New model of antivirus protection of computer network users

Jaroslav Lámer, Ivan Klimek and František Jakab

***Abstract*—This article describes theoretic design and implementation of a network model for protecting computer network users from malicious software. Data stream analysis is realized as a "man in the middle" service. The implementation also suppresses threat of DDoS attacks originating in stations of the protected segment of network. The model is implemented as socks proxy server in C/C++ programming language. Main functionality is verifying downloaded files against Cloud AV system and databases of malware and phishing websites. Files that are not binary and still can become source of an infection are checked using file type analyzer. The implementation is appropriate for deployment by internet service providers. It includes methods for data stream optimization and database structure representing known results of previous inspections. SmartScreen, a closed source protocol created by Microsoft was analyzed using reverse engineering methods for purpose of using Microsoft cloud database.**

***Index Terms*—security, antivirus protection, dataflow protection, badware protection firewall, socks proxy**

I. Introduction

The main problem, that this work solves is to find one of effective antivirus protection model, that is placed in I/O computer network node, checks all of network traffic going to or from internet and finally combines more effective antivirus solutions from different developers to one solution. One type of solution is a centralized type protection system located in I/O network node, that logical placement is characterized as "man in the middle" system. All kind of network traffic to/from internet flows through this system. In case of executable file (or other which can carry any kind of infection) detection is this file scanned with some scanning procedures. All kind of potentially dangerous files are scanned in on-line badware database for malicious content or optionally in reputation based database. Executable files are scanned with cloud of commercial antivirus software. Other files like images are analyzed with file content analyzer. If result of all tests is negative, file is directly forwarded to client. In other cases is file dropped and warning page forwarded.

This architecture ensure actual and homogenic antivirus protection in whole network and limit spread of infection from weak secured computers in network. Problem of this system architecture is impossibility to scan files stored on clients PCs. This problem is solvable with locally installed antivirus software. System is therefore not a primary antivirus protection. It is an additional protection system in local area network to standard antivirus software installed on users computers. Centralized solution is a way to eliminate DDoS attack at its inception [1].

Manuscript received November 20, 2012.

Jaroslav Lámer is with the Department of Computer and Informatic, Technical University of Košice, Slovakia (jaroslav.lamer@tuke.sk).

František Jakab is with the Department of Computer and Informatic, Technical University of Košice, Slovakia (frantisek.jakab@tuke.sk).

Ivan Klimek is with the Department of Computer and Informatic, Technical University of Košice, Slovakia (ivan.klimek@tuke.sk).

II. Goals

The goal of this work is to design an centralized antivirus system model based on third party antivirus and reputation kind of services, which no need to update local services and run separately on remote trusted server. The second goal is, that will be absolutely not important, what web browser user use for its work (security in all browsers will be equal and high). And finally the last goal is to try to use (explain) one of non-open source (closed) service, originally used for one commercial system (service). Designed model implement as socks proxy server in C or C++ language.

III. Analysis

As next image shows, the main reason of bot-net network and other badware existence is not an ideology. It's financial benefit. When attacker need to disable any server, pay some credits to provider of bot-net network and wait for subscribed attack.[2]

To build this network, an infiltration to user PC is needed. This is in many situations the same process. Seamlessly harmless code is downloaded from USB key, or internet. [3] This simple and small application is not shy for antivirus and run (optionally with owner rights) is allowed. When application detects internet connection, download malicious software and run it. [4]

This means, that run of those simple applications is harder to stop (it's not important, because they are not malicious). But downloading malicious software is stoppable by antivirus system in network node.

Other problem is based on updating. If exist any solution to detect the malicious code, would be unknown for service, they have not actual database of solutions or signatures of malicious codes. Whole system need to be up to date.

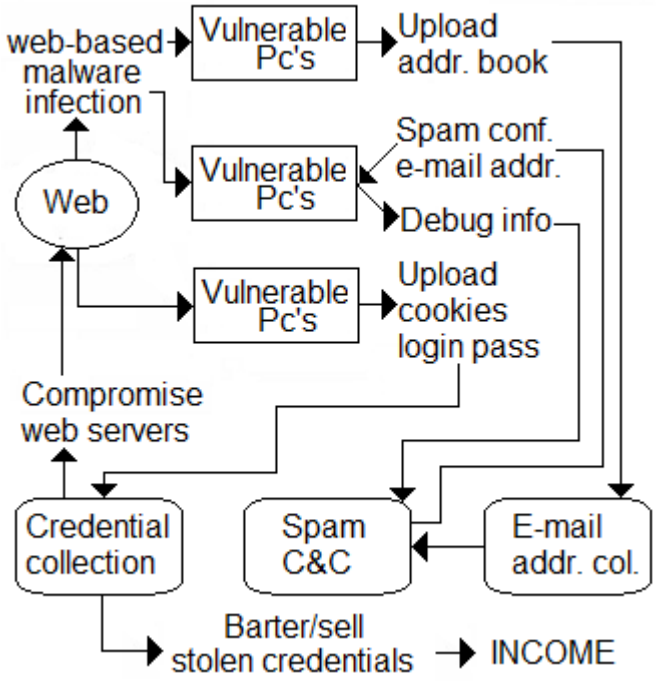

Fig. 1 Life cycle of malware with result for its provider. [2]

IV. Solution and Results

A. *The design of described model*

Next image shows position of Protect system in LAN network.

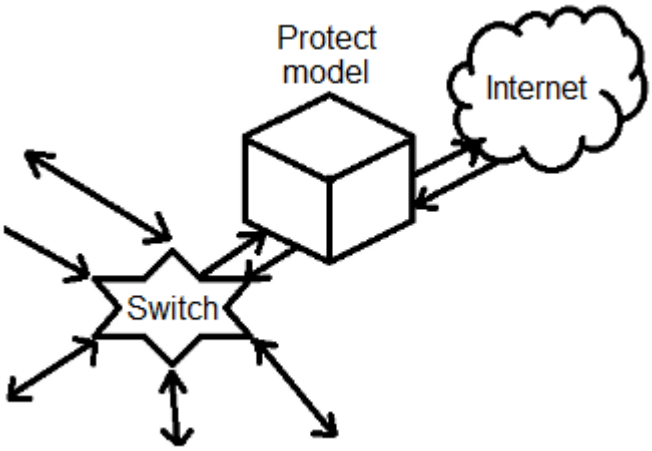

Fig. 2 Position of designed model in computer network.

Every part is implemented as separate function. It is based on fast C++ programming language.

System is directly depending on third party services. The main part is a socks proxy server, which provides connectivity to internet. Each http request is forwarded to public server. Main filter and functionality is based on respond traffic. This helps protect users from redirecting or sending an error responds from malicious servers.

Identifying Malicious Web Sites Via A Remote URL Blacklist will be performed with two of existing solutions: Google's Safe Browsing (primary designed for Firefox/ Safari) and Microsoft's SmartScreen (original only for the Internet Explorer).[18]

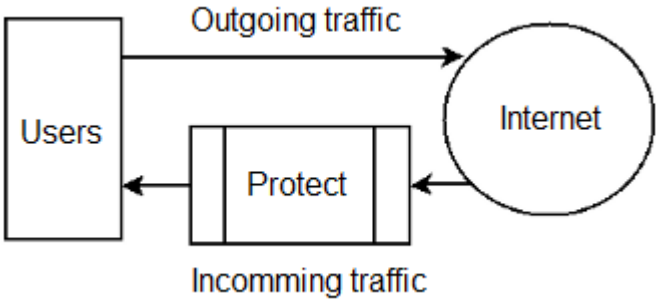

Fig. 3 Captured traffic by designed model.

Tests of these solutions (from year 2009) explain, that SmartScreen blocked up to 83% of phishing and 81% of malware attacks. SafeBrowsing blocked 80% of phishing and 27% of malware attacks.[19]

Implementation of Protect model is divided to three services.

1) Google SafeBrowsing malware scanner or Microsoft SmartScreen Filter (the last filter is a kind of closed (non-open source) service, therefore his reverse engineering is needed)
2) Jotti Cloud AV scanner
3) Libmagic file analyzer

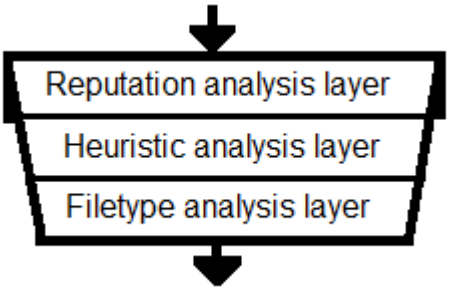

Fig. 4 Model protect layers

When user send request to web server, request is forwarded directly to internet. Scanning will start when proxy catch a respond. No all responds are controlled. Proxy have filter to recognize potentially dangerous files. Viruses and malware infections pack or append herself to other not dangerous files like PDF, JPG, FLASH and other documents with size no larger than 2MB. These files will be also ideal candidates to scan to. In all parts of system, when scan is positive, file is dropped and scan results is delivered to client.

First scan is one of Google safe browsing malware check. Google Safe Browsing initially designed and developed by Google and distributed as part of the Google Toolbar, the former Safe Browsing extension is now licensed under the Mozilla Public License and an essential part of Firefox and Safari. The component checks the sites a user visits against regularly downloaded lists of reported phishing and malware sites. If a URL or domain matches an entry in the list, a warning message is displayed to the user. In the newest

version, it also supports live lookups with up-to-the-minute fresh lists for every URL instead of using the cached local versions. Since the protocol is well-structured and openly defined, the provided lists could come from any server that implements the system. However, due to its origin, both browsers use the Google servers by default. That is, the lists of phishing and malicious Web sites are maintained by Google which, according to ZDNet, uses a combination of automatic (honey clients) and community-driven efforts to analyze a Web site". The protocol is based on simple HTTP request/response-cycles and supports blacklists as well as white-lists. It differentiates between malware-, phishing- and white-lists and supports various list formats, including regular expression lists or hashed lists of URLs or domains, respectively. It typically updates them every 30 minutes and usually only compares the visited URLs to the local lists. However, if a Web site matches a local list entry, it double-checks the URL using a live lookup to make sure that the entry is still up-to- date. [18]

Safe Browsing is a service provided by Google that enables applications to check URLs against Google's constantly updated lists of suspected phishing and malware pages. This service have next three functions:

1) Warn users before clicking on links that appear in your site when they lead to malware-infected pages.
2) Prevent users from posting links to known phishing pages from your site.
3) Check a list of pages against Google's lists of suspected phishing and malware pages. [12]

The APIs consists of two different types of HTTP requests:

1) https://sb-ssl.google.com/safebrowsing/getkey: This request requires SSL support. It provides the client with a private key for confidential communication with the server.
2) http://sb.google.com/safebrowsing/update: The client provides a list of tables that it wants the server to update. The server either provides the full content of the current tables or incremental updates to bring the client's tables up to the current version. [13]

For using this kind of services, API-key identifier is needed. This key authenticates the user as a valid API user. It is possible to get it on Google pages. [13]

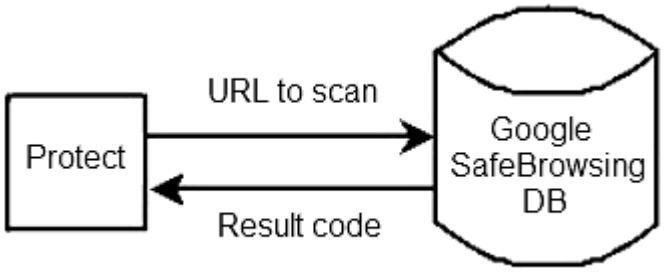

Fig. 5 SafeBrowsing API v1. communication process.

This scan checks URL in respond. Google provides two versions of malware scanner API (Application Interface). API v1 and API v2. [5] Both versions have limited count of http request per day (10 000). API v1 is easy to use GET or POST request oriented system, with malware database on remote server. This system doesn't need any local storage systems. All data are stored in remote database. Thanks to this feature is still up to date and a good choice for Protect model.

API v2 is a distributed database system. Database is saved on local server and is partially updated every day. Main database is stored on SafeBrowsing remote server. This interface is economically better variant, because it safe I/O Internet traffic. Local database is implemented as SQLite file, where URL and infection code is saved. But this advantage becomes to disadvantage, because there is need to update.

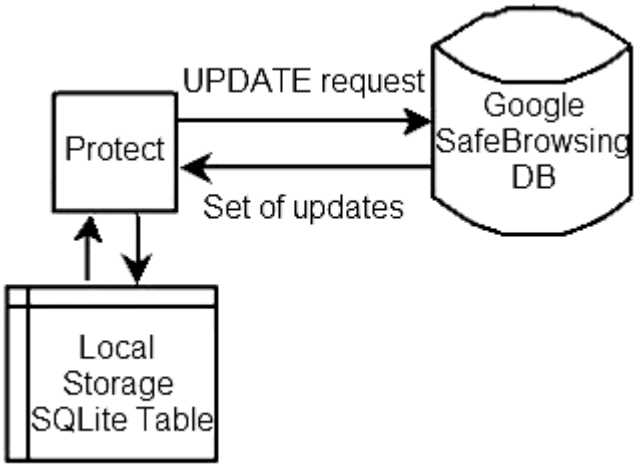

Fig. 6 SafeBrowsing API v2. communication process.

When this test past, the Microsoft SmartScreen filter becomes to be call (optional). SmartScreen is just other reputation type database based on user page rating. It is a remote database on Microsoft server cloud.

Since version 7 of the Internet Explorer, Microsoft also included phishing protection in its browser. The so called Phishing Filter detects only phishing attempts, but does not protect users from drive-by downloads on malicious Web sites. The recently released Internet Explorer 8 extends the Phishing Filter by the missing anti-malware protection and has been rebranded to SmartScreen. [18]

SmartScreen Filter can also help protect you from downloading or installing malware (malicious software). Through browsing the web, it analyses webpages and determines if they have any characteristics that might be suspicious. If it finds suspicious webpages, SmartScreen will display a warning message. It check the sites they are visited against a dynamic list of reported phishing sites and malicious software sites. If it finds a match, SmartScreen Filter will show a warning notifying. It check files that user download from the web against a list of reported malicious software sites and programs known to be unsafe. If it finds a match, SmartScreen Filter will warn user that the download has been blocked for user safety. It also checks the files that user download against a list of files that are well known and downloaded by many Internet Explorer users. If the file that user downloading isn't on that list, SmartScreen Filter will display warning. [10]

Although it also keeps a regularly downloaded list of known

safe sites, it queries Microsoft's server for most of the visited Web sites. [18]

These sites and malware downloads are sourced by Microsoft Internal and 3rd party database. The database also includes sites with Extended Validation Certificates which attest to the identity of legitimate business. [19]

This feature is original used to stop attackers from installing malicious controls (ActiveX components into IE) or stealing users sensitive information.[20]

It helps combat these threats with a set of sophisticated tools:

1) Anti-phishing protection to screen threats from imposter websites seeking to acquire personal information such as user names, passwords, and billing data.
2) Application Reputation to remove all unnecessary warnings for well-known files, and show severe warnings for high-risk downloads.
3) Anti-malware protection to help prevent potentially harmful software from infiltrating your computer. [11]

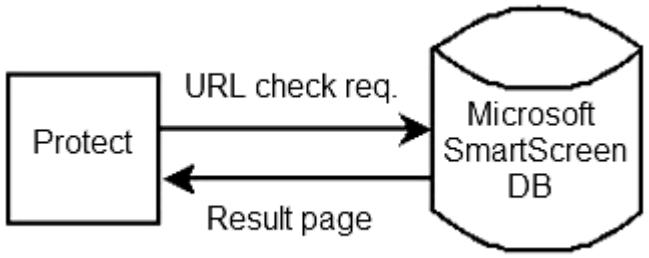

Fig. 7 SmartScreen communication process with protect model

Protect model send checked URL to this server through POST type of http request and wait for response. If URL is safe, scan return blank page. Otherwise link to Mozilla warning page is returned. That is pointer to dangerous file on requested URL. Positive scan of this type have equal solution to warn user as SafeBrowsing API positive result type. System is function equivalent to SafeBrowsing API v1 with no limit of http request per day. Therefore is SmartScreen a kind of ideal solution, there is no need to locally update. But this service has one big disadvantage. It is a kind of commercial product. Therefore is the communication protocol not open and not descripted (reverse-engineering is needed).

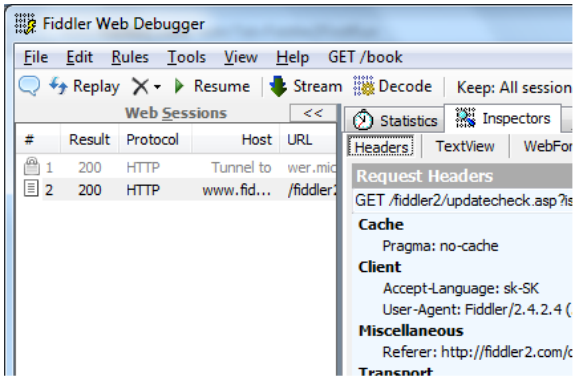

Fig. 8 Fiddler - software for communication view

Kind of reverse engineering is done. As was discovered, the communication protocol is based on SSL / TCP type of connection. In this communication channel communicate client and server through HTTP POST requests and reply's. Implementation of SSL certificate and view of message content is provided by software – Fiddler.

Analyzing of communication was producing next results. Communication for one database check is performed with one client request and one server reply. Requests are divided to few parts:

1) Link - POST https://urs.microsoft.com/urs.asmx
2) Parameters
   MSURS-Client-Key
   MSURS-Patented-Lock

These parameters are changing in time. Request body has more different parameters, but those never changing. But one parameter change. It is the parameter between tags <P> </P>.

The server respond is one of two types (first is negative result, second is positive result):

url.of.requested.serverPHSH:005:1:11:110080010000000000-0000-0000-0000-0000000000000

or

url.of.requested.serverPHSH:005:1:11:1100800crackz.wsPHSH:005:1:11:1100800www.mozilla.com/firefox/its-a-trap.htmlPHSH:100:0:10:0300www.mozilla.com/firefox/its-an-attack.htmlMALW:090:0:10:0300100930FA926-F4A0-4BE8-B56B-7B4F3ECCACB00

Reverse engineering was provided by one of debugging software – OllyDbg. For exploring a running program, attaching to running process in memory and finding analyzed strings is needed. To Internet Explorer is dynamically linked module called ieapfltr.dll. This is SmartScreen filter module.

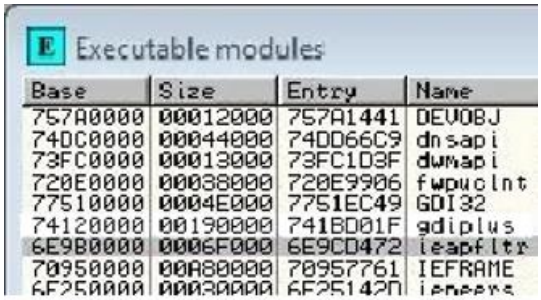

Fig. 9 SmartScreen filter module in computer memory

In this module were found all of analyzed strings and parameters in HTTP requests.

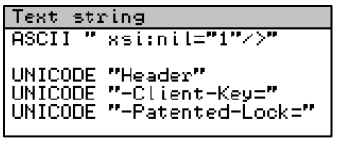

Fig. 10 Parameters in memory (found in strings)

In place, where this strings become to concatenate with calculated parameters was placed one break-point and with back-trace method found function that calculating this parameters. For patented lock and client key parameters is this function known as BASE64 encode. For example (client key parameter):

1) In URL ENCODE format:
   Ev62tIBFw8cafhSN62PExQ%3d%3d
2) In URL DECODE format:
   Ev62tIBFw8cafhSN62PExQ==
3) In BASE64 DECODE mode:
   12 FE B6 B4 80 45 C3 C7 1A 7E 14 8D EB 63 C4 C5

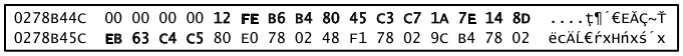

Fig. 11 Decoded client-key parameter

These bytes were generated by another function called before BASE64 algorithm. Input of this function is the body of client request. Therefore is this parameter directly depended to <P> parameter (included in body). Output of this function is every time of the same length. This evokes use of hash kind of function. Software called HashCalc is designed to calculate the most widely used hash algorithm (MD5, SHA256, etc.).

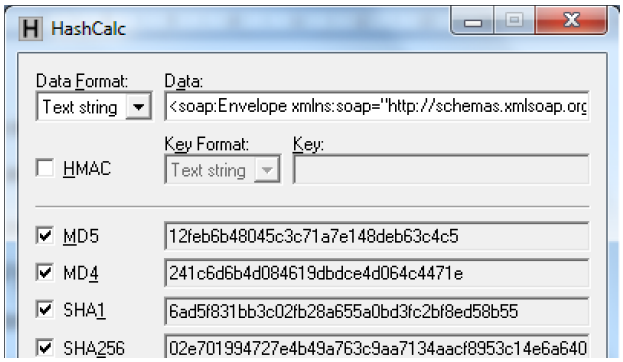

Fig. 12 HashCalc software to calculate hash strings

As picture shows, the string is result of MD5 hash function. The parameter Client-Key is therefore defined as BASE64encode(MD5(request body)). This explains the parameter time-changing (dependency on parameter <P>). Parameter <P> is calculated before other parameters. On next image is illustrated function for generating this parameter (found by back-tracing from MD5 function call).

Fig. 13 Function to generate <P> parameter

Description of this function is simple, because on 16 row is directly called Windows shared functions ole32.CoCreateGuid and ole32.StringFromGUID2. Second function converts Global Unique identificator (GUID) to string of printable characters [15]. GUID is Microsoft implementation of UUID. One part of UUID is a time variable [16]. This explains time-variability of generated parameters.

Patented-lock is the last parameter. Result of generation of this parameter is converted by BASE64 algorithm, but the core generating function is other. Using back-tracing, the generation function was found (is calculated after client-key). Function is similar to MSNpiki Challenges algorithm. But have some differences. Core of function is combination of the MD5 hash of request body (client-key) and the request body divided to parts of 4 bytes. If the length of request body string is not dividable with number 4, the request and its length is clipped down to the next length, which is dividable with number. 4. Function code was re-generated from binary to pseudo-C language and is available at [17].

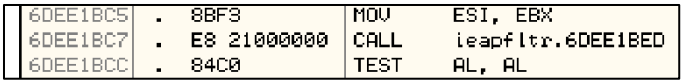

Fig. 14 Function to generate patented-lock parameter

The file scanning method is Cloud Antivirus scan. This scan is the most complicated, because it need a lot of time to do. Scan is provided by more anti viruses at once. Implementation uses Jotti cloud antivirus.

Jotti's malware scan is a free online service that enables to scan suspicious files with several anti-virus programs. Scanners used are Linux versions; detection differences with Windows versions of the same scanners may occur due to implementation differences. There is a 25MB limit per file (not important - protect scan only files smaller than 2MB). Files uploaded to this service are shared with anti-virus companies so detection accuracy of their anti-virus products can be improved. It is possible to do scan via MD5 or SHA1 hash string. This is great advantage to reduce dataflow, when needed file was already scanned. [14]

Requested file is downloaded to local temporary storage and uploaded to Cloud AV (Jotti). Because this operation cost a lot of time, wait page is send to client. When scan finished, result if positive or link to file if negative scan is delivered to waiting client.

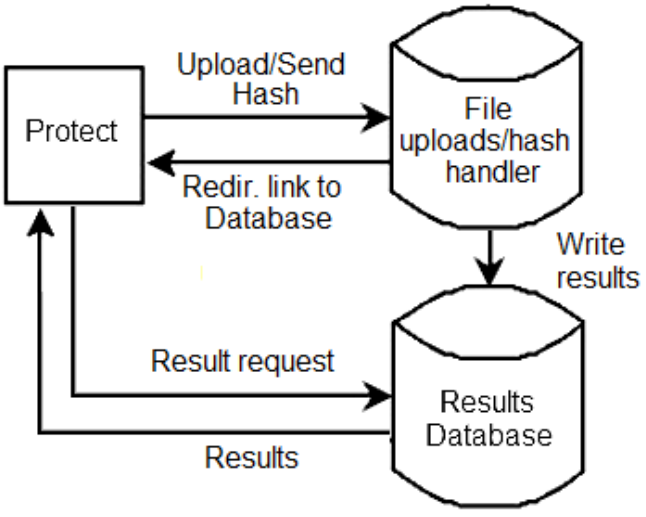

Fig. 15 Getting results of scan in Jotti Cloud antivirus

Files like images, they are dropped from Cloud AV scan are analyzed with file content analyzer - libmagic. File type is set as Mime-Type string. This function return file type (Mime-Type string) based on file content. Result is compared with file type string delivered in http request header. When different file type returned, file is dropped and warning page is returned to client. In other case is file directly forwarded. Libmagic is the only service in Protect model, that is based on local computing. This is not a problem, because file types are long-time the same. Therefore is no need to daily (or weekly) update.

Magic file (database) is used by FILE command. The FILE command identifies the type of a file using, among other tests, a test for whether the file contains certain "magic patterns". [8]

File tests each argument in an attempt to classify it. There are three sets of tests, performed in this order: filesystem tests, magic tests, and language tests. The first test that succeeds causes the file type to be printed. The filesystem tests are based on examining the return from a stat system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you are running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they are defined in the system header file. The type printed will usually contain one of the words text (the file contains only printing characters and a few common control characters and is probably safe to read on an ASCII terminal), executable (the file contains the result of compiling a program in a form understandable to some UNIX kernel or another), or data meaning anything else (data is usually 'binary' or non-printable). Exceptions are well-known file formats (core files, tar archives) that are known to contain binary data. [9]

The file "/usr/share/misc/magic" specifies what patterns are to be tested for, what message or MIME type to print if a particular pattern is found, and additional information to extract from the file. Each line of the file specifies a test to be performed. A test compares the data starting at a particular offset in the file with a byte value, a string or a numeric value. If the test succeeds, a message is printed. Each top-level magic pattern is classified as text or binary according to the types used. A top-level pattern is considered to be a test text when all its patterns are text patterns; otherwise, it is considered to be a binary pattern. When matching a file, binary patterns are tried first; if no match is found, and the file looks like text, then its encoding is determined and the text patterns are tried. Tests are arranged in a tree-like hierarchy: If a the test on a line at level n succeeds, all following tests at level n+1 are performed, and the messages printed if the tests succeed, until a line with level n (or less) appears. [8]

If a file does not match any of the entries in the magic file, it is examined to see if it seems to be a text file. ASCII, ISO-8859-x, non-ISO 8-bit extended-ASCII character sets (such as those used on Macintosh and IBM PC systems), UTF-8-encoded Unicode, UTF-16-encoded Unicode, and EBCDIC character sets can be distinguished by the different ranges and sequences of bytes that constitute printable text in each set. If a file passes any of these tests, its character set is reported. ASCII, ISO-8859-x, UTF-8, and extended-ASCII files are identified as 'text' because they will be mostly readable on nearly any terminal; UTF-16 and EBCDIC are only 'character data' because, while they contain text, it is text that will require translation before it can be read. [9]

Problem with downloading scanned files to local cache memory is usable in reduction I/O Internet traffic. When respond header with file, which is in local cache and have the same size is returned, system don't download this file. Forwarded is file stored in local cache.

B. Comparison with other similar products

Model was compared with AVG Network and Eset Gateway systems. Either compared products are commercial. [6,7] The difference from Protect model is their local database, which need to be periodically updated and there have not any kind of reputation based scan (ranking). Model protect don't need any update, because database of malicious software is stored on remote third party servers. This is one of model advantages, but one weakness too. When remote server with virus database fail or shut-down, scanning process is not fully functional. Databases both of compared antivirus products are included in Jotti antivirus cloud. Efficiency of Protect model is therefore great or equal as compared products.

TABLE 1
COMPARSION WITH OTHER SIMILAR PRODUCTS

Product	MD	VD	WD	Cach	MD	AR	Lic
Model Protect	Y	Y	N	Y	Y	Y	GPL
AVG Network [6]	Y	Y	Y	N/A	N	N	Trial
ESET Gateway [7]	Y	Y	Y	N/A	N	N	Trial

MD - Malware detection, VD - Virus detection, WD - Whole dataflow analysis, Cach - dataflow caching, MD - More databases, AR - Application reputation, Lic - License type, Y, N, N/A - Yes, No, Not Available

V. CONCLUSION

Prototype of designed model demonstrates possibility to secure users of computer networks with one centralized system and additional support from local antivirus installed on users PCs.

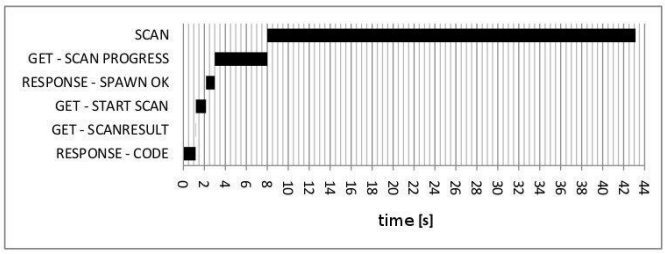

Fig. 16 Timeline of Protect model Jotti scanning process.

It is possible to improve prototype stability, security and performance properties. One example is the local antivirus cloud system. As next image shows, the length of the SCAN part of scanning process provided by Jotti malware scanner is the most critical.

Jotti's scanner scan times for files with different sizes up to 1MB are close to equal and depended on server load. Server load of this malware scan service is frequently on high values and directly affect whole respond and scan time. Local antivirus cloud system improvement will therefore drastically reduce scan waiting time, but add one big disadvantage – local update need.

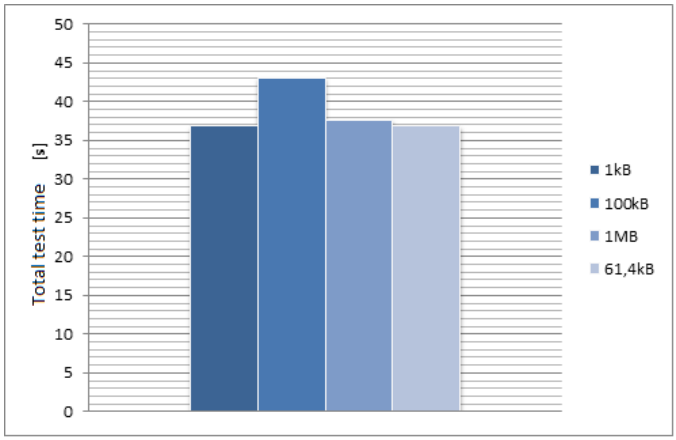

Fig. 17 Test times of Jotti malware scanner for different file sizes

Google SafeBrowsing and Microsoft SmartScreen scan speed is depended on client network connection quality and server load. The biggest part of its response time is network delay.

The biggest existing disadvantage of prototype is, that this kind of system is not effective for internal network attacks detection, or internal infections (for example infections from USB keys, etc.). Therefore help from local installed antivirus systems is needed.

ACKNOWLEDGMENT

Paper is the result of the Project implementation: Competency Centre for Knowledge technologies applied in Innovation of Production Systems in Industry and Services, ITMS: 26220220155, supported by the Research & Development Operational Programme funded by the ERDF.

REFERENCES

[1] Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." ACM SIGCOMM Computer Communication Review 34.2 (2004): 39-53.

[2] Polychronakis, Michalis - Mavrommatis, Panayiotis - Provos, Niels., Ghost turns Zombie: Exploring the Life Cycle of Web-based Malware , 2008, [online; accessed 25-April-2012], [Online], Available: http://static.usenix.org/event/leet08/tech/full_papers/polychronakis/polychronakis_html/

[3] Trend Micro., UNDERSTANDING USB MALWARE , 2010, [Online; Accessed 25-April-2012], [Online], Available: https://imperia.trendmicro-europe.com/imperia/md/content/us/trendwatch/researchandanalysis/56_understanding_usb_malware__april_19__2010_.pdf

[4] Igor Hák: Moderní počítačové viry. Bachelor Thesis. Hradec Králové: University of Hradec Králové, Faculty of informatics and management. 2010. 96 pages

[5] Google., What is Safe Browsing? , 2012, [Online; Accessed 25-April-2012], [Online], Available: http://code.google.com/intl/sk-SK/apis/safebrowsing/

[6] AVG., AVG ANTI-VIRUS NETWORK EDITION, 2009, [Online; Accessed 25-April-2012], [Online], Available: http://www.cirjconcepts.com/docs/AVG Anti-virus Network Edition Datasheet.pdf

[7] Eset., ESET Gateway Security for Linux / BSD / Solaris, 2012, [Online; Accessed 25-April-2012], [Online], Aviable:http://www.eset.com/us/business/products/gateway-linux/.

[8] (2008, August 30). magic(5) - Linux manual page. Retrieved from http://linux.die.net/man/5/magic

[9] (2008, October 9). file(1) - Linux manual page. Retrieved from http://linux.die.net/man/1/file

[10] Microsoft, SmartScreen Filter: frequently asked questions, 2012, [Online; Accessed 5-November-2012], [Online], Available: http://windows.microsoft.com/en-US/windows7/SmartScreen-Filter-frequently-asked-questions-IE9

[11] Microsoft, SmartScreen Filter, 2012, [Online; Accessed 5-November-2012], [Online], Available: http://windows.microsoft.com/en-US/internet-explorer/products/ie-9/features/smartscreen-filter

[12] Google, What is Safe Browsing?, 2012, [Online; Accessed 5-November-2012], [Online], Available: https://developers.google.com/safe-browsing/

[13] Google, Reference Guide, 2012, [Online; Accessed 5-November-2012], [Online], Available: https://developers.google.com/safe-browsing/reference

[14] Jotti, Service description, 2012, [Online; Accessed 5-November-2012], [Online], Available: http://virusscan.jotti.org/en-gb

[15] Microsoft, StringFromGUID2 function (COM), 2012, [Online; Accessed 5-November-2012], [Online], Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms683893(v=vs.85).aspx

[16] P. Leach and M. Mealling and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[17] http://lamer.s.cnl.sk/area51/diplomovka/patented-key.c

[18] Vijayrania, Anju, and Abdul Rahman. "Application of DHT Protocol in IP Cloaking." (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3.2 (2012): 3313-3317.

[19] Desti, Jude. A Framework for Smart Trusted Indicators for Browsers (STIB). Diss. Florida A&M University, 2010.

[20] Heravi, Mani. ANALYZING THE ACTIVEX CONTROL ON IE7 VS. IE8. Diss. CALIFORNIA STATE UNIVERSITY, 2012.