Trap E-mail Address for Combating E-mail Viruses

Boldizsár Bencsáth István Vajda
Laboratory of Cryptography and Systems Security (CrySyS)
Department of Telecommunications
Budapest University of Technology and Economics, Hungary
{bencsath, vajda}@crysys.hu

Abstract— **The biggest infection events show that the most dangerous viruses propagate via the Internet email systems. We propose a new solution to accelerate the identification of infected computers after an attack by e-mail viruses. Our approach uses trap e-mail addresses secured by a feasible cryptographic technique. The novelty of our work is the introduction of virtual e-mail addresses to identify virus sources. Furthermore we propose a technique for a secure file identifier that allows secure identification of a file for trusted parties, but makes it impossible for others. Our solution is viable: It has been successfully embedded in a localized version of a common desktop software.**

1. Introduction

Computer viruses have evolved in the last decade. In the beginning, viruses were written in assembly to be highly optimized, to save space and to achieve hardware-specific functions ([1]). Current computers have far more capacities and possibilities, thus currently there is no need to optimize the code of a virus. Viruses are written in easy-to-use development languages and systems (e.g. Visual Basic). New possibilities have also reached the propagation medium. While the first viruses used to modify executable files to carry their code, current viruses use the Internet as the propagation medium and many different data formats to carry the code.

The statistics of the biggest infections show, that the most potent viruses are using the e-mail system for propagation. Several recent viruses involving variants of W32/Sobig ([2]), W32/Bagle and W32/Netsky have achieved widespread propagation at rates significantly faster than any other viruses before. The overwhelming majority of viruses are spread via spoofed e-mail. Recent research tries to introduce Internet-wide systems to prevent e-mail address forging, but the proposals are far from general deployment ([13], [14], [15], [16], [17]). Meanwhile, our solution is scale-independent and promptly deployable.

As a countermeasure for e-mail viruses, multi-layer virus protection systems can be deployed. The first layer is to run virus scanner in the Internet e-mail gateways and servers. Another layer is to deploy antiviral products onto clients and internet hosts. This reduces the number of infections but still does not eliminate the problem itself. Infected computers send out thousands of infected messages to other hosts. What can be done about infected computers? The owner of the infected host is responsible for carrying out the disinfection. The idea is to inform the owner of the computer about the problem as early as possible. Most of the recent internet e-mail viruses spoof the sender's e-mail address. Therefore, the operator of an internet e-mail gateway is unable to recognize the owner of the infected computer. His only possibility is to recognize the IP address of the infected host, but the only way to use this information is generally to send the suspicious IP address to the ISP of the specific IP address range, and to ask the ISP to notify the owner of the host. As long as this method is not generally automatized and standardized, thousands of infected computers remain infected for weeks or months.

Our goal is to inform the owner of the computers in a faster way and therefore reduce the time required to cure an infected internet host. With this aim we propose the use of Trap E-mail Address (TEA).

The general idea of using traps against attackers from the Internet is known and used by honeypot systems ([3], [4]). Honeypots are intentionally weakly protected computers. Intrusion Detection Systems ([5], [6]) or proprietary software elements are used to detect the stolen (trap) information from the honeypot . This information is an input to a countermeasure system. Our solution provides a countermeasure against e-mail viruses without requiring special software components at the protected host. In addition, the owner of the protected host is assumed to be a common user without special skills as to the protection or administration of the system.

Trap e-mail addresses are standardly used to identify spammers and to prevent e-mail address harvesting (spam bait, [7], [8], [9]). These addresses are propagated through public web pages. In our solution the corresponding e-mail address is not public, and is unique to the protected host.

Our TEA is built in a file protected with a secure file identifier, which prevents the attacker from manipulation and successful identification of the trap address.

The structure of the paper is the following:

Section 2 gives the functional description of the proposed Trap E-mail Address. Section 3 describes the security considerations of the proposed solution and contains our proposed secure file identifier structure for deploying the Trap E-mail Address. Our prototype application and our selected cryptographic solution is described in Section 5. We also show a simple approach to distribute TEA along with software registration information. Summary is given in Section 6.

2. Trap E-mail Address

Trap E-mail Address is unique for each internet host served by our protection system. It is generated by the Trap E-mail Server (TES) and deployed by the owner of the computer. The TEA is used to identify infected computers by TES and to rapidly inform the owner of the host accordingly. Every e-mail containing a TEA as recipient address is routed to the TES

and never used as a regular e-mail address.

The steps of TEA protocol are the following.

2.1 Procedure of using Trap E-mail Address

(1) The Trap Email Server (TES) gains registration information about the Internet host and contacts the owner of the host. (This information contains the data about the owner and about the host.) The host information is used by the owner to identify the host, so generally it is the name of the computer or any other specific data.

(2) The TES stores the registration information in a long-term database and generates a unique Trap Email Address (TEA) for the host. The owner of the host stores this TEA on the host. This generally means storing the e-mail address in a specific file (Trap Email File - TEF), and storing the e-mail address in the address book of the user of the computer.

(3/1) When the host is infected by an internet virus, the virus looks for e-mail addresses on the host. The virus finds the TEA and tries to propagate itself to this e-mail address as well as others stored on the host.

(3/2a) Many viruses send infected e-mails to all the e-mail addresses found on the computer. During this propagation many viruses spoof the e-mail address of the sender to one of the e-mail addresses found on the host. A third party can receive an infected mail, that contains the TEA as sender.

(3/2b) The third party might send a bounce message (virus alert) message to the TEA with the information about the infection.

(3/3a) The TEA can also be used to detect information leakage. A malicious code can not only infect other computers from the host, but can steal sensitive information, e.g. e-mail addresses from the computer. In this case e.g. a spammer can grab the TEA e-mail address along with the other e-mail addresses found on the host.

(3/3b) Using the stolen data, a malicious party can send unsolicited email messages to the TEA that arrives to the TES. The stolen data can be used for other unlawful activities also.

(4) After the TES receives any email messages from anybody to the TEA, the server considers the TEA as a compromised address, as nobody else than the owner or user of the computer could disclose this e-mail address without a system compromise. According to the e-mails received to the TEA, the TES can inform the owner about the type of the danger. This can be an information about the infection (type of the virus, etc.) or the fact that the TEA has leaked.

(5) The owner of the computer is informed so the disinfection of the host may begin. After successful disinfection the old TEA should be replaced by a fresh one, as the old address could potentially be found on many infected internet hosts or in spammer's e-mail address database. The reinitialization process should find the old TEA on the host and replace it with a new one. (see Fig 1 also)

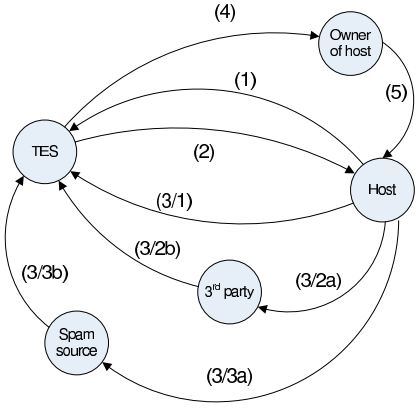

Fig. 1. Flowchart of TEA protocol

3. Security Considerations

Many internet viruses gain e-mail addresses from the files of the computer. These viruses scan files with common text file extensions for e-mail address-like character strings. We propose that the TEA should be stored in the address book of the host computer and also in specific file. The TEA should be stored in a file (the Trap E-mail File) that looks like a normal text file, in order to let the virus to scan and use the information found in this file.

3.1 TEF design objectives

Our design objectives in construction of the Trap Email File (TEF) were the following:

D1 TEF should not be distinguishable from common files, so the file name or the file content should not contain an identifier recognizable by a virus. The file name should not contain any specific information, thus the owner of the computer can rename it and store it anywhere in the file system. A special identifier in the file name can be very suspicious for the virus and it can simply discard the file. We don't expect that the owner remembers the place where the TEF is stored. TEF must follow some common file format, avoiding any uncommon, suspicious-looking parts.

D2 In order to be able to replace a compromised TEA, the TEF should be identifiable by our system. The TEF identification process should be fast: cpu intensive calculations should be avoided. The owner of the protected host has to be able to uniquely identify a TEF, while his knowledge about cryptographic materials should be kept at the needed minimum.

D3 The integrity of the TEF should be verifiable. If the virus can change the contents of the file, it can force our system to modify/delete important files on the system during the reinitialization process.

D4 TEA in itself cannot function as a secure identifier. If we trusted in TEA as such, the virus could deploy all e-mail addresses found on the host to different places to enforce our system to modify other files.

D**5** The attacker has to be prevented from fabricating valid TEFs, otherwise he could deteriorate the effectiveness of the trap. Only the TES should be able to produce valid TEFs.

D**6** The applied protection technology should match the resources of a standard PC. For instance, the computational complexity of cryptographic solutions must not impose unfeasible requirements on hosts. An uncomfortable solution might lead to the disregarding of the service by the users. The protection technology should apply only freely available elements.

In the following we detail how our solution meets the above requirements:

The TEF begins with a special security header described in Table I. The header contains security information listed in Table II and described subsequently. The binary data (field values) in the header is encoded via Base64 encoding [12] as described. The header is followed by the TEF message body. The body is user selectable and should not have any system-wide structures. (The intuition behind this is that independent users will not select correlated files for the body) The TEA is stored at random location within the message body

We do not set any constraints as to the name of the TEF. The secure identification header can be used to find the file on the system. The same method could also be used to organize other data files, like general documents or pictures. The security header therefore can be used for multiple purposes so a virus cannot discard files with a formatted security header while gaining e-mail addresses. The format of the message body is a simple text file. Text files are often opened by viruses to gather e-mail addresses from the system.

3.2 Security operations

Deployment of TEF on the protected host

The TEA is generated randomly by the TES and stored in its registration database. The TES uses a long term secret key to generate the header fields of the TEF. Using the long term secret key and the TEA, server TES calculates a session key (SK). From this session key a unique $FILEID$ is calculated by the TES. A $FILEAUTH$ code is generated from the SK and the message body of the TEF for the purpose of secure authentication of message integrity. The public $DOCMD$ field is calculated from the message body for simple file integrity checking operations. Using the hereby calculated header fields ($FILEID$, $FILEAUTH$, $DOCMD$) the server builds up the TEF and sends it to the protected host (H).

Cleaning up security attributes

Initialized by the arrival of an email the TES looks up the TEA in its registration database to identify the compromised status of the address. A compromised TEA is replaced the following way: The server marks the compromised TEA in the database. The server recalculates SK from the long term secret key, then it recalculates the $FILEID$ from SK and TEA. The server sends the $FILEID$ and SK (together with the TEF removal software) to the owner of the host.

AUTHheader	::=	'<!DOCTYPE' S 'TEXT' S 'Fieldname' Eq '"' Value '"' S '>'
S	::=	$(0x20\|0x09\|0x0D\|0x0A)+$
Eq	::=	S? '=' S?
Fieldname	::=	[a-zA-Z0-9]
Fieldvalue	::=	Base64 alphabet

TABLE I

TEF SECURE FILE IDENTIFIER HEADER DECLARATION

Field	Public	Description
STYPEVER	yes	the decimal value of version number of secure type identifier. currently $stypever = 010$.
FILEID	no	secret identifier key of the file
FILEAUTH	no	secret integrity protection field of the file
DOCMD	yes	the message diges of the message body

TABLE II

TEF HEADER FIELDS

The owner first removes all malicious codes from the protected host then runs the TEF removal software on the so disinfected host. The software searches for files by matching the $FILEID$ field. If a file with a $FILEID$ field identical to the received $FILEID$ is found, the server calculates the expected $FILEAUTH$ value from SK and the actual file body. The server removes the files with valid $FILEAUTH$ fields.

Reinitialization of TEF

The server generates a new TEA and stores in the registration database. Then it generates a new TEF according to the operations described above and sends it to the host. The owner selects the new location (file name and directory) of TEF and stores it.

In cryptographic terminology the protection applied to TEF via $FILEAUTH$ is secure message authentication. Therefore, without knowing the secret key the attacker is not able to produce authentic TEFs. The owner will be able to identify fabrication trials of the attacker, because he gets SK after an infection. The role of SK is to protect the long term secret key such a way that even the owner of the protected host has no access to it. The $FILEID$ makes it easier for to the owner to identify a candidate TEF file on the host after an infection.

Table II shows the header fields of the secure file identifier in the TEF.

4. COMMON PROBLEMS

Our proposed solution raises common security problems:

- An attacker can try to generate random e-mail messages and send them to the TES. With a distributive brute force attack he tries to find valid addresses. The TES does not return any information to the attacker about the validity

of the address, but it puts a label on the compromised attackers, therefore it is possible to deploy a Denial of Service (DoS) attack against our system. To prevent this, the TEA should be generated from a large keyspace. The size of the keyspace should be derived from the possibilities of an attacker (e.g. possible number of attacking host, query speed for one host, possibility of detection of an attacker).

- The number of possible generated e-mail addresses (the keyspace) is also important if the number of registrants is high. The generated e-mail address should look like a real, common e-mail address. The e-mail address can be generated as
  firstname.lastname<number>@subdomain.domain.tld,
  where domain.tld is selected from an available domain range (domain contributors) firstname, lastname and subdomain can be any valid wordlike string, and using a number field in email addresses is also common. This way the number of combinations of the fields can much greater than 10^{10}. This e-mail should be undistinguishable from valid real-life e-mail addresses. For this purpose the format of the generated e-mail addresses should be varied randomly. We believe the keyspace is large enough to maintain a workable system and can be enlarged by introducing new domain ranges to the system.
- It is possible to create a virus that searches for TEF-like files and distributes the identifiers found in the file across other files in the system (e.g. $FILEAUTH$, TEA, $FILEID$) and removes the original file. This way it is impossible to safely clean up the compromised TEA from the system, but using our secure file header, the reinitialization process won't harm other system files. Our system -of course- cannot protect from the damages caused by the virus itself.
- If the TEF is stored in a directory of the computer that is accessible from a network (e.g. shared network drive), then it is possible that other infected computers use the TEA and our notification is not proper. The owner of the computer is responsible for placing the TEF in an appropriate location or identifying the computer using the provided information. Of course an infected host in a network signals serious problems at the company, therefore checking all computers in the subnetwork is advised.
- The business model for maintaining such a system is also an important question. The system alone does not seem profitable. However the system can speed up the process of cleaning infected hosts on the network, therefore the reduced damages throughout the Internet can be accounted as the profit of the system. Supporting governments and service providers can help finance the system. While additional services (e.g. professional help for cleaning infected hosts) can be self-financing.

5. Our method in practice

A very important question of the trap e-mail system is the deployment of the TEA and TEF. It would be best if the owner of every Internet host would register at the TES voluntarily. Many users however don't know about our system so we decided to find a better way to distribute TEFs. Our selected procedure is to replace the registration certificate file of the Hungarian native OpenOffice.Org project with a TEF file.

The OpenOffice.Org is the leading open-source free (as in beer) international office suite. The Hungarian version of the software contains Hungarian localization files, a slightly modified spellchecker and other advancements.

Every user of the software can obtain a registration certificate file by registering the software through an on-line registration form. This registration helps the developers to gain information from the users. The registration certificate is a simple text file describing the most important parts of the OpenOffice.Org licence agreement. After clicking the "send registration information" button, the user gets a downloadable certification file generated by the server.

We modified the structure of the certification file to provide the functionality of a TEF. First, the server generates a TEA according to the registration information provided by the registrant. The server inserts the TEA into the certification message. The server then calculates and inserts the security identification header at the beginning of the file. Other users can get a TEF by a web-based registration on the TES. The direct registration provides a TEF as a TEA injected into a random text file.

A virus could avoid using any addresses found in the OpenOffice.Org registration files by analyzing their content, but currently no such specialized virus exists. It would be useful to provide other TEF variations (bearer message body) making the task of identifying the TEF more complicated.

5.1 The actual values of the TEF header fields

In our prototype system we used MD5 hash function to generate secure file identifier header fields.

Formally

$$FILEAUTH = MD5(SK_1, filebody, SK_2) \quad (1)$$

$$FILEID = MD5(SK) \quad (2)$$

$$SK = MD5(K_1, TEA, K_2) \quad (3)$$

where SK_1 and SK_2 are the 1st and the 2nd 8 bytes of SK, furthermore where K_1 and K_2 are the 1st and the 2nd halves of long term secret key K (see Figure 2 and Figure 3).

Secret key K is selected at random by the FES and is kept secret by TES ([10]). The TEA is selected once, and never recurs during the expected lifetime of the service. Calculation of SK follows the secure "sandwich" type keyed hashing technique ([11]). Because TEA is non-recurring, SK can be modeled as a randomly selected 16 byte value in the view of the attacker. Calculation of $FILEAUTH$ is similar, where the key is injected through SK.

We had to deal with the privacy problems of the trap e-mail system. The OpenOffice.Org registration data is also used by the TES. The registration of the OpenOffice.Org is totally

voluntary, anybody can use the software legally without any registration. We provided the appropriate licence information to be able to decide providing information for OpenOffice.Org development team and the TES as well. The TES should be a trusted third party and we expect that the users trust the joint effort of the OpenOffice.Org development team and our work.

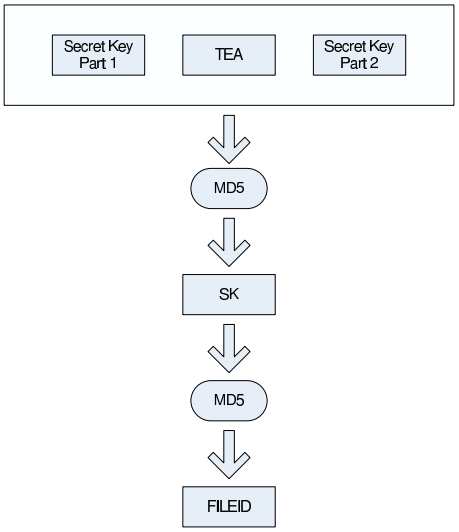

Fig. 2. Current method for generating $FILEID$

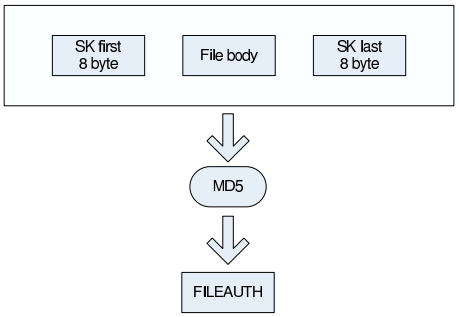

Fig. 3. Current method for generating $FILEAUTH$

5.2 Results of registration

In the first month after we introduced the TEF as the OpenOffice.Org registration certificate, our system deployed about 500 TEF certificates to registrants. 500 registered computers should be enough to gain useful information about the usability of the system. Up to today we only received 3 emails showing 2 compromised hosts. The low number of compromised addresses shows that most of the registrants care better of their computer than a general user. One of the events signalled the importance of proper registration information. The notified owner, an administrator of multiple computers told us that he do not properly remember the computer he used during registration and he also did not enter usable data about it. The proper description of the protected host during the registration is vital.

6. Summary and future work

In this paper, we introduced the idea of trap e-mail addresses to identify infected Internet hosts. Our approach does not require any modification in the fundamental internet infrastructure or the use of special software components as it exploits the behavior of the viruses to identify them. We analyzed the attacking environment and proposed efficient countermeasures. Our security solution is based on a secure file identifier header.

This header prevents the attacker from manipulation and successful identification of the trap, however it makes the task of identification and the resetting procedure easy for the owner of the protected host. This security identifier can also be used to securely identify other files on the system.

The trap e-mail address technique makes it easier to detect stolen e-mail addresses this way our solution can also be used to detect a special kind of information leakage.

We also presented our working system that solves the problem of deployment. The address is generated and deployed during a software registration process.

As future work, we intend to extend our system with other ideas and methods of the identification of infected computers. We believe that our technique is a worthy building block of an Internet-wide protection system.

References

[1] Eugene H. Spafford, "Computer Viruses — A Form of Artificial Life?", in Artificial Life II, ed. Langton, Taylor, Farmer and Rasmussen, Addison-Wesley 1992.
[2] Sobig.F Virus Fastest Spreading Ever. http://www.messagelabs.com/news/virusnews/detail/default.asp?contentItemId=528, August 20 2003.
[3] The Honeynet project, Know Your Enemy: Learning About Security Threats. Addison-Wesley. 2002.
[4] L. Spitzner, Honeypots: Tracking Hackers, Addison-Wesley, 2002.
[5] C. Endorf, E. Schultz, J. Mellander, Intrusion Detection, Osborne/McGraw-Hill, 2003.
[6] S. Staniford-Chen, B. Tung, P. Porras, C. Kahn, D. Schnackenberg, R. Feiertag and M. Stillma, "The Common Intrusion Detection Framework and Data Formats", 1998.
[7] S.Hird. Technical Solutions for Controlling Spam In the proceedings of AUUG2002, Melbourne, 4-6 September, 2002.
[8] Ronald F. Guilmette, wpoison – small CGI script to combat junk email, http://www.monkeys.com/wpoison/.
[9] Devin Carraway - Sugarplum automated spam-poisoner, http://www.devin.com/sugarplum/.
[10] B. Schneier, Applied Cryptography, John Wiley and Sons,Inc.
[11] G. Tsudik, "Message authentication with One-Way Hash Functions", ACM Computer Communications Review, v. 22, n. 5, 1992, pp. 29-38.
[12] RFC 1341. MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, 1992.
[13] Sender ID Framework. Microsoft, http://www.microsoft.com/senderid
[14] M. Abadi, A. Birrell, M. Burrows, F. Dabek, and T. Wobber, "Bankable Postage for Network Services", Proceedings of the 8th Asian Computing Science Conference, Mumbai, India, December 2003.
[15] The Penny Black Project. Microsoft, http://research.microsoft.com/research/sv/PennyBlack/
[16] J. Levine, A. DeKok, et al., "Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison", Internet Draft, IETF.
[17] SPF: Sender Policy Framework. http://spf.pobox.com/