Practical Experiences with Purenet, A Self-Learning Malware Prevention System

Alapan Arnab[1], Tobias Martin[2], and Andrew Hutchison[1]

[1] T-Systems South Africa, International Business Gateway, New Road Midrand, 1685, South Africa
{alapan.arnab, andrew.hutchison}@t-systems.co.za
[2] Deutsche Telekom Laboratories, Deutsche Telekom Allee 7 64295, Darmstadt, Germany
tobias.martin@telekom.de

Abstract. This paper introduces Purenet, which is a self-learning malware detection system aimed at avoiding zero-day attacks and other delays in patching application systems when attacks are identified. The concept and architecture of Purenet are described, specifically positioning anomaly detection as the system enabler. Deployment of the system in an operational environment is discussed, and associated recommendations and findings are presented based on this. Findings from the prototype include various considerations which should influence the design of such security software including latency considerations, multi protocol support, cloud anti-malware integration, resource requirement issues, reporting, base platform hardening and SIEM integration.

1 Introduction

Malware (viruses, worms, spyware etc) is often cited as one of the main electronic security problems [3],[4]. While active attacks against a system are often targeted, and usually require a high degree of knowledge (of both technology and the targeted system), malware attacks are generic, high volume and - with the advent of relevant tools - can be largely automated.

Systems to prevent malware prevention therefore have a high visibility, and have more awareness in the general public than more advanced security solutions such as hardware encryption modules. Despite the larger awareness of such solutions, there are two problems with the effectiveness of such solutions:

1. Anti-Malware (and in general, endpoint security) solutions are not always kept up-to date. [9]
2. Systems (including all the relevant software on the system) are not patched with regularity, allowing certain types of malware to spread and infect before they can be detected by anti-malware solutions.

Furthermore, anti-malware solutions will often struggle to identify new threats that have not yet been identified (such as zero-day attacks, or attacks launched

on a software vulnerability that has no patch). The rapid growth of the Internet has also increased the number of malware threats, with a trend in faster exploitation of unremediated vulnerabilities. [10]

The Netcentric Security project, conducted at Deutsche Telekom Laboratories, considered a two pronged approach - firstly to prevent known malware from entering a protected network domain (in similar fashion to an Intrusion Prevention System); and secondly to detect new malware using machine learning techniques by Purenet. After a new malware has been detected, a signature of the file containing the malware is derived automatically and the intrusion prevention device is updated.

The structure of the paper is as follows: background information to motivate the approach is provided, Purenet is introduced as a concept, and then described in terms of its approach and architecture. The testing of the system is then outlined, and recommendations and findings based on live deployment of a system prototype are provided. Future work and conclusions are then provided.

2 Background

The approach of attacking malware dissemination at the Internet Gateway, is increasingly attracting attention - and analysts such as Gartner recommend increasing focus on this approach [2]. While Gartner's recommendations are based on leveraging existing technologies, such as Intrusion Prevention Systems and e-mail/URL filtering applications, Purenet's approach is to further increase the effectiveness of such systems by *proactively* scanning for new malware.

Figure 1 provides insight into why a proactive security approach is desirable. When unknown or new malware starts to circulate, a window of opportunity arises between the identification of such malware and new anti-virus patches and/or signatures being released. Attackers aim to exploit the Window of Opportunity, the time AV and software vendors need to identity new threats and provide patches. New malware can spread until signature databases are updated. Security flaws in software may be exploited until patches are available.

As detailed in figure 1 a security flaw arises in a timeline starting on 17^{th} January, and a window of opportunity exists between 20^{th} January (when an exploit is created / released) and the 4^{th} February when signature updates are released. So called 'zero day attacks' can take advantage of such windows of opportunity to do large amounts of damage in a short time.

3 Purenet as Concept

The "Netcentric Security" project, at Deutsche Telekom Laboratories, has a two pronged approach:

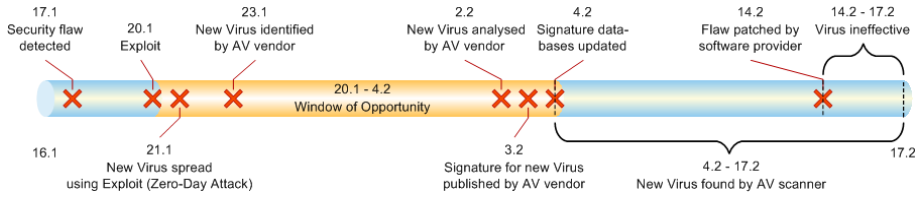

Fig. 1. Zero Day Attacks.

1. to prevent known malware from entering a protected network domain (similar to an Intrusion Prevention System); and
2. to detect new malware using machine learning techniques by Purenet.

In essence, the Purenet goal can be stated as detection of unknown malware (viruses, trojans or spyware) by code classification using machine learning techniques used in Artificial Intelligence (AI).

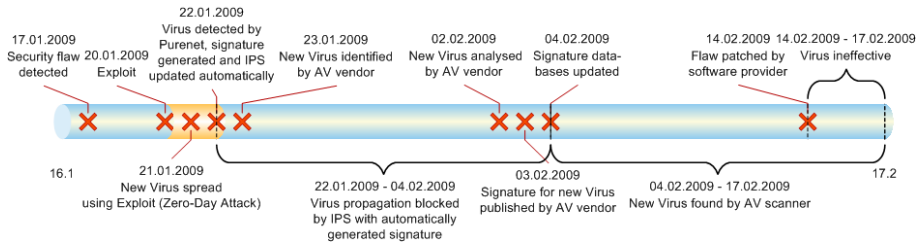

Fig. 2. Detection of unknown Malware

In contrast to Figure 1, Figure 2 shows how a system based on proactive malware detection can stop virus propagation immediately. New threats are detected by an AI system, IPS signatures are generated automatically. Identified threats are stopped at the entry to the network until AV signatures or software patches are available. The window of opportunity for attacks, and zero day possibility, are eliminated if direct identification of threats is possible.

Figure 3 illustrates the positioning of Purenet within an Internet services environment. Purenet is deployed in a DMZ and must be connected to the copy port of an IDS/IPS or network tap. Purenet performs sniffing and file reconstruction, new threat detection in files, signature generation and filter device update and alerting.

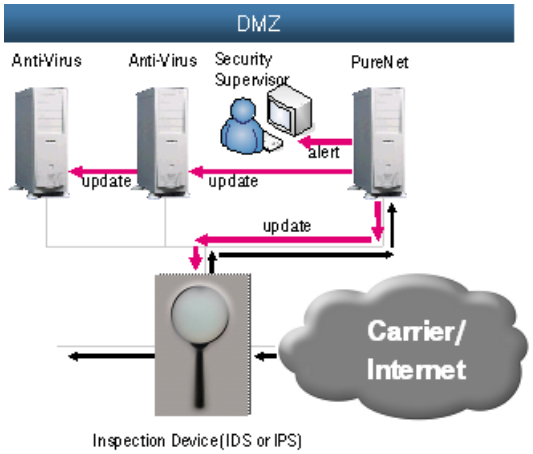

Fig. 3. Purenet deployment for an Internet Service Provider

A data link between the Purenet system and other Anti-Virus (AV) or Intrusion Prevention Systems (IPS) is configured so that any findings of the Purenet system can be propagated to other scanning devices immediately. The concept of Purenet is to do analysis and update in parallel to the inspection by the IPS with a slight time delay.

From a scalability point of view a network of distributed Purenet instances can also be deployed, with findings at any one instance propagated to other Purenet deployments for the purpose of updating associated AV or IPS devices at their locations.

4 Purenet Approach and Architecture

Purenet was designed with a modular architecture to allow live scanning of Internet data, across multiple file types for known malware. Identification of unknown malware is facilitated by time-delayed analysis of files in a so-called New eThreat Detection Module (NeDM). Once this NeDM classifies a file as malware, or as containing malicious code, a signature is generated automatically and directly so that the same pattern can be detected subsequently by the IPS-like live scanning process. In the initial implementation only HTTP traffic was monitored, and only Microsoft Windows executables and DLL files were scanned for malware.

Figure 4 shows the conceptual architecture of Purenet. Known malware is identified and blocked by the IDS/IPS appliance. Remaining traffic is monitored and collected by the Data Stream Manager (DSM). The New eThreat Detection

Module (NeDM) analyzes the captured files and the Signature Builder (SB) is activated in order to synthesize a signature of newly detected malware. The Storage Manager (SM) stores hash values for all analyzed files to detect recurrences. Purenet Control Center may be used by security experts to resolve conflicts.

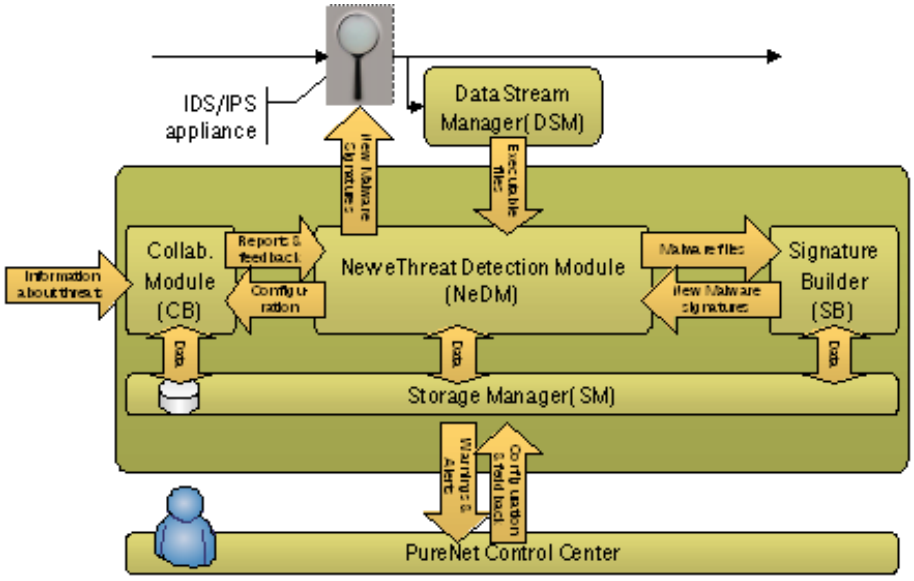

Fig. 4. Conceptual architecture of Purenet

4.1 Data Stream Manager (DSM)

The DSM extracts packets from a TCP/IP stream and extracts relevant files for the New eThreat Detection Module. It is fed with IP packets from the network and performs as a sniffer, filtering packets to discard traffic which is not relevant for the NeDM.

IP traffic is first filtered by the packet filter passing only http (TCP/IP port 80) and snf SMTP (TCP/IP port 25) packets to the capture module. The capture module delivers captured packets to the session extractor module reassembling a session and passing on to the file extractor module. Whenever a file is extracted, it is filtered by the recurrence filter to reduce duplications of files to be inspected by NeDM. New files are stored in a local file buffer before they are inserted in the Storage Manager and passed to the New eThreat Detection Module. The local file buffer is also used by the recurrence filter to check for duplications.

The Purenet prototype only contains a file extractor plug in for files contained in http and SMTP sessions. However, since the DSM has a modular architecture, additional plug ins, e.g. for POP3, IMAP4 etc. might be added.

4.2 New eThreat Detection Module (NeDM)

The NeDM is the most important module in the Purenet architecture. It does not process information in real-time. The architecture of NeDM is depicted in Figure 5. New files are received from the DSM and analyzed by various plug-ins which employ the feature extractor component that extracts for each plug-in the necessary features for the analysis process. Results from plug-ins are forwarded to the risk weighting module generating a final recommendation by combining the results from the plug-ins.

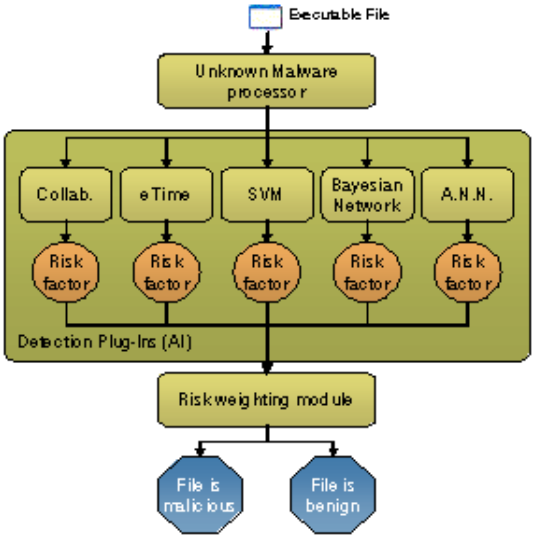

Fig. 5. Conceptual architecture of NeDM

The Unknown Malware processor is controlling the whole detection process. Its input is files collected by the DSM which are inserted into the Storage Manager (SM). If the file already exists, NeDM updates its statistics in the SM. Otherwise it is stores in the SM and its status is set tu 'unchecked'. Whenever NeDM is idle it retrieves the unchecked file with highest priority.

The various detection plug-ins analyze the file using different algorithms. Then, the results are combined by the risk weighting module, which takes also alerts from the Collaborative Module into account. If the risk grade is above a

maximum threshold, the file is classified as malicious and a signature is extracted. If the malicious risk grade is below a minimum threshold, the file is classified as benign. Otherwise it is forwarded to the Purenet Control Center for further inspection by human experts.

4.3 Detection Plug-ins

To enable flexibility in employing various algorithms for identifying new malware, plug-ins are used. All plug-ins have a similar interface, a file as input and a risk factor as output. Generally NeDM supports two types of plug-ins:

1. Static (structural) analysis plug-ins: An executable file is analyzed before being executed. Structural features are extracted from each file and analyzed by various machine learning techniques, such as Decision Trees, Bayesian Networks, etc.
2. Behavioral (dynamic) analysis plug-ins: Analyze executable files by executing the program in a supervised run-time environment.

Dynamic plug-ins require longer time to reach a decision as the file needs to be executed and it might take a long time until it starts its malicious activity. On the other hand, static plug-ins generate a decision in a few seconds.

Dynamic plug-ins should hence be used selectively and not each time NeDM inspects a new file. They might be used for instance on files that were classified as "expert review".

In the evaluation of Purenet presented in this paper static plug-ins based on machine learning algorithms were used exclusively.

These plug-ins work as follows: First, features that represent the file are extracted. Then the plug-in applies the classification model on the features and returns the results. The implemented plug-ins in this evaluation employ machine learning classifiers which were trained using the WEKA software (Witten and Frank, 2005), commonly used for these types of tasks [11].

In the evaluation nine different classifiers were used: Bayesian Networks, Nave-Bayes, K-Nearest Neighbors, Hyper Pipes, VFI, J48 Decision Tree, Random Forrest, OneR and PART. More details can be found in [8] and [1], see also [5],[6] and [7].

In the training phase a repository of 7,688 malicious files and 22,735 benign programs running on a Windows XP machine were used. For each file, three types for features were extracted: n-grams based features, Portable Executable (PE) features and function-based features.

The n-grams (3-grams, 4-grams, 5-grams and 6-grams of bytes) were extracted from the binary representation of a file. The PE headers of Win32 binaries (EXE or DLL) might indicate that a file is infected by a virus, e.g. inconsistencies between different parts of version numbers or internal/external name of a file. In total 88 features are extracted from PE headers. The Function method is a new method for extracting features from files: The beginning and end-points of functions in binary code is marked. Using the marks the following 17 features are extracted:

- Size of file; File's entropy value;
- Total number of detected functions;
- Average size and size of the longest and shortest detected function;
- Standard deviation of size of detected functions;
- Number of functions divided into fuzzy groups by length in bytes;
- Function ratio and function code ratio.

4.4 Risk Weighting

The risk weighting module provides a final rank for each file suspected as malware. It collects all risk factors from the relevant plug-ins and calculates a weighted rank. If the final rank is beyond a threshold, the file will be transferred to the Signature Builder which will construct a unique signature.

For the Purenet test the Distribution Summation weighting process is used, which is quite simply and outperformed most of other weighting algorithms. It accumulates the risk factors generated by the plug-ins for each class (i.e. benign and malicious). The class with the highest summed grade is chosen.

5 Purenet Testbed

A live testbed of the solution was implemented in the Internet gateway of T-Systems South Africa, which serves only large corporate clients. The intention of the testbed was to investigate the feasibility of the solution for corporate customers and to investigate the effectiveness of the detection module in a large environment with many concurrent connections. While the Purenet solution did not change the operational environment itself, all data collected and analysed was real world data, using a real time traffic feed.

Figures 6 and 7 provide typical traffic volumes through the T-Systems South Africa Internet gateway. It was in this context that the Purenet prototype was tested.

In addition to the actual throughput of data transversing the system, another important factor to consider is the number of simultaneous connections transversing the solution; as well as the traffic types of these connections. The

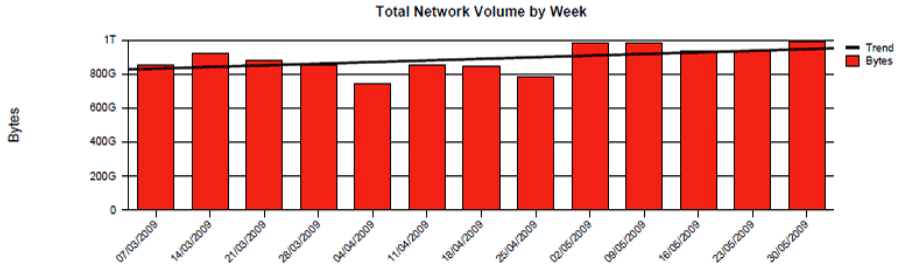

Fig. 6. Total Network Volume, per week (T-Systems SA Internet Gateway) in Bytes

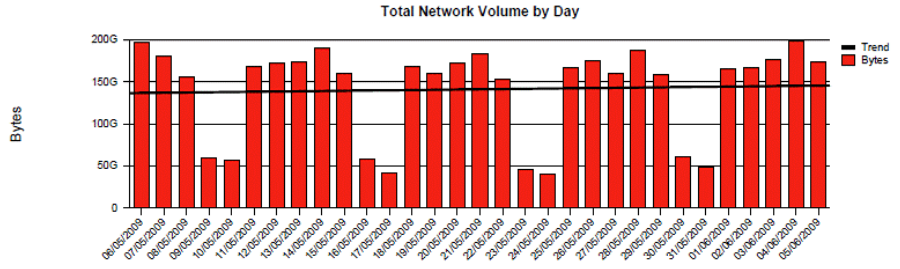

Fig. 7. Total Network Volume, per day for May 2009 (T-Systems SA Internet Gateway) in Bytes

performance of Purenet is not constrained by the data throughput but rather by the number of simultaneous connections that needs to be processed. Table 1 provides the total number of simultaneous TCP connections passing the testbed environment over a 5 minute interval at 12h50 on Friday 19 June 2009. This time period usually features the peak number of Internet users, and thus provides a good overview of the peak constraints for the testbed.

The architecture in which the Purenet prototype was deployed is shown in Figure 8. A domestic MPLS network acts as a distribution platform to/from the corporate customer community. Demilitarized zones are traversed as traffic moves between the Internet gateway and servers or users.

The configuration of Purenet for the testbed was such that various detection modules were incorporated. A "Known eThreat Handling Module" was present, together with a "Data Stream Manager", "New eThreat Detection Module", "Signature Builder" and "Storage Module". A control centre was also utilized to co-ordinate this combination of known and unknown, static and dynamic detection modules. A graphical overview of this configuration is depicted in Figure 9.

Application	Total TCP Connections
FTP	138
VPN	416
SSL	428
MSN	491
Skype	615
SMTP	15 681
HTTP/HTTPS	183 355

Table 1. Total TCP Connections over 5 Minute Interval

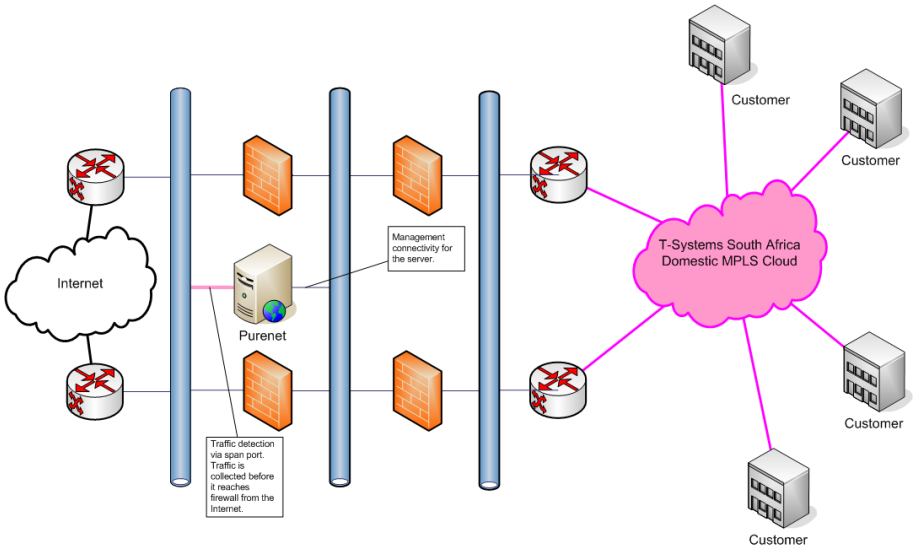

Fig. 8. Deployed Architecture for Purenet Testbed

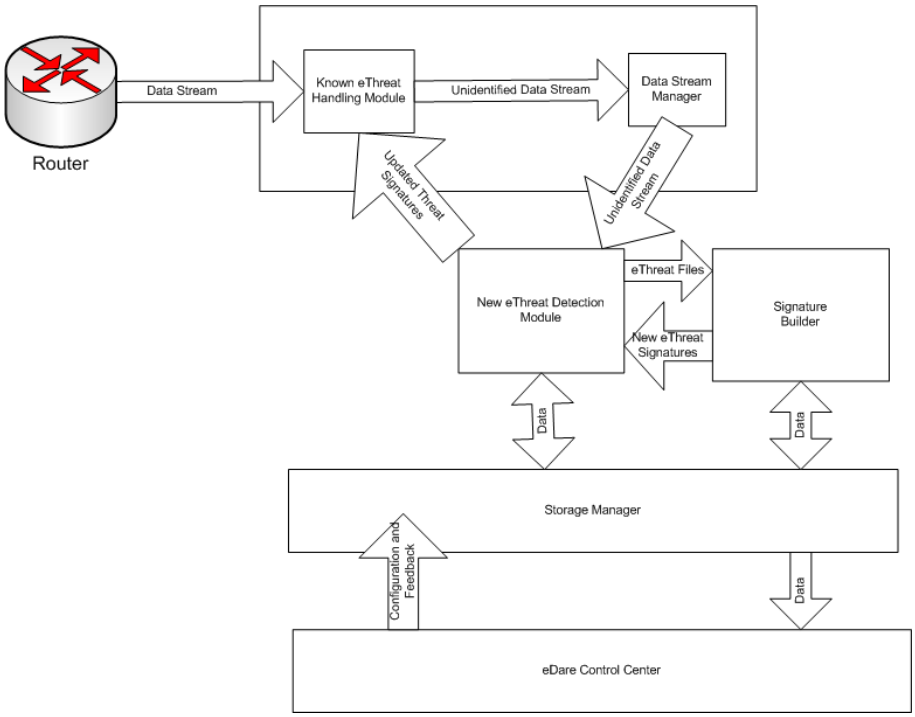

Fig. 9. Purenet Modules and Interaction (as setup on the Testbed)

6 Findings and Recommendations

The majority of findings relate directly to features and functionality requirements relevant to product development. In this section some details of each finding are presented.

6.1 Latency Impact

The practicality of the solution will be largely impacted by the speed at which it will process information; and therefore, the latency impact upon network traffic. This will also impact the implementation design of the solution - whether it should be placed in line with the network (thus forming part of an IPS device for example) or on a span port as discussed in this paper.

The other impact of processing latency, specifically in the out-of-line deployment scenario, is that in the case of a zero-day attack, the IPS signatures will not be updated before the first zero-day attack penetrates the network (since the malware packet will most likely enter the network before Purenet concludes that the examined packet was indeed malware). While the spread of zero-day attack malware can still be largely mitigated in this approach, a total protection against detected zero-day attacks cannot be enforced; limiting the applicability of this solution. However, since an alert will be issued by the Purenet control center and a signature will be created by the Signature Builder, the zero-day attack can still be stopped even after it entered the protected network.

The in-line application of the solution will address this problem (by allowing only "clean" files through. This approach will however have an obvious latency impact, and this latency requires further analysis. Furthermore, the latency impact will differ in terms of end user experience depending on the protocol used. For example, in non-synchronous applications such as e-mail (SMTP) there is no impact on the user experience despite the fact that some e-mails are delayed (e-mails containing new files are received not before the sending MTA starts the second try).

6.2 Analysis of malware over multiple protocols

The current Purenet solution only analyses http and SMTP traffic. While http and SMTP do comprise a substantial portion of Internet traffic, other protocols such as FTP, Skype and Bittorrent are also widely used and are more frequently used for transferring executable files. Support for a wider range of protocols is thus necessary for the solution to be more generally effective for its purpose.

6.3 Support for more file types and platforms

The current Purenet solution focuses on Microsoft Windows executables and Microsoft Windows DLL files. A significant proportion of malware exists in other

file formats such as embedded files in Microsoft Office documents and Active-X applets. Detecting and mitigating vulnerabilities in Web-2 application frameworks (such as Microsoft's Silverlight and Adobe's AIR) will also enhance the value proposition of Purenet.

Another enhancement of Purenet's value proposition would be to detect malware on multiple platforms; specifically Unix based platforms such as GNU-Linux and Solaris. While there are a number of anti-malware solutions for Microsoft Windows platforms, there is little variety in Unix based anti-malware solutions. Despite the importance of Unix servers in enterprises, and the growth of Unix based operating systems such as GNU-Linux and Apple's OS X on the desktop, anti-malware solutions are rarely deployed on these platforms.

Implementing the anti-malware solution on POSIX standards, the complexity of catering for multiple Unix variants could be diminished.

6.4 Integration with Cloud Anti-Malware Solutions

In the current Purenet solution, a detection scan can produce three possible outcomes:

1. Determine that the file is not malware
2. Determine that the file is malware
3. Request that the file is examined by an expert to determine its status

In the last scenario, Purenet could utilise the emerging Cloud Anti-Malware solutions to increase the speed of resolving the status of undetermined files.

6.5 Reporting

The advantage of solutions such as Purenet, lies in the ability of the solution to generate data regarding the emergence of new malware threats; as well as adaptive solutions to combat such threats. The analysis of the data collected through the operation of solutions such as Purenet would further enhance the general understanding of malware threats; especially in environments with large diverse endpoints - such as Internet service providers.

Statistical data to back-up the functionality of Purenet is also required, and can be a separate module, directly interfaced with the database. The following reports would be expected from such a system:

- Analysis of file types over a specific time period
- Trend analysis of detected malware
- Analysis of files whose status could not be detected
- Analysis of top sources/domains of malware
- Analysis of malware based on protocol usage
- Traffic analysis in terms of clean vs malware infected files.

The above information would not only enhance the security of the networked environment, but also allow Internet service providers to better understand the traffic patterns of malware distribution and if applied correctly; top source and destination of malware distribution.

7 Future of Purenet

The Netcentric Security project has been used to introduce the concept to potential business customers of T-Systems in Germany. The data collected from Purenet showed promising results in detecting new and unknown malware.

From market analysis, feedback from assessment reports indicates that the product concept is perspicuous. However, while enterprise customers were interested in principle they would rather expect to purchase such a product from an established vendor of anti virus solutions, since there is an expectation of the product being maintained and further developed. This ultimately led to a decision to sell patents and technology to an established anti-malware vendor. There is continued development however on the analysis approach, and there are still opportunities to deploy this approach as part of a differentiated value added service for service providers.

8 Conclusion

In summary, the Purenet testbed provided a valuable confirmation of the requirement and solution approach. In terms of deploying the system in practice, the research endeavor was an example of a successful collaboration between the DT research entity and its enterprise provider unit, in terms of developing and field-testing the system. It was beneficial to deploy the Purenet solution as a testbed within the TSSA Internet gateway.

From the testbed it was possible to obtain live data from a production system, and this gave insights into areas including:

- Buffer design
- Enhancements for higher traffic volume
- Data obtained in terms of scalability of solution
- Hardware utilisation showed over-spec of systems requirements

The concepts of the Purenet testbed can be included in future projects and solutions, and the findings and recommendations are of general usefulness for internal successors and for other security researchers to consider.

References

1. Elovici, Y., Shabtai, A., Moskovitch, R., Tahan, G., Glezer, C.: Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic, The 30th German Conference on Artificial Intelligence (KI-2007), Osnabruck, Germany, September 2007.

2. Firstbrook, P: Why Malware Filtering Is Necessary in the Web Gateway. Published 2008-08-26 Gartner. Gartner ID: G001584595.
3. Heidari, M: Malicious Codes in Depth (2004) http://www.securitydocs.com/pdf/2742.pdf
4. Kienzle, D.M., and Elder, M.C: Internet WORMS: Past, Present, and Future: Recent worms: a survey and trends. ACM workshop on Rapid malcode (WORM03), (2003)
5. Moskovitch, R., Stopel, D. Feher, C., Nissim, N. and Elovici, Y.: Unknown Malcode Detection via Text Categorization and the Imbalance Problem, IEEE Intelligence and Security Informatics, Taiwan, 2008.
6. Moskovitch, R., Feher, C., and Elovici, Y.: Unknown Malcode Detection - A Chronological Evaluation, IEEE Intelligence and Security Informatics, Taiwan, 2008.
7. Moskovitch, R., and Elovici, Y.: Unknown Malicious Code Detection - Practical Issues, 7th European Conference on Warfare and Security, Plymouth, UK, 2008.
8. Moskovitch, R., Nissim, N., Elovici, Y., Acquisition of Malicious Code Using Active Learning, KDD08 Workshop on Privacy, Security and Trust in KDD (PinKDD08), Las Vegas, USA, 2008.
9. NCSA Study: http://www.staysafeonline.info/pdf/safety_study_2005.pdf (2005)
10. Symantec: 2006 Security Report (2006) http://www.symantec.com
11. Weka software, http://www.cs.waikato.ac.nz/ml/weka/