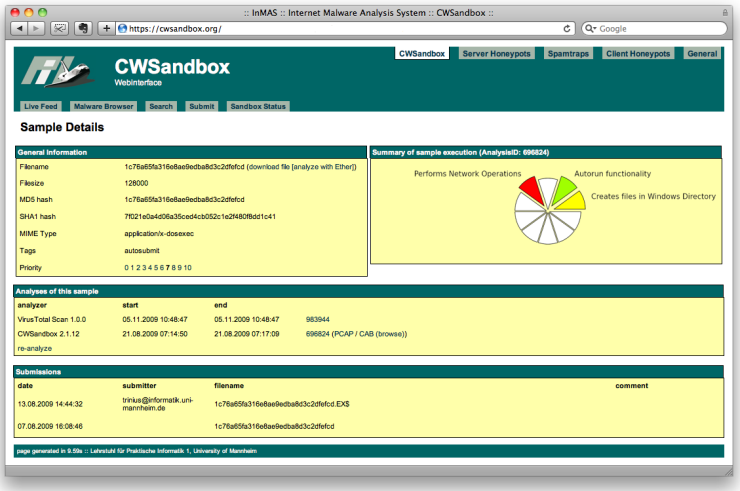
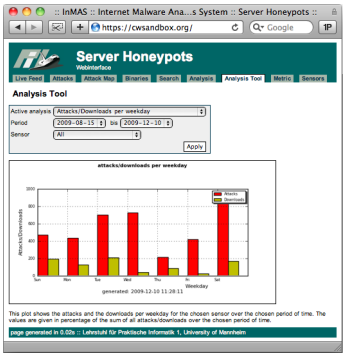
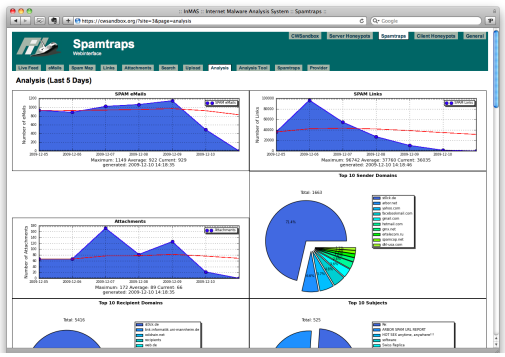


Figure 3: Details subpage of a CWSandbox binary analysis



(a) The server honeypots analysis tab showing recent attacks.



(b) The spamtraps analysis tab showing the number of emails received during the last seven days.

Figure 4: Examples for analysis results provided by InMAS webinterface.

tion for the spamtrap webinterface. This page shows all statistics that are also accessible in the *analysis tool* section, but with a fixed time period.

4.2 Visualization

Besides the visual preparation of statistics which shows global correlations, like the number of monitored attacks or the top command and control server, we also use visualization to support the analysis of behavior reports [18]. The CWSandbox reports, which often consist of several hundred lines of text, usually offer a level of detail that allows careful analysis, but overwhelm a human analyst who is interested in a quick assessment of a particular malware sample.

Figure 5 shows two visualization forms generated from a CWSandbox analysis report. The *tag cloud* in Figure 5a shows all Windows API calls which are monitored during an analysis run. The font size of each API call corresponds to the frequency within the behavior report. The tag cloud provides a quick overview of the main operations executed by the analyzed sample.

(a) Tag cloud visualization.

(b) Treemap visualization.

Figure 5: Examples for different visualization techniques used within InMAS.

The second form of visualization are so-called *treemaps* (see Figure 5b). A treemap displays information as a set of nested rectangles. For construction of the treemap we need to define a tiling algorithm: The width of each rectangle is proportional to the percentage of Windows API calls from the behavior report belonging to this section, i.e., if more API calls belong to a specific section, the according rectangle is wider. We also split each rectangle according to the operation of this API call to obtain another dimension within the treemap. The individual sections within the treemaps are plotted in a fixed order and color.

Since both the tag cloud and the treemap do not consider the time sequence in which operations occur, we use a third method for visualization. *Thread graphs* visualize the temporal behavior of each individual thread spawned by a sample. A thread graph can be seen as a behavioral fingerprint of a sample. Figure 6 shows an example for the thread graph of a behavior report.

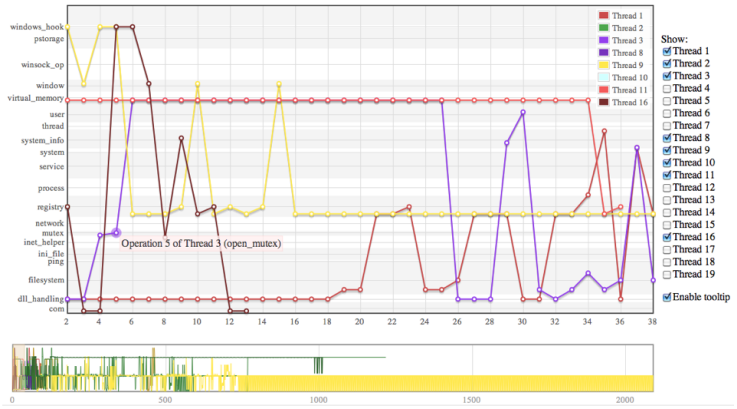


Figure 6: Thread graph visualization of a CWSandbox analysis report.

The x-axis represents the time (sequence of performed actions), while the y-axis indicates the operation/section of the performed action. The thread graph implementation – like the tag cloud and treemap – offers a zoom-functionality that shows detailed information about the Windows API calls selected. In addition, an analyst can choose which threads of the behavior report should be plotted. Overall, the thread graph shows all information which are contained in the CWSandbox analysis report in a more accessible way.

5 Conclusion

In this paper, we presented the Internet Malware Analysis System (InMAS), a highly automated system to collect and analyze malicious software. The system combines a unique set of sensors to capture malware as well as analysis tools for both static and dynamic malware analysis.

Currently, InMAS is operated at the University of Mannheim and receives an average of 200 malware samples per hour. Samples are submitted either through the public webinterface of CWSandbox or by deployed honeypots. With 15 sandbox systems running in parallel, we are able to analyze an average of 300 samples per hour. Thus, our installation successfully handles all requests and has additional capacity to re-analyze older or interesting binaries with different priorities. The modular setup of InMAS allows the integration of further sandbox hosts, in case the incoming rate of samples increases. Overall we have collected more than 910,000 distinct binaries and generated a total of about 1.77 million analysis reports within the last two and a half years. The analysis reports can be split up into about 840,000 VirusTotal results and 924,000 CWSandbox reports. As a result, our systems offers a detailed insight into todays' malware and its development over the past years.

Acknowledgments.

This work has been accomplished in cooperation with the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik (BSI)).

References

[1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. C. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *9th International Symposium On Recent Advances In Intrusion Detection, RAID06, Hamburg, Germany, September 20-22, 2006, Proceedings*, Lecture Notes in Computer Science 4219. Springer, 2006.

[2] U. Bayer, A. Moser, C. Krügel, and E. Kirda. Dynamic analysis of malicious code. *Journal in Computer Virology*, 2(1):67–77, 2006.

[3] J. Bennett. AutoIt Script Home Page. Internet: `http://www.autoitscript.com/`, 2009.

[4] J. Davis. Hackers Take Down the Most Wired Country in Europe. Internet: `http://www.wired.com/politics/security/magazine/15-09/ff_estonia`, Accessed: 2009.

[5] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. In *Conference on Computer and Communications Security*, pages 51–62, 2008.

[6] M. Engelberth, F. C. Freiling, J. Göbel, C. Gorecki, T. Holz, P. Trinius, and C. Willems. Frühe Warnung durch Beobachten und Verfolgen von bösartiger Software im Deutschen Internet: Das Internet-Malware-Analyse System (InMAS). In *11. Deutscher IT-Sicherheitskongress*, Bonn, Germany, May 2009.

[7] M. Engelberth, C. Willems, and T. Holz. MalOffice – Analysis of various application data files. In *19th Virus Bulletin International Conference (VB2009), Geneva, Switzerland, September 23-25, 2009, Proceedings*, 2009.

[8] J. Göbel. Amun: Python honeypot. Internet: `http://amun.sf.net`, Accessed: 2009.

[9] C. Gorecki. Trumanbox: Improving malware analysis by simulating the internet. Master's thesis, RWTH Aachen University, Department of Computer Science, 2007. Source code available at `http://trumanbox.s6y.org/`.

[10] R. Hes, P. Komisarczuk, R. Steenson, and C. Seifert. The Capture-HPC client architecture, 2009.

[11] T. Holz, M. Engelberth, and F. C. Freiling. Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. In *14th European Symposium On Research in Computer Security (ESORICS), Saint Malo, France, September 21-23, 2009, Proceedings*, Lecture Notes in Computer Science 5789. Springer, 2009.

[12] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. C. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In *LEET*, 2008.

[13] M. G. Kang, P. Poosankam, and H. Yin. Renovo: a hidden code extractor for packed executables. In *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 46–53, 2007.

[14] National Infrastructure Security Co-ordination Centre. Targeted Trojan Email Attacks. Internet: `http://www.cpni.gov.uk/Docs/ttea.pdf`, 2005.

[15] Norman. Norman sandbox information center. Internet: `http://sandbox.norman.no/`, Accessed: 2005.

[16] P. Porras, H. Saidi, and V. Yegneswaran. A Multi-perspective Analysis of the Storm (Peacomm) Worm. Technical report, Computer Science Laboratory, SRI International, 2007.

[17] B. Stock, J. Göbel, M. Engelberth, F. C. Freiling, and T. Holz. Walowdac: Analysis of a Peer-to-Peer Botnet. In *European Conference on Computer Network Defense*, 2009.

[18] P. Trinius, J. Göbel, T. Holz, and F. C. Freiling. Visual Analysis of Malware Behavior Using Treemaps and Thread Graphs. In *6th International Workshop on Visualization for Cyber Security (VizSec2009), Atlantic City (New Jersey, USA, October 11, 2009, Proceedings*. IEEE, 2009.

[19] S. Vömel. Using Honeypots to Monitor and Analyze Attacks Against Computer Systems. Master's thesis, University of Mannheim, 2009.

[20] C. Willems, T. Holz, and F. C. Freiling. CWSandbox: Towards automated dynamic binary analysis. *IEEE Security and Privacy*, 5(2), 2007.

[21] V. Yegneswaran, H. Saidi, P. Porras, and M. Sharif. Eureka: A framework for enabling static analysis on malware, April 2008.

[22] J. Zhuge, T. Holz, X. Han, C. Song, and W. Zou. Collecting Autonomous Spreading Malware Using High-Interaction Honeypots. In *Information and Communications Security*, Lecture Notes in Computer Science. Springer, 2008.