Voice2Web security - Camnep integration

Technical report

Jan Staněk

Research & Development center(RDC) for Mobile Applications
Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic

March 2009

Abstract

This document describes the integration of CAMNEP [1], an intrusion detection system developed by the Department of Cybernetics ČVUT, into security solution in the Voice2Web project. Since CAMNEP is only an IDS providing reporting but no blocking capability, the main goal of this work is to create an automatic solution of CAMNEP reports processing which will do the blocking part.

Table of Contents

Acknowledgement

We would like to thank the CAMNEP developer team from the Department of Cybernetics ČVUT for the possibility to test CAMNEP in the V2W project and for their great cooperation and help with its set up and configuration.
We also wish to thank IBM Research for generous support of the project and Jan Kleindienst, Borivoj Tydlitat and Jan Sedivy for many thoughtful suggestions.
We also thank CESNET for the generous lending of project equipment and Vodafone Foundation Czech Republic for the kind student scholarship support.

Introduction

Voice2web project is one of RDC projects aiming on voice applications. It gives the user the opportunity to build and test his own voice applications in an easy way using VXMLIDE application. The security part of the Voice2Web project is to secure the servers used in the project, making them as invulnerable as possible. In the previous stage, documented in the first [2] and second [3] reports, were the servers secured and tested with some common simulated attacks. The results were somewhat satisfactory but it showed up that flooding DoS and DDoS attacks might be a problem. Therefore we decided to try to use the CAMNEP IDS which seems to have the ability to detect these types of attacks. Cooperating with the Department of Cybernetics ČVUT we deployed a CAMNEP monitor in the V2W project network and launched automatic reporting. Since the detection is not enough and we also need to block the attacker I created a set of scripts that provides automatic processing of CAMNEP reports and blocking of IP addresses marked in these reports as possible attackers.

Motivation

Security is one of the most important things in the field of publicly accessible applications. Because we plan to present V2W project openly and afterwards also to open the VXMLIDE application for public use, we need to have the servers as protected as possible. The previous results showed that our server is still quite vulnerable to flooding attacks e.g. that we are not able to detect general types of these attacks. We hope that CAMNEP will help us defend against these attacks. We also want to test the capabilities of CAMNEP used for the VoIP traffic since it was developed for general network traffic and nobody knows how it will behave for this specific traffic yet.

Short introduction of CAMNEP

As mentioned earlier, CAMNEP is an intrusion detection system (IDS) developed by the Department of Cybernetics ČVUT. Its purpose is to monitor network traffic and automatically report any detected unusual traffic. More about CAMNEP and its methods of detection can be found at [4]. For us the most important is that CAMNEP works on the basis of flow controlling. It does not inspect the payload of packets as Snort (an IDS already used in the V2W project, more information about it can be found in [2] and [3]) does but it aims on the general amount of traffic in the monitored network and therefore it should be able to detect flooding attacks because they have common pattern - continuous heavy load from the client (attacker) to the target (server).

Setting up CAMNEP

CAMNEP deployment and installation was done mainly by the people from the Department of Cybernetics ČVUT. After some meetings, we decided to use an automatic monitor on the router before the RDC network that will send the data about network traffic to the Department of Cybernetics. Their running instance of CAMNEP processes received data and evaluates them, sending us only reports containing alerts of unusual traffic. We chose this solution because it was the easiest and the most straightforward one and it is sufficient for the testing purpose. We were made sure that it will not be a problem to set up our own independent instance of CAMNEP if it proves usable in the V2W project environment. Nowadays the reports generated by CAMNEP are being sent to the bolekrep@gmail.com account. This is set up for testing purposes only because we do not need to react to the reports in real-time yet. Future plan is to set up a mail server in the V2W network that will deliver the incoming reports directly to the processing script.

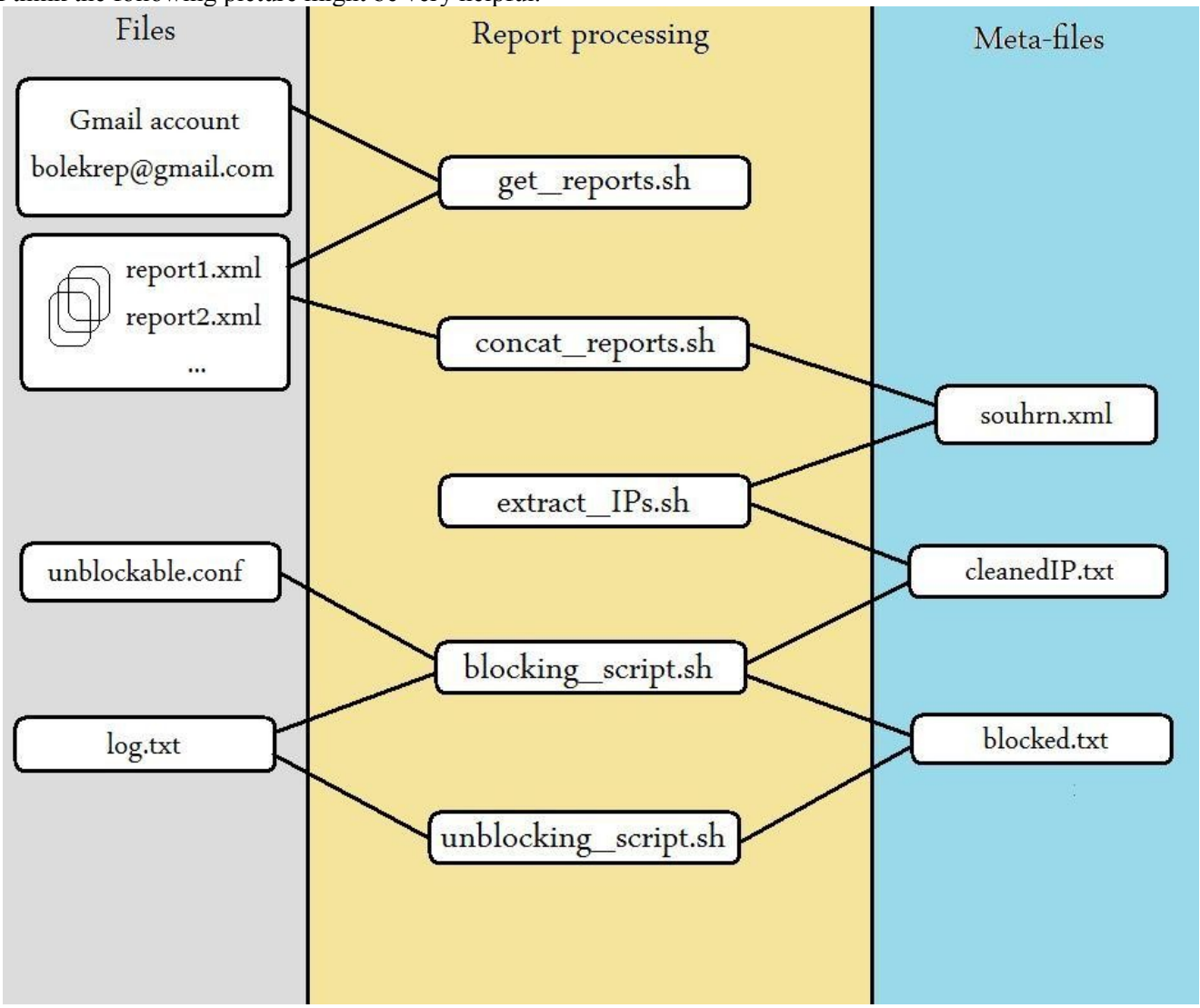
CAMNEP reports analysis

Because CAMNEP is designed for overall network monitoring and it has not been tested for VoIP traffic, the first step was to do an analysis of the reports that CAMNEP produces in the V2W environment. Sadly the results of this analysis show that CAMNEP is not able to work properly nowadays in our network. The reason is known and we have been warned of it before by the developer team from the Department of Cybernetics. It is because of the very low traffic load in our network. Since V2W project is still in the development phase and it has not been publicly opened yet, there is only a testing traffic about few kilobytes per hour and that is not enough for CAMNEP analytical algorithms to decide which traffic is potentially malicious and which is not. It might look like CAMNEP proved unusable and we cannot use it in the V2W project but we hope that it is a rather pessimistic view. Since we plan to open the VXMLIDE application for public use soon (possibly may 2009) we hope to get much more traffic when people are using our applications and that CAMNEP will be able to produce accurate reports then. Because of that I decided to create the scripts for automatic reports processing even though these cannot be used nowadays (since it might block anyone of our developers testing our applications) but it might be very useful in the future, if it shows up that our predictions were right.

Scripts for automatic reports analysis

The main part of this work was to produce an automatic solution for CAMNEP reports processing. I decided to make that solution as simple and as configurable as possible since it might need to be altered once we get the results based upon the traffic after VXMLIDE application is publicly opened . After some consideration I decided to create a set of shell scripts which can be run both independently and continuously. These scripts exchange necessary information through formatted meta-files (common files meant to be used by scripts only, not read nor altered by human) so one has a control of every step and tracing a bug is quite easy. The blocking is done via generating DROP rules for currently used firewall iptables. Every script has its own unique functionality and can be replaced by another script if the need occurs. For example if we decide to change the way the reports are delivered from the nowadays proprietary gmail account to another solution like our own mail server we can replace the current report getter script by a new one which will react real-time on the event

of mail delivery etc. Lets have a closer look at a way the scripts cooperate and what every one of them does. I think the following picture might be very helpful.



As can be seen, the report processing can be divided into a few independent phases

- get the mail(s) and extract the report(s)
- merge the reports into one file
- extract unique IP addresses of the supposed attackers
- block extracted IP addresses

Because we do not want to block the IP addresses forever since it might have been a false positive accusation, IP addresses are often dynamic nowadays etc. there is one more step which is not a part of report processing itself but is highly related

- unblock the blocked IP addresses after a defined period of time

The previous picture illustrated how the scripts cooperate through the files they are working with and which one is taking care of which processing phase (it is obvious from the name of each script). The following short articles will briefly explain the structure and functionality of each script along with a few words about why this solution was chosen, for better understanding consult the Appendixes which contain the source code of these scripts.

get_reports.sh – report getter

The first script is used to download a mail with an attached report from the mail account and extract the report

to the specified directory. Since it is not necessary to react real-time on the event of mail arrival nowadays I decided to use synchronous solution based on IMAP[5]. Once the script is run it contacts the mail account (which is bolekrep@gmail.com now) using IMAP and synchronizes local mail directory with the mail account. Simply put it downloads all new mails that arrived to the account after the last time this script run. To prevent the processing of old reports, every mail is transferred into another directory after the report is extracted from it so it is not in the directory with new mails next time the script runs. For this script I used two free third-party applications that can be easily run from within a shell script - offlineimap [6] and mpack [7].

concat_reports.sh – report aggregation

The functionality of this script is just to concatenate all the newly downloaded reports to one file. This step is not really necessary and it might be omitted but it is more comfortable to always work with one file instead of traversing varying number of separated files.

extract_IPs.sh – IP extractor

Once we have all reports in one file we need to extract the IP addresses of potential attackers so we can block them. That is not a big deal when we use a built-in shell command awk [8] and a suitable regular expression for the IP address format. Because the XML format of the report is well-structured and contains plenty of additional information, we might want to get some description for each attack and possibly divide the attackers into categories with varying period for which they will be blocked. That might be a bit difficult and a simple shell script would probably be not enough but there is no problem to simply replace this script with a more sophisticated one if the need occurs.

blocking_script.sh – IP blocker

This is one of two a little bit more complicated scripts along with the unblocking script. Once we have a list of IP addresses of potential attackers we want to block them for a specified period of time. The straightforward solution is to take the IP addresses one by one and evoke a firewall block rule (in our case it is an iptables DROP rule) for each one. That can be done using less then ten lines of code, but...but things obviously are not that easy:).
First problem is that we should store the time when the IP address was blocked along with the duration of block period somewhere so we are able to tell whether the blocking period has already run out or not when the unblocking script runs. I created a *blocked.txt* file in the *meta* directory with the following format

```
blocked_IP time_when_the_block_started_in_UNIXTIME_format block_period_in_seconds
```

An example record might look like the following.

```
82.208.56.89 1237910090 300
```

I have chosen the unix time format because it is easy to compare it with the actual system time and tell how long passed since the time when the block started.
Second problem is what to do with IP addresses that are already blocked and their blocking period has not run out yet? Evoking a new blocking rule does not look like a good idea because having more equal blocking rules makes the firewall rules a mess. I decided to traverse through all the already blocked IP addresses and if the address to be blocked is found as already blocked, the block will be extended for another blocking period. Technically this is done via altering the time when the block started for that IP address in the *blocked.txt* meta-file changing it to actual system time.
Third problem to be considered is a human-readable log with blocking events so the administrator has information about which addresses have been blocked by the script. To solve this one I created a *log.txt* file in the *camnep* directory where three types of events are stored in the following format

```
date time blocking IP X.X.X.X for Y minutes.
date time extending block for X.X.X.X for Y minutes.
date time unblocking X.X.X.X.
```

Fourth problem that occurred during the testing was that I accidentally blocked myself. That lead me to the idea that there should be a list of unblockable IP addresses that may never be blocked (DNS server for example). For

that reason I created an *unblockable.conf* file in the *camnep* directory that contains list of IP adresses in a simple format one address per line and implemented a part in the blocking script so that none of these addresses is ever blocked by the script even if it accidentally shows up in one of the CAMNEP reports.
Except these, I did not encounter any more pressing problems that have to be solved now and I think that this solution is usable. It might be considered to extend the blocks for some longer time periods for repetitively blocked IP addresses in the future so the attackers will be blocked for longer and longer periods etc. but the basic functionality is already done.
Note that because of nowadays inaccurate reports the blocking script has limited functionality – it only logs the potential blocking events but does not block the addresses. Once the reports become usable, the blocking functionality can be activated by simply removing the commentary sign # from the
`#iptables -A INPUT -s $1 -j DROP;`
line 54 of this script. Do not forget to uncomment the appropriate part in the unblocking script too.

unblocking_script.sh - IP unblocker

This script functionality is to unblock the IP addresses that has already been blocked for a specified blocking period. It is done via traversing through the list of blocked IP addresses written in the meta-file *blocked.txt* and unblocking the addresses that have a timestamp older than (actual system time - blocking period). Since the script processes every line of a *blocked.txt* file, I decided to use awk. When the script finds an IP address with a timestamp older than (actual system time - blocking period) it evokes an unblocking rule e.g. deletes the DROP rule for the specified IP address in the iptables rule set.
Because it will be unnecessary complicated to set a trigger for every block and run the unblocking script according to a timer in the precise times when some IP address should be unblocked I decided to simplify the process and run the unblocking script periodically. That is easily achieved using cron [9]. This solution has one aspect that can be seen as a flaw, but I find it rather useful. It is that the IP address is not blocked for precise block period but for a varying period instead, consisting of block period + time to the next scheduled unblocking script run. Because the unblocking script is run periodically in relatively short periods, it does not change the blocking time of an IP address much and can be even taken as a defense against timed attacks - the attacker cannot find out how long the typical block period is and run his attacks with according delay simply because there is no typical block period thanks to the described aspect.
Note that because of nowadays situation where the blocking script does not actually block the IP addresses extracted from the reports, this unblocking script does not delete any DROP rules either (since there are none). Once the fully functionality of the blocking script is activated, this unblocking script must be altered too, removing the commentary sign # from the
`#system(blocking)`
line 21 of this script.

process_reports.sh - all-covering script

Simple script that runs all previously described scripts except the unblocking script in a continuous row. That means one does not have to run the scripts manually, just run this all-covering script and the reports are automatically downloaded, concatenated, processed and marked attackers IP addresses are blocked.

Conclusion

CAMNEP IDS monitor was successfully deployed in the V2W network and automatic reporting has been set up. The reports from CAMNEP are not usable nowadays because it needs a bigger traffic load to work properly but we hope to fulfill this condition once V2W project is opened for public use. Scripts for automatic processing of CAMNEP reports have been prepared and tested so once the reports become accurate and usable to detect the attackers we have an automatic solution able to block their IP addresses contained in the reports.

Resources

[1] http://agents.felk.cvut.cz/projects/camnep/
[2] Staněk, J., Rudínský, J. - Secure Voice2Web servers from misuse, technical report, Prague, 2008
[3] Staněk, J.- Secure Voice2Web security testing, technical report, Prague, 2009
[4] http://www.muni.cz/research/publications/767621
[5] http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol
[6] http://software.complete.org/software/projects/show/offlineimap
[7] http://ftp.andrew.cmu.edu/pub/mpack/
[8] http://www.gnu.org/software/gawk/manual/gawk.html
[9] http://unixhelp.ed.ac.uk/CGI/man-cgi?cron+8

Appendixes

get_reports.sh

```
#!/bin/sh
#
# This script downloads new mails from the testing mailbox bolekrep@gmail.com
# and extracts the reports from them, storing these reports to $REPORTDIR/reports.
# Processed mails are stored to $PROCESSEDMAILDIR afterwards.
#
# Note that the functionality of this script might be affected
# 1/ by firewall - imap traffic allowance must be properly configured
# 2/ by offlineimap configuration - the .offlineimaprc file in the /root/ directory

# directory where new mails with attached reports are stored
NEWMAILDIR=/home/kelanth/mail/INBOX/new/*
# directory where mails are stored after the report extraction
PROCESSEDMAILDIR=/home/kelanth/mail/INBOX/cur/
# working directory where the reports/ directory to store xml reports is situated
REPORTDIR=/root/camnep/

# download new mails and extract the reports, if there are any
offlineimap && munpack -C $REPORTDIR $NEWMAILDIR

# move already processed mails to another directory so they will not be processed
when the script runs again
mv $NEWMAILDIR $PROCESSEDMAILDIR
```

concat_reports.sh

```
#!/bin/sh
#
# This script concatenates all reports from the $REPORTS directory into
# one file $SOUHRN so they can be easily processed later.

# path to the file where concatenated reports will be stored
SOUHRN="/root/camnep/meta/souhrn.xml"
# path to the directory containing extracted reports
REPORTS="/root/camnep/reports/*"


# delete the old file since it might contain old reports
rm -rf $SOUHRN
```

```
# concat all the reports to one file
for file in $REPORTS; do
        cat $file >> $SOUHRN
done
```

extract_IPs.sh

```
#!/bin/sh
#
# This script extracts all IP adresses from a $SOUHRN file.
# Only unique IPs are contained in the resulting list $IPLIST,
# duplicates are automatically merged.

# the file to be processed, contains concatenated reports
SOUHRN="/root/camnep/meta/souhrn.xml"
# resulting list with extracted IPs
IPLIST="/root/camnep/meta/cleanedIP.txt"

# detects IP adresses on each line of $SOUHRN file and stores them to buffer,
# prints deduplicated IPs to the $IPLIST file
awk '
BEGIN{
    r = "[0-9][0-9]?[0-9]?"
    r = r "\\." r "\\." r "\\." r
}
match ($0, r){
    a[substr($0, RSTART, RLENGTH)] = 1
}
END{
    for (x in a)
        print x
}
' $SOUHRN > $IPLIST
```

blocking_script.sh

```
#!/bin/sh
#
# This script processes a list of IPs $SRC_IP and generates a firewall
# blocking rule for each IP for a specified time. It also generates a
# human-readable log file $LOG_BLOCKED containing all generated blocking
# events.

# list of IPs to be blocked by a fw rule
SRC_IP="/root/camnep/meta/cleanedIP.txt"
# list of unblockable IPs that cannot be blocked by this script
UNBLOCKABLE="/root/camnep/unblockable.conf"
# log file for blocking events
LOG_BLOCKED="/root/camnep/log.txt"
# meta-file generated for the unblocking script
BLOCK_LIST="/root/camnep/meta/blocked.txt"
# temp file
TEMP_FILE="/root/camnep/meta/tmp.txt"
# the default period used for each block
DURATION_MINUTES=5

# tries whether the IP is blockable e.g. is not contained in the $UNBLOCKABLE
is_blockable()
```

```
{
        for j in $(cat $UNBLOCKABLE); do
                if [ $1 == $j ] ; then
                        return 0
                fi
        done;
        return 1;
}

# looks up whether the IP is already blocked
is_already_blocked()
{
        for k in $(cat $BLOCK_LIST); do
                if [ $1 == $k ] ; then
                        return 1
                fi
        done;
        return 0;
}

# blocks the IP and logs the blocking event
block_ip()
{
        is_already_blocked $1
        if [ $? -eq 1 ] ; then
                BLOCK_UTIME=`date +"%s"`
                BLOCK_TIME=`date +"%D %H:%M:%S"`
                sed "s/$1 [0-9]*/$1 $BLOCK_UTIME/" $BLOCK_LIST > $TEMP_FILE
                mv $TEMP_FILE $BLOCK_LIST
                echo "$BLOCK_TIME extending block for $1 for $DURATION_MINUTES
minutes." >> $LOG_BLOCKED
        else
                #iptables -A INPUT -s $1 -j DROP;
                BLOCK_TIME=`date +"%D %H:%M:%S"`
                BLOCK_UTIME=`date +"%s"`
                BLOCK_DUR=`expr $DURATION_MINUTES '*' 60`
                echo "$BLOCK_TIME blocking IP $1 for $DURATION_MINUTES minutes." >>
$LOG_BLOCKED;
                echo "$1 $BLOCK_UTIME $BLOCK_DUR" >> $BLOCK_LIST;
        fi
}

# main blocking loop processes each IP in the $SRC_IP file
for i in $(cat $SRC_IP); do
        is_blockable $i
        if [ $? -eq 1 ]
        then block_ip $i
        fi
done;
```

unblocking_script.sh

```
#!/bin/sh
#
# This script traverses list of actually blocked IPs and unblocks the IPs
# which block already exceeded the default block period.

# list of blocked IPs
BLOCK_LIST="/root/camnep/meta/blocked.txt"
```

```
# temporary list of blocked IPs used during script run
SWITCH_LIST="/root/camnep/meta/switch.txt"

# Warning - Take extreme care if changing this script, using external
# variables fail when inserted into (g)awk part of script. Also the time
# constants should not be imported from outside the awk part.

gawk '
function unblock(ipaddr)
{
        block_time="'"`date +"%D %H:%M:%S"`"'"
        print block_time " unblocking " ipaddr"." >> "/root/camnep/log.txt"
        blocking = "iptables -D INPUT -s " ipaddr " -j DROP"
        #system(blocking)
}
{
x="'"`date +%s`"'"
y=x-$3
if(y < $2)
        print $0
else
        unblock($1)
}' $BLOCK_LIST > $SWITCH_LIST

# finally rewrite the old blocked list with the newly generated one
mv $SWITCH_LIST $BLOCK_LIST
```

process_reports.sh

```
#!/bin/sh
#
# This script automatically downloads and processes CAMNEP reports,
# blocking the IP addresses marked by CAMNEP as possible attackers.

./get_reports.sh
./concat_reports.sh
./extract_IPs.sh
./blocking_script.sh
```