SMTP sniffing for intrusion detection purposes

Maurizio Aiello*, David Avanzini*, Davide Chiarella†*, Gianluca Papaleo†*

*National Research Council, IEIIT, Genoa
†University of Genoa, Department of Computer and Information Sciences, Italy

Abstract. Internet e-mail has become one of the most important ways for people and enterprises to communicate with each other. However, this system, in some cases, is used for malicious purposes. A great problem is the worm and spam spreading. A smart e-mail content checking system can help to detect these kinds of threats. We propose a way to capture, store and display e-mails transactions through SMTP packet sniffing. We worked on pcap files dumped by a packet sniffer containing SMTP traffic packets of a real network. After reassembling the TCP streams and SMTP commands, we store the captured e-mails on a database: for privacy reasons, only e-mails headers are stored. Having a tool for clear understanding and monitoring SMTP transaction may help in manager security tasks.

Keywords: SMTP, E-mail, Intrusion Detection, Network Security, Worm Detection, sniffing.

1 Introduction

Nowadays e-mail has already changed people’s life and work habits thoroughly: in fact the majority of people use the e-mail system to share any type of information ranging from business purposes to illegal ones.

A pestering and energy draining problem for mail domain administrators is answering to users' requests regarding e-mail sent that never arrived to destination, or messages that he should have received but he didn't. However the main and most important problem related to e-mail utilization is virus and spam diffusion: software like Spamassassin [1] make an e-mail content checking in order to limit spam while Amavis [2] scans e-mail attachments for viruses using third-party virus scanners in order to detect viruses, but both have to be installed in every mail server in order to work. In all these cases the use of a packet sniffer can be a good solution to the problem; in fact it centralizes the work, enabling administrators to monitor all e-mail traffic of several servers with just one installation, making possible to build a gateway defense system and permitting to identify all the SMTP traffic, including e-mails sent to unknown servers or from infected hosts (trojans, viruses etc.). In fact, in some cases the misconfiguration of firewalls permit a normal user to install and use a SMTP server on his own machine without administrator authorization or to use a SMTP server to send mail that is not internal to the company. The disadvantage of our solution is that the use of ESMTP-TLS and PGP make the packet sniffer useless. While PGP is a server independent solution, ESMTP is bounded to both servers: source and destination have to use ESMTP with TLS enabled to permit an encrypted e-mail traffic; if one of the two servers doesn’t have TLS enabled, the communication is not encrypted.
The purpose of this paper is to present a smart solution to check e-mail for intrusion detection purposes. We propose a program based on SMTP flux reassembling\cite{3}\cite{9}\cite{10} that can be useful for e-mail auditing and intrusion detection purposes.

2 Packet sniffing

To allow our system to work we need information about packets within the network we are interested to monitor. We implemented a packet sniffing program to analyze the SMTP traffic using the Perl library Net::Pcap \cite{4}. The sniffer process begins by determining which interface to sniff on. The function lookupdev() is used to get the network interface and the function open_live() to set the interface to promiscuous mode for sniffing. Then, the function compile() and setfilter() are used to set a filter to get the packet; we can filter the packet sniffing on the port 25: this is the standard port in which mail servers are listening for connections. If we have multiple networks, we can furthermore filter our traffic, picking the network we are interested to. The function dump_open() allows the sniffer to dump the captured packets in a pcap file. Finally the function loop() allow the sniffer to start.

The pcap files are then passed to the reassembler: reassembler can be located in the same machine of the sniffer or, due to the SMTPSniffer modularity, in another host. To allow the reassembler to read pcap file in on-line mode (almost in real time), the sniffer creates pcap files every n packets sniffed on the network. The pcap files are overlapped to be sure not to lose packets between the different slices. The dimension of pcap files is set by the user. The architecture of the system is shown in Figure 1.

![System architecture](image-url)

Figure 1. System architecture

3 Reassembler

The next phase of our system is to take pcap file produced by the sniffer and reconstruct the whole e-mail sent by a client. To do this operations we use the Perl library Net::Analysis \cite{5}. To allow SMTPSniffer to work in online mode, a scheduler check every minute if there are new pcap files to check and then passes the new pcap files to the reassembler. The scheduler has been build, using Schedule::Cron \cite{6} library. First of all, for every packet captured we store in a file the source IP and port, destination IP and port, the timestamp and the data part of the packet. Timestamp...
represents the number of seconds between the present date and the Unix Epoch (January 1st, 1970).

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ipserver:port, Ipclient:port</td>
</tr>
</tbody>
</table>
| 192.168.0.1: 25 | 192.168.0.5: 3390 | 220 mail.com ESMTP Postfix
Helo client 250 server MAIL FROM:<sender@mail.com> 250 Ok
RCPT TO:<receiver@mail.com> 250 Ok
DATA
354 End data with <CR><LF> <CR><LF>
Message-ID: <44084FD4.6090309@mail.com>
Date: Fri, 03 Mar 2008 14:16:52 +0000
From: <sender@mail.com>
MIME-Version: 1.0
To: receiver@mail.com
Subject: prova
Content-Type: text/plain; charset="us-ascii";
Hi, how are you?
QUIT
220 Ok: queued as 045419981
221 Bye |

Figure 2. Flux hash table

Once we obtained all information interested about TCP packet, we have to reconstruct every SMTP session. To do this, we use two hash tables: the first hash table is called flux (Figure 2) and represents all the SMTP sessions.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Ipserver:port, Ipclient:port</td>
</tr>
<tr>
<td>2006/03/03 14:16:52</td>
<td>192.168.0.1: 25</td>
</tr>
</tbody>
</table>

Figure 3. Timeflux hash table

The second hash table is called timeflux (Figure 3), it contains a list of ordered timestamp referred for every smtp session. The second hash table is useful to reorder all the SMTP fluxes.. For a better view the timestamp format of Timeflux hash table in Figure 3 is YYYY/MM/DD hh:mm:ss.

During normal operations, if we find a TCP packet related to SMTP port 25, we check if the key in the flux already exists: in positive case we append the data part of the packet in the value of flux hash table, otherwise we create a new entry both in flux hash table and timeflux hash table.

Once we have information of all e-mails sent, the next step is to reorder all the fluxes related to their timestamp (this information is taken in timeflux hash table). For privacy reason all the parts about body message are cut.
4 Database features

Once we have reassembled the flux stream of our e-mails, the system dump the information about e-mails in a MySQL database. The info about e-mails stored in the database are: Timestamp (converted in a human readable format), mail client name (obtained by a reverse lookup), IP mail client, IP mail server, sender address, receiver address, SMTP code (e.g. 250, 450), e-mail size.

The usefulness to have all the mail stored in a database is the capability to search information we are interested in at any time through db-queries.

For example, you can find how and which e-mails have been sent by a user or to a particular user. Another choice is to see the whole e-mail traffic in a certain period of time or listing all the e-mails rejected by your e-mail server.

In Figure 4 we can see an example of e-mails captured and stored in our database.

<table>
<thead>
<tr>
<th>id</th>
<th>timestamp</th>
<th>nomenclature</th>
<th>ipclient</th>
<th>ipserver</th>
<th>fromfield</th>
<th>tofield</th>
<th>status</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2006-03-05 13:00:05</td>
<td>client1.net.it</td>
<td>192.168.0.3</td>
<td>10.0.0.1</td>
<td><a href="mailto:user@mail.it">user@mail.it</a></td>
<td><a href="mailto:receiver@mail.it">receiver@mail.it</a></td>
<td>250</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>2006-03-05 13:00:07</td>
<td>client2.net.it</td>
<td>192.168.0.4</td>
<td>10.0.0.1</td>
<td><a href="mailto:user2@mail.it">user2@mail.it</a></td>
<td><a href="mailto:receiver2@mail.it">receiver2@mail.it</a></td>
<td>250</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>2006-03-05 13:00:07</td>
<td>client3.net.it</td>
<td>192.168.0.3</td>
<td>10.0.0.1</td>
<td><a href="mailto:user@mail.it">user@mail.it</a></td>
<td><a href="mailto:receiver3@mail.it">receiver3@mail.it</a></td>
<td>250</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>2006-03-05 13:00:07</td>
<td>client4.net.it</td>
<td>192.168.0.3</td>
<td>10.0.0.1</td>
<td><a href="mailto:user@mail.it">user@mail.it</a></td>
<td><a href="mailto:receiver4@mail.it">receiver4@mail.it</a></td>
<td>450</td>
<td>234</td>
</tr>
</tbody>
</table>

Figure 4. Database example

5 Test scenario

We tested SMTPSniffer in a network segmented in more subnets with a total of about 400 hosts. We compare the result between reassembling the SMTP session analyzing a single big pcap file (off-line mode) and reassembling SMTP session analyzing more pcap files overlap slices (on-line mode) and we were able to reconstruct exactly all the e-mails. In Figure 5 we can see a scheme of our test scenario. The SMTPSniffer PC is attached to our network through two interfaces; one interface is connected to the hub to allow SMTPSniffer to capture all the traffic outgoing the network, (the interface is set in promiscuous mode), the other interface is connected to the switch to communicate with hosts in our network. The purpose to have an interface connected to the switch is to control in remote mode the SMTPSniffer pc. The hub is not strictly required, in fact you can connect the promiscuous interface directly to a switch mirror port.

Figure 5. Scenario schema
6 Security purposes

SMTPSniffer is, as already said, a powerful tool for a system administrator, because it allows a fast check of e-mail traffic in real time. This fact takes a lot of possible choices.

SMTPSniffer can be used for intrusion detection purposes, and in particular for indirect worm detection [7]. At the present, in fact, worms continue to improve in terms of their sophistication and detrimental effect and by exploiting the benefits of e-mail system they spread very widely and very fast, exhausting network resources. When a worm infects a host, it tries to send the greatest amount of e-mail in the shortest time interval: this behaviour can’t pass unnoticed because SMTPSniffer lists the e-mail activity and so you can try to discover why a host is sending a lot of e-mails.

It is possible to collect the data stored in the database to detect anomalies using statistical methods. Through DB-Query you can filter the data before analyzing them, getting a more efficient result and making possible an instant raw anomaly detection.

In fact worm spreading hurry produces a lot of e-mail rejected by the mail-server and if you filter the traffic through a query that consider only rejected e-mails (in database the field status must be 450 or 550) and then analyze the various peaks, there’s a lot of chance to identify worm activity on your network.

Moreover all these features permits you to anticipate the antivirus patch: in fact the speed at which viruses spread is faster than the speed to develop and distribute virus signatures for anti-virus protection.

7 Wormpoacher integration

In [7] we describe a worm detection technique and system. The program we are developing using these techniques is called Wormpoacher.

Wormpoacher use a tool for log mail analysis called LMA [8]. LMA is bounded to the specific mail server, if we install a mail server not supported by LMA we aren’t able to allow Wormpoacher work properly. Moreover many worm use its own SMTP engine to propagate and in this case, Wormpoacher isn’t able to analyze this type of traffic. To overcome the LMA limits, we can add SMTPSniffer feature in Wormpoacher. This operation is very easy to achieve, because Wormpoacher architecture is completely modular. Clearly, if we want to analyze communications between our hosts (internal communications) we have to place the sensor inside our internal network: the same concepts holds if we want LMA work on different servers.

8 Conclusion

In this paper we discussed a program useful to store and display all the e-mails sent in a local network. It can be used to prevent attack actions within local network or to investigate mail servers misconfigurations or mail traffic firewalls settings.

You can view all e-mails almost in real time through the db feature and, through various kind of queries, it is possible to perform different mail analysis.

SMTPSniffer can be integrated in Wormpoacher in order to perform worm detection and it can overcome the LMA constraints in order to have a total view of network mail configuration and activity.
Acknowledgments

This work was partially supported by National Research Council of Italy, University of Genoa and PRAI-FESR Programme, Innovative Actions of Liguria.

REFERENCES

[4]. Net::Pcap - Interface to pcap LBL packet capture library http://search.cpan.org/dist/Net-Pcap/Pcap.pm
[5]. Net::Analysis - Modules for analysing network traffic http://search.cpan.org/~worrall/Net-Analysis-0.06/lib/Net/Analysis.pm
[6]. Schedule::Cron - cron-like scheduler for Perl subroutines http://search.cpan.org/~roland/Schedule-Cron-0.97/Cron.pm
[7]. Maurizio Aiello, David Avanzini, Davide Chiarella, Gianluca Papaleo. Worm Detection Using E-mail Data Mining, 2006. PRISE 2006, Primo Workshop Italiano su PRIvacy e SEcurity.