Web-based IP telephony Penetration System Evaluating Level of Protection from Attacks and Threats

MIROSLAV VOZNAK, FILIP REZAC
Department of Telecommunications
VSB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava Poruba
CZECH REPUBLIC
miroslav.voznak@vsb.cz, filip.rezac@vsb.cz

Abstract: - This article deals with detection of threats in IP telephony, the authors developed a penetration testing system that is able to check up the level of protection from security threats in IP telephony. SIP is being widely used in building VoIP networks. Unlike the traditional telephone networks VoIP networks does not have a closed communication which makes communication medium vulnerable to all kinds of attacks from the in truders. The SIP server is a key component of VoIP infrastructure and often becomes the aim of attacks and providers have to ensure the appropriate level of security. We have developed web-based penetration system which is able to check the SIP server if can face to the most common attacks. The developed application is distributed as an open-source and is equipped with four modules. The result is reported to the particular e-mail and information supplemented to the report should help to improve the overall protection of the SIP server. The developed application represents effective tool which is able to point out the weaknesses of the tested system.

Key-Words: - IP telephony, UDP flood, SIP server, Penetration test, Flood attack, SIPVicious, Vulnerability

1 Introduction

System designed to test and monitor networks or other components are quite wide-spread these days. Examples of the principle ones are Nessus, Retina, Snort and other. The majority of these systems allows for testing the whole network infrastructures and protocols used for communication between components. None of these solutions, however, enables a complex testing of VoIP infrastructure and SIP servers which are the key and most vulnerable component of the network. The system we developed, under a working title SPT (SIP Penetration Testing), was designed as a penetration tests simulator for SIP servers. Based on the analysis of intersections, the person who initiated the testing ("the tester") receives feedback in the form of test results, as well as recommendations how to mitigate potential security risks that were discovered. The advantage of this solution is that the system simulates real attacks from the external network, i.e. the system does not need to be placed in the same network as the target component DUT (Device under Test). This is frequently one of prerequisites to be able to use other testing tools. The SPT system was implemented as a web application accessible through a standard web browser and therefore independent on the operation system's platform. As the solution was developed as a part of the research intent of the CESNET association, this system will also be incorporated into its network and will be accessible after signing in using the SSO (Single Sign-On) service - Shibboleth. This should also prevent the system being used for other than testing purposes. Once signed in, the tester enters the required data into a web form and chooses tests to be run. The output of the application once the tests have been completed is an e-mail report to the tester. This paper contains the results of the tests; and in case some penetrations were successful it also contains recommendations and measures to mitigate such attacks in the future. Fig. 1 illustrates the concept of the SPT system. The following chapter describes individual testing methods in detail, their implementation, algorithms used and the impact on the target SIP server.

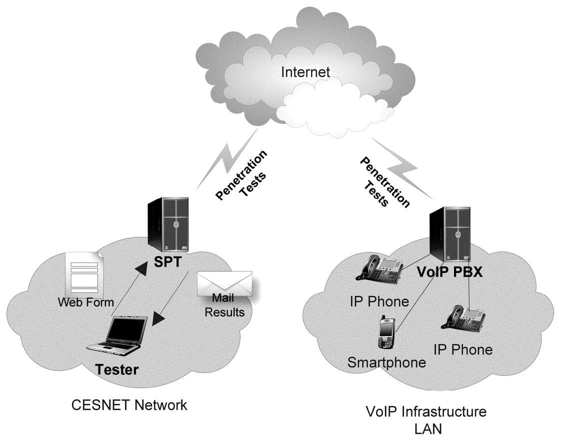


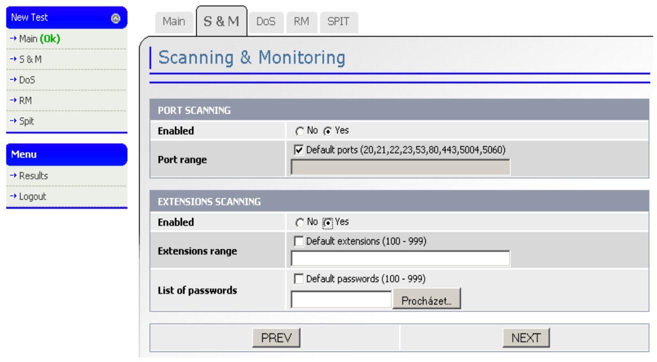
Fig. 1. SIP Penetration Tests System Scheme.

2 Methods

Although the system is primarily designed for penetration tests on SIP servers, in reality it can perform full-scale attacks on a particular component and provide feedback on it to the tester. Thus, it is necessary to ensure that the developed system cannot be abused by a third party. The system was designed as a LAMP (Linux, Apache, MySQL, PHP) server and its complete administration including the installation is carried out via a web interface. For reasons stated above, the system will be incorporated into the CESNET's network and will only be accessible to authorized persons once they pass through the authentication. Once the tester fills in the IP address or domain name of the central SIP server and the email address to which the test results will be sent to. Using checkboxes, the tester may define the range of the modules offered for testing. Individual modules are described below in detail.

2.1 Scanning and Monitoring Module

In order to be able to carry out an efficient and precise attack on a SIP server, the potential attacker needs to find out the most information about a particular component [1], [2]. This is why we first developed a Scanning and Monitoring ("S&M") module for the SPT system, which is used to test the security of the central against attacks aimed at obtaining information by means of common and available tools (Fig. 2).



Scanning & Monitoring

PORT SCANNING	
Enabled	No [✓ Yes]
Port range	[✓] Default ports (20,21,22,23,53,80,443,5004,5060)

EXTENSIONS SCANNING	
Enabled	No [✓ Yes]
Extensions range	☐ Default extensions (100 - 999)
List of passwords	☐ Default passwords (100 - 999)

Fig. 2. SPT System – S&M Module.

These tools include for instance *Nmap* or ever more popular *SIPVicious*. SPT system also uses these testing tools. By means of these tools, it is possible to obtain a list of listening ports or a list of user accounts created on the central concerned from an unsecured server [3]. Where the server is not secured sufficiently, they can obtain even the most important, that is passwords to individual accounts. If the tester ticks the test to be carried out, the *Nmap* application is used first to establish open ports. Given the time requirements of the [s] test, the testing is by default restricted only to several most frequently used ports. Using the web form, the tester can set the range of the tested ports. However the total time set for testing using *Nmap* is 1800s (30 minutes). The list of available ports is subsequently included in the assessment report together with recommendations how to minimise such ports' scanning. Another test which the SPT system can carry out aims at establishing whether SIP server's security allows for obtaining a list of user accounts. For this purpose, *SIPVicious* is used. By sending out OPTION and ACK requests, the application detects what accounts are defined on the SIP server. By default, the system tries the 100-999 range of accounts.

Again, the tester may define own range of tested numbers E_{nr} or import a text file containing strings of alpha-numeric characters or words E_{dr} . Time required to check and create a list of T_e [s] accounts can be expressed by equation (1) where 0.02603 is a time constant obtained by repetitive measurements on a sample of 1000 potential accounts on different target SIP servers.

$$T_e = (E_{nr} + E_{dr}) \cdot c \tag{1}$$

Number of valid accounts E_{valid} is derived from equation (2) where $E_{invalid}$ is the number of accounts that have been reviewed by the system but not defined on the SIP server.

$$E_{valid} = (E_{nr} + E_{dr}) - E_{invalid} \tag{2}$$

Once the system has tested security of the SIP server against detecting accounts, possibility to detect passwords for individual accounts is tested. Again, this testing is carried out by *SIPVicious*. Using a pre-defined range of possible numeric passwords P_{nr} or an imported text file with alpha-numeric characters or words P_{dr} , it obtains a list of passwords for individual accounts. Time requirements on this test are expressed by the following equation (3).

$$T_p = \left[E_{valid} \cdot (P_{nr} + P_{dr})\right] \cdot c \tag{3}$$

$$T_{sm} = T_e + T_p + T_n \tag{4}$$

Now we can determine the estimated time required to carry out the complete S&M test T_{sm} (4). Using the module, we can verify whether the target SIP server is sufficiently secured against such scanning and monitoring attacks [9].

The followin Fig. 3. shows a structure of PHP code which enables to perform the above described tests. First of all, the incompleted subtests corresponding to only S&M module (type 2) are selected from database. Variable *$RRow* provides serialized information which were carried back from database. These information contain what is enabled and disabled in prepared tests and information about target (Device under test), it represents IP address or domain name of the tested SIP server. Variable *$Data* contains the deserialized information field.

```
$Result = mysql_query("SELECT id,rid
FROM t_test WHERE type='2' AND
value='0'");
for($i=0;$i<mysql_num_rows($Result);$i
++)
 {
  $Row = mysql_fetch_row($Result);

$FResult = mysql_query(" SELECT id,rid
, value FROM t_test WHERE
rid='".$Row[1]."' AND type='1'");
  $FRow = mysql_fetch_row($FResult);
  if($FRow[2] == -1 || $FRow[2] == 1)
   {
    $RResult = mysql_query("SELECT
data FROM t_raw WHERE
id='".$Row[1]."'");
    $RRow = mysql_fetch_row($RResult);
    $Data = unserialize($RRow[0]);
    $MData = $Data['Sam'];
    mysql_query("UPDATE t_test SET
start=NOW(),value=10 WHERE id='".$Row[
0]."'");
    $Res = "";
```

Fig. 3. Výběr informací s databáze pro S&M testy.

As we have already stated in texts above, S&M tests are performed by two open-source application – *Nmap* and *SIPVicious*. This situation is depicted on Fig. 4, this figure represents the way of realization the tests based on two tools mentioned above. The commnad *shell_exec* calls particular string for nmap, in dependence on fact if the testing of default ports was selected or own range was submitted. The final result is stored into variable *$Res*. If the detection of SIP aacounts and passwords is allowed then the application SIPVicious is launched. For this case, a script *svmap.py* in Phyton was prepared. The script is able to recognize which distribution and service is applied on SIP server. The result from script is then processed by command *preg_split* that is used for parsing and final values are stored in variables *$SvMapRes* and *$SvMapResL.* System continues with testing on SIP server and for this purpose a script *svwar.py* is applied. The retrieved valid accounts are checked with next script *swcrack.py* which tries to find out the passwords for individual accounts, the brute-force attackt was adopted and it takes the time expressed in relations (1) – (4).

```
if($MData['PortEnable']==true)
   {
    if($MData['PortDefault'])
     {
$Res .= shell_exec("nmap ".$Data['Main
']['Ip']." -PN -sS -sU -
p 20,21,22,23,53,80,443,5060 -A --
host_timeout 30m");
     }
    else
     {
$Res .= shell_exec("nmap ".$Data['Main
']['Ip']." -PN -sS -sU -p
".$MData['PortRange']." -A --
host_timeout 30m");
     }
   }
    $Res = str_replace("'","''",$Res);
  if($MData['ExtensionEnable']==true)
   {
$SvMapRes = shell_exec("/server/sipvic
ious/svmap.py ".$Data['Main']['Ip']);
$SvMapResL =
preg_split("|\n|",$SvMapRes);
$SvMapResD =
preg_split("/\|/",$SvMapResL[2]);
$SvMapDevice = trim($SvMapResD[1]);
$SvMapAgent = trim($SvMapResD[2]);
$SvMapFinger = trim($SvMapResD[3]);

if($MData['ExtensionDefault'] == true)
 {
$SvWarRes = shell_exec("/server/sipvic
ious/svwar.py -e100-999
".$Data['Main']['Ip']);
 }
else
 {
$SvWarRes = shell_exec("/server/sipvic
ious/svwar.py -
e".$MData['ExtensionRange']."
".$Data['Main']['Ip']);
 }

$SvWarResL =
preg_split("|\n|",$SvWarRes);

for($w=2;$w<sizeof($SvWarResL);$w++)
 {
```

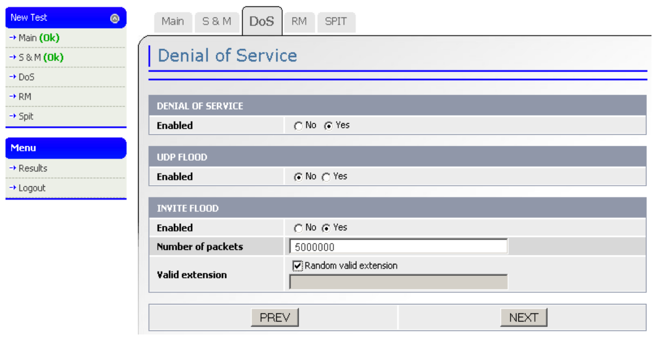
```
$SvWarResD[$w] = preg_split("/\|/",$Sv
WarResL[$w]);
if(strlen(trim($SvWarResD[$w][1]))>0)
$SvExtensions[trim($SvWarResD[$w][1])]
 = trim($SvWarResD[$w][2]);
 }

if($MData['PaswDefault'] == true)
 {
$SvCrackRes = shell_exec("/server/sipv
icious/svcrack.py -u"..“ -r100-
999 ".$Data ['Main']
['Ip']);
 }
```

Fig. 4. Volání aplikací Nmap a SIPVicious.

2.2 Denial of Service Module

One of the most frequently occurring attacks is DoS (Denial of Service). In reality, it consists of several attacks with the same characteristic feature – to lock up or restrict the availability of the attacked service so that it does not function properly. Several types of DoSs can be used to achieve this; our system tests the SIP server using the most frequently used one, Flood DoS. The principle of the attack is to send a large volume of adjusted or otherwise deformed packets to the target component so that it is unable to provide its core services [5]-[8]. As a result of the attack, CPU load increases and most of the available bandwidth is consumed, resulting in the SIP server being unable to service regular calls, or only a minimum amount of them. To generate Flood DoS, the SPT system uses two applications: *udpflood* and *inviteflood* [12]. When using *udpflood*, the system generates UDP packets of 1400 bytes which are directed at SIP default port 5060 of the target SIP server. The tester defines the number of generated packets and the system tests whether the packets arrived at the SIP server and whether they cause some restriction of the service availability, see Fig. 5.



Denial of Service

DENIAL OF SERVICE
Enabled: No [✓ Yes]

UDP FLOOD
Enabled: [✓ No] Yes

INVITE FLOOD
Enabled: No [✓ Yes]
Number of packets: 5000000
Valid extension: [✓ Random valid extension]

Fig. 5. SPT System - DoS Module

Moreover, we had to solve an issue of remote detection to determine of the test on SIP server was successful or not. The DoS attacks affect the overall SIP server performance and impairment of computing time can be detected but how to remotely recognize and determine that CPU load was increased? Following example of PHP script, depicted on Fig. 6, represents a solution utilizing ICMP message *ping*. At first, arithmetic average is calculated and then DoS test si launched. The tool *udpflood* is running and ping response are evaluated by the PHP script, the results are continuously compared with previous results and if the computed value is 150 times higher then previous then we consider DoS to be successful. If the *ping* is not allowed on tested SIP server then we are not able recognize if *udpflood* attack was successful or not.

```
$Res = "dos ok.";

if($MData['Enable'] == true && $MData[
'UdpEnable'] == true)
   {
    $ping = new Net_Ping;
    $fmin = 9999;
    $fmax = 0;
    $favg = 0;
    $favgc = 0;
    $pingd = 0;
    for($f=0;$f<20;$f++)
     {
      $ping-
>ping($Data['Main']['Ip'],1,6);
      if ($ping->time)
       {
        if($fmin>$ping-
>time) $fmin=$ping->time;
        if($fmax<$ping-
>time) $fmax=$ping->time;
        $favg+=$ping->time;
        $favgc++;
       }
      else
       {
        $pingd++;
       }
     }
    if($favgc>0)
     {
      $favg = $favg/$favgc;
     }

system("nohup /server/udpflood/udpfloo
d 195.113.113.137 ".$Data['Main']['Ip'
]." 5060 5060 ".$MData['UdpPackets']."
> /dev/null &");
```

```
usleep(10000);

    $f2min = 9999;
    $f2max = 0;
    $f2avg = 0;
    $f2avgc = 0;
    $p2ingd = 0;
    for($f=0;$f<20;$f++)
     {
      $ping-
>ping($Data['Main']['Ip'],1,6);
      if ($ping->time)
       {
        if($f2min>$ping-
>time) $f2min=$ping->time;
        if($f2max<$ping-
>time) $f2max=$ping->time;
        $f2avg+=$ping->time;
        $f2avgc++;
        }
      else
       {
        $p2ingd++;
       }
     }
    if($f2avgc>0)
     {
      $f2avg = $f2avg/$f2avgc;
     }

if($pingd>0 || $p2ingd>0)
 {
  $DoSUDPOk = false;
 }
else
 {
  if(($favg*150)<$f2avg)
   {
    $DoSUDPOk = true;
   }
  else
   {
    $DoSUDPOk = false;
   }
 }
```

Fig. 6. Detection and udpflood start-up .

Since we know the packet's size and therefore also the size of the Ethernet framework Fs_{udp}, we can, based on the number of generated packets P_n and the bandwidth provided B_w , determine time T_{udp} [s] required to carry out the test (5).

$$T_{udp} = (Fs_{udp} \cdot P_n) / B_w \tag{5}$$

Table 1. Overview of time required for different numbers of generated packets P_n and different bandwidth B_w .

TABLE I
UDPFLOOD ATTACK TIME DURATION WITH DIFFERENT BANDWIDTH AND NUMBER OF GENERATED PACKETS

Number of Packets - P_n	Bandwidth [Mbps] and the Attack Time T_{udp} [s]			
	10	25	50	100
100 000	113,12	45,25	22,63	11,31
200 000	226,24	90,50	45,26	22,62
300 000	339,36	135,75	67,89	33,93
400 000	452,48	181	90,52	45,24
500 000	565,60	226,25	113,15	56,55
600 000	678,72	271,5	135,78	67,86
700 000	791,84	316,75	158,41	79,17
800 000	904,96	362	181,04	90,48
900 000	1018,08	407,25	203,67	101,79
1 000 000	1131,20	452,5	226,3	113,1

When the other application, *inviteflood*, is used for testing, the system generates INVITE requests at the SIP server which are directed at an existing account. This method is very successful as most of today's SIP servers require an authentication for INVITE requests. As the INVITE requests generated by our system do not contain any authentication string, the SIP server returns SIP answer 407 Proxy Authentication Required. With the large volume of incoming requests, the load of SIP server's CPU increases. The tester can set the value of a valid account in the system manually, or it can be randomly selected from the previously obtained list of valid accounts E_{valid} . As in the previous case, we can, based on the number of generated packets P_n and the bandwidth provided B_w, determine time T_{invite} [s] required to carry out the test (6).

$$T_{invite} = (Fs_{invite} \cdot P_n) / B_w \tag{6}$$

Fig. 7 illustrates the impact of the change in bandwidth on CPU load when simulating an *udpflood* attack. The chart also clearly shows resistance of the two popular open-source SIP servers, Asterisk PBX and OpenSIPS, to UDP Flood DoS attacks. Both centrals have been installed on the same HW of Dell PowerEdge R510 server to eliminate any potential difference in computational performance. To change bandwidths, we used HW emulator of the Simena networks. CPU load on individual centrals was measured by means of *dstat* [10]. The chart shows that OpenSIPS is many times more resistant to UDP DoS attacks than Asterisk. Total time required to carry out DoS tests T_{dos} is determined as follows (7).

$$T_{dos} = T_{udp} + T_{invite} \qquad (7)$$

Results and success rate of DoS tests carried out are included in the report for the tester.

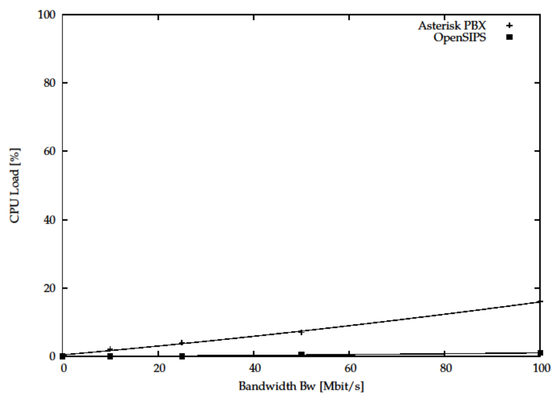
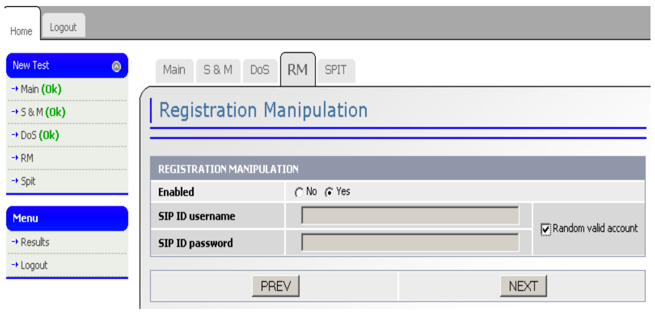


Fig. 7. Impact of change in bandwidth on CPU load in case of *udpflood* attack.

2.3 Registration and Manipulation Module

Once the potential perpetrator obtains information about existing accounts, he can manipulate these accounts quite easily. The SPT system we developed can also test SIP servers' security, i.e. measures against manipulating the registration, see Fig. 8.



Registration Manipulation

REGISTRATION MANIPULATION		
Enabled	No [✓ Yes]	
SIP ID username		[✓ Random valid account]
SIP ID password		

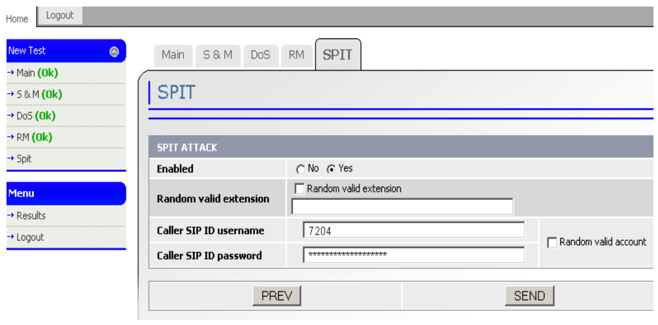
Fig. 8. SPT System - RM Module.

To carry out this test, the system uses *reghijacker* which substitutes the legitimate account registration with a fake, non-existing one. This type of attack can easily be expanded to a so called MITM, Man-in-the-Middle. In this attack, a non-existent user is substituted by a valid SIP registration and all incoming signaling and media to the legitimate registration will be re-directed to the newly created registration. In this case, the tester needs to define the value of the SIP account which is to be stolen in the system and where authentication of REGISTER request is allowed, also a password to this account. Where the tester fails to define these values, the system automatically assigns an account and its password from the list created while scanning and monitoring the central. Time required to carry out the test is insignificant compared to operational times of other modules.

2.4 Spam over Internet Telephony Module

Today, one of the most popular attacks on the Internet is spam. It is estimated that Spams account for 80 - 90% of total attacks on the Internet. Security experts predict that Spam over Internet Telephony (SPIT) will be a major threat in the future. The level of annoyance is even greater than with classical spam. Out team in CESNET had developed *SPITFILE* [12] which served as a testing tool while developing security against such type of attacks. The SPT system uses the core of this application, together with *Sipp* with pre-set call schemes in the XML format [10], to simulate a SPIT attack on the target SIP server (Fig. 9).

In the form, the tester defines the values of a valid SIP internal, national and international accounts – the called parties to which the SPIT call will be directed and then the value and password to a valid SIP account – the caller through which the call will be initiated. Where the tester fails to define these values, the system automatically assigns an account and an appropriate password from the list created while scanning and monitoring the central.



SPIT

SPIT ATTACK		
Enabled	No [✓ Yes]	
Random valid extension	Random valid extension	
Caller SIP ID username	7204	Random valid account
Caller SIP ID password	******************	

Fig. 9. SPT System - SPIT Module.

To be able to process parameters entered into *Sipp* and XML dynamically, we applied two methods. Using the first one, a correct parameter is assigned to the values entered into the forms by the tester, and it is then sent to *Sipp*. In order to dynamically switch the telephone numbers based on the inserted numbers in the forms, values in the XML scheme for *Sipp* need to change dynamically too. This is the purpose of the .csv file which is generated by the application directly after the tester inserts numbers in the form. This is where the

XML scheme gets the telephone numbers and *Sipp* creates the SIP message for the softswitch.
Created scheme serves as instructions according to how *Sipp* generates the calls. Fig. 10 shows an INVITE request in XML scheme for Sipp.

```
 <send retrans="500">
<![CDATA[

INVITE sip:2717@[remote_ip] SIP/2.0
Via: SIP/2.0/[transport][local_ip]:[l
ocal_port];branch=[branch]
From: [service]<sip:[service]@[remote
_ip]>;tag=[call_number]
To: 2717 <sip:2717@[remote_ip]>
Call-ID: d///[call_id]
CSeq: 3 INVITE
User-Agent: Grandstream GXP2000
1.1.6.37
Contact: <sip:[service]@[local_ip]:[l
ocal_port];transport=[transport]>
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,R
EFER,OPTIONS,INFO,SUBSCRIBE,UPDATE,PR
ACK,MESSAGE
Supported: replaces, timer, path
Subject: Performance Test
Content-Type: application/sdp
Content-Length: [len]

v=0
o=[service] 8000 8001 IN
IP[local_ip_type] [local_ip]
s=SIP Call
c=IN IP[media_ip_type] [media_ip]
t=0 0
m=audio [auto_media_port] RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=ptime:20
a=sendrecv

]]>
</send>
```

Fig. 10. INVITE Request in Sipp XML Scheme.

INVITE message initiates a call to the SIP server but before *Sipp* sends the registers on dedicated account on SIP server, over which the calls will be initiated. Fig. 11 shows code in a XML scheme, that is used for sending prerecorded voice message in .pcap format. This message is sent by a *Sipp* as RTP flow to the called party.

```
<nop>
<action>
<exec
play_pcap_audio="/home/filip/Plocha/S
PITFILE/pcap_examples/g711u.pcap"/>
</action>
</nop>

<pause milliseconds="9000"/>
```

Fig. 11. Sending .pcap Message over Sipp.

If the test was successful, a SIP call is initiated from the caller's account, and the end device with the registered account of the called party starts ringing. Once the call is answered, a pre-recorded message is played and the call terminated. Time required to carry out the test T_{spit} is determined by the length of the pre-recorded message. The final report on penetration tests which the tester receives via e-mail, will, besides information on all previous tests, also contain an analysis and success rate of the SPIT module's test.

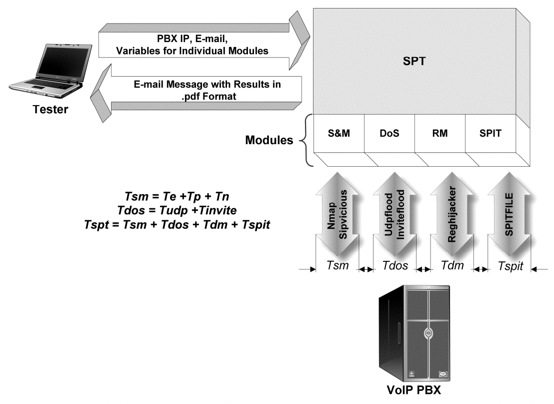


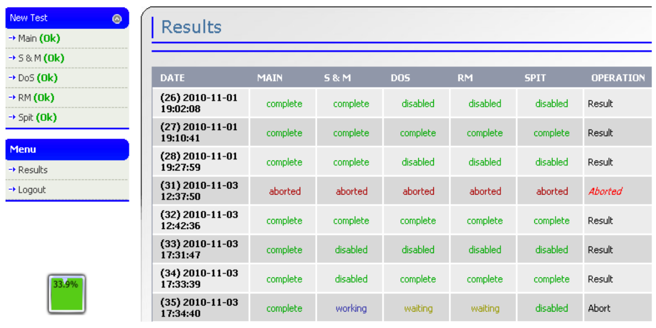
Fig. 12. Division of the SPT system into individual modules.

Fig. 12 illustrates the division of the SPT system into individual modules and shows time intervals necessary to carry out individual tests in respective modules. Time requirements of the whole SPT system can be expressed by equation (8). Its value depends on many factors and can radically change in accordance with the type of tests requested by the tester. Its value is for reference only.

$$T_{spt} = T_{sm} + T_{dos} + T_{rm} + T_{spit} \tag{8}$$

3 Results

Although the SPT system is still in the phase of intensive testing and development, basic operational tests of all available modules were carried out (Fig. 13). Each test is accompanied by a short description of countermeasure's principles and methods [12] which should limit or completely mitigate potential security gaps that were revealed during SIP servers' testing.



DATE	MAIN	S & M	DOS	RM	SPIT	OPERATION
(26) 2010-11-01 19:02:08	complete	complete	disabled	disabled	disabled	Result
(27) 2010-11-01 19:10:41	complete	complete	complete	complete	complete	Result
(28) 2010-11-01 19:27:59	complete	complete	disabled	disabled	disabled	Result
(31) 2010-11-03 12:37:50	aborted	aborted	aborted	aborted	aborted	*Aborted*
(32) 2010-11-03 12:42:36	complete	complete	complete	complete	complete	Result
(33) 2010-11-03 17:31:47	complete	disabled	disabled	disabled	disabled	Result
(34) 2010-11-03 17:33:39	complete	disabled	complete	complete	complete	Result
(35) 2010-11-03 17:34:40	complete	working	waiting	waiting	disabled	Abort

Fig. 13. SPT System – Results.

Before the final report is sent to the tester's e-mail, the our system is going to check up the completness and correctness. The variable *$CRow[0]==1* represents the successful validation while *$CRow[0]==-1* is returned in fault. The final report contains the gained results and security recommendations. A method *$mail->Send()* is applied for sending and the appropriate record si created in database to avoid resending. The part of script is depicted on Fig. 14.

```
$Result = mysql_query("SELECT id,rid F
ROM t_test   WHERE   type='0'   AND
value='0'");
for($r=0;$r<mysql_num_rows($Result);$r
++)
 {
  $Row = mysql_fetch_row($Result);
  $RResult                         =
mysql_query("SELECT data FROM    t_raw
WHERE id='".$Row[1]."'");
  $RRow = mysql_fetch_row($RResult);
  $Data = unserialize($RRow[0]);
$CResult = mysql_query("SELECT value F
ROM t_test WHERE rid='".$Row[1]."' AND
type>=1 AND type<=5 ORDER BY type");
  $Ok = true;

for($i=0;$i<mysql_num_rows($CResult);$
i++)
   {
    $CRow = mysql_fetch_row($CResult);
    if(!($CRow[0]==-1 || $CRow[0]==1))
    {
     $Ok = false;
    }
   }
  if($Ok)
   {
    mysql_query("UPDATE   t_test   SET
start=NOW(),value=10             WHERE
id='".$Row[0]."'");

$From = 'info@pen.cesnet.org';
$To = $Data['Main']['Email'];
$Subject = 'Results of test pen.cesnet
.org';
$Body = "";
$CResult2 = mysql_query("SELECT type,v
alue,result    FROM    t_test    WHERE
rid='".$Row[1]."'   AND   type>=1   AND
type<=5 ORDER BY type");

for($i=0;$i<mysql_num_rows($CResult2);
$i++)
      {
       $CRow                           =
mysql_fetch_row($CResult2);
       switch($CRow[0])
        {
         case 1:
          $Body   .=   "Result   of   Main
test\n";
          $Body .= "===========\n";
          break;

         case 2:
          $Body .= "Result of Scanning
and Monitoring test\n";
          $Body .= "===========\n";
          break;



        case 3:
          $Body .=  "Result  of  Denial
of Service test\n";
          $Body .= "============\n";
          break;

         case 4:
          $Body .=       "Result       of
Registration Manipulation test\n";
          $Body .= "=============\n";
          break;

         case 5:
          $Body   .=   "Result   of   SPIT
test\n";

          $Body .= "=============\n";
          break;
        }
       $Body .= $CRow[2];
       $Body .= "\n\n";
      }
    $mail = new PHPMailer();
    $mail->IsSMTP();
    $mail->CharSet  = "utf-8";
    $mail->Host = "smtp.cesnet.cz";
    $mail->From = $From;
    $mail->FromName =
'info@pen.cesnet.org';
    $mail->AddAddress($To);
    $mail->Subject = $Subject;
    $mail->Body  = $Body;
    if(!$mail->Send())
```

Fig. 14. Validation and final report sending.

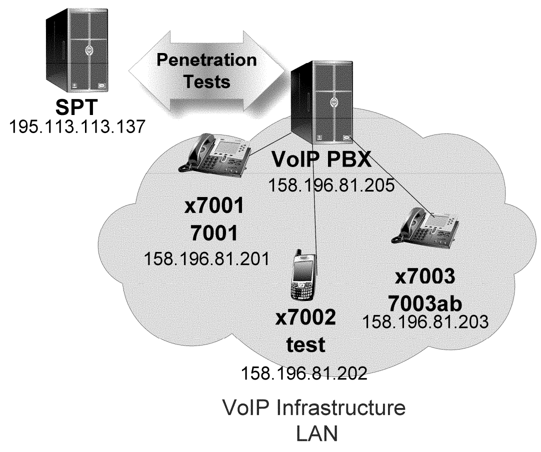


Fig. 15. SIP Penetration Tests System Testbed.

Fig. 15 describes the basic testing topology. The system is denoted as SPT and Asterisk as VoIP PBX. Asterisk was installed at Dell PowerEdge R510 server.

3.1 Scanning and Monitoring

The first step was to record SIP server's IP address and the e-mail address to send the final report to. Next, the S&M module and subsequently *Nmap* and *SIPVicious* applications were launched. Values for *Nmap* were set by default, value of E_{nr} for *SIPVicious* was set to range between 1000-9999. The device found all three registered accounts E_{valid} 7001-7003 and listed open TCP and UDP ports at Asterisk. Once P_{nr} was set to 7001-7003 and a text file P_{dr} containing test and 7003ab string, the test to obtain passwords to individual accounts was also successful. Total time incurred on testing module $T_{sm} \cong 235$s. If we had to protect and prevent SIP server from scanning and monitoring, then an implementation of firewall is the effective solution or an intrusion detection system that is able to distinguish scanning and monitoring. The next effective solution is to divide the network logical infrastructure into VLANs and decompose the provided services into more physical servers (TFTP, HTTP servers). The prevention of accounts and passwords detection is difficult, moreover, the tools for detection apply the standard SIP methods and is not trivial to distinguish legitimate behavior from an attack. In this case, there is recommended to divide the infrastructure into individual VLANs so that the detection for intruder was as difficult as possible.

3.2 Denial of Service

Using *udpflood*, the tester sent 500000 UDP packets directly to port 5060. Bandwidth was set to 100Mbps, Asterisk processed 90% calls. Once the test was completed, Asterisk recovered to a full operation mode. To be able to compare, we substituted Asterisk by OpenSIPS in this test. Call processing under the same attack was entirely error-free. When testing using *inviteflood* on the valid account 7001, we found out that this attack is much more destructive in terms of computational power. As early as at 100000 INVITE request when $T_{invite} \cong$ 9s, CPU load for both Asterisk and OpenSIPS reached 100% and failed to process a single incoming or outgoing call. Once the test was completed, both centrals recovered to a full operation mode. The possibilities, how to protect from Flood DoS attacks, are the following: to divide the network infrastructure into separate VLANs, to have in use solely TLS, to implement L2 network elements with DoS detection or to apply SIP firewall that can detect DoS attacks and minimalize their impact.

3.3 Registration Manipulation

When testing possibility for registration manipulation, we entered values of account 7003 and its password 7003ab manually into the system. Once the test was completed, we established whether the attack was successful. The aim of the attack was to de-register account 7003 and to direct all incoming calls to a fake account which does not exist. Thus, calls were terminated as unconnected. The call to 7003 was not put through. The TCP protocol is recommended at transport level to prevent a registration hijacking because the manipulation with TCP requires higher complexity. Next option, how to minimalize this threat, is to use REGISTER message authentication. We could decrease the registration interval, as well, it is quite simple but effective.

3.4 Spam over Internet Telephony

As stated above, we used *SPITFILE* application, developed by this paper's authors, to test the central's vulnerability to SPIT attacks. The tester entered manually into the system the value of a valid account 7002 on which a SPIT attack was to be initiated, as well as the value of a valid account 7003 and password to it (7003ab) which was supposed to initiate the SPIT call. Once the test was launched, *SPITFILE* registered on the participant 7003 and then started to generate a call to account 7002. The end device registered on 7002 began ringing, and once the call was answered, a recording with an advertisement was played. A few methods exist how to restrict the SPIT propagation, which are more or less efficient, but their combination bring quit strong protection against the type of attack. Among these methods the utilization of the various automatically or manually editable lists belong, on their base the call is permitted or prohibited, eventually an interaction with voice menu can be the effective protection against call bots. Authors developed own solution *ANTISPIT* [12] that exploits the specific human behaviour and automatically modifies the Blacklist table without participation of called party, the approach is based on the statistical Blacklist.

4 Conclusion

The aim of the authors was to develop a tool to carry out penetration tests on SIP servers. The system that was designed and implemented consists of several modules that are able to generate selected types of attacks which the authors deem most popular. The system then analyses to what extent is the target component secured, drafts assessments containing tests' results and proposes factual recommendations to ensure security against the threat concerned. The assessment report is sent as a text document to an e-mail. The system is currently under intensive testing. It is planned that in the future, it will be extended to include other testing modules and functions such as for instance testing of the whole VoIP infrastructure and heavy testing of individual components.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 218086.

References:

[1] J. Bates, C. Gallon, M. Bocci, S. Walker and T. Taylor, "*Converged Multimedia Networks*", Wiley, 364 p., 2006.

[2] R. Chochelinski and I. Baronak, "Private Telecommunication Network Based on NGN", *In 32nd International Conference on Telecommunications and Signal Processing,* August 2009, Dunakiliti, HUNGARY, pp. 162-167.

[3] F. Rezac, M. Voznak and J. Ruzicka, "Security Risks in IP Telephony," *CESNET Conference 2008 Security, Middleware and Virtualization,* September 2008 , Prague, ISBN 978-80-904173-0-4.

[4] Z. Kocur, V. Machula, J. Kulda and L. Vojtech, "Analysis of Influence of Disturbance Level on Data Transmission in Wireless Networks." *In 33rd International Conference on Telecommunications and Signal Processing.* Budapest: Asszisztencia Szervező Kft., August 2010, p. 292-296.

[5] M. Kumar, M. Hemalatha, P. Nagaraj and S. Karthikeyan, "A New Way Towards Security in TCP/IP Protocol," *14th WSEAS International Conference on COMPUTERS,* Corfu Island, Greece, July 23-25, 2010, pp. 46-50.

[6] J. Asim and U. Shafique, "Network Risk Management," *4th International Conference on Communications and Information Technology,* Corfu Island, Greece, July 22-25, 2010, pp. 141-145,

[7] S. Vincent and J. Raja, "A Survey of IP Traceback Mechanisms to overcome Denial-of-Service Attacks," *Proceedings of the 12th International Conference on NETWORKING,VLSI and SIGNAL PROCESSING,* University of Cambridge, UK, February 20-22, 2010, pp. 93-98.

[8] V. Patriciu and A. Furtuna, "Guide for Designing Cyber Security Exercises," *Proceedings of the 8th WSEAS International Conference on INFORMATION SECURITY and PRIVACY,* Puerto De La Cruz, Tenerife, Canary Islands, Spain, December 14-16, 2009, pp. 172-177.

[9] M. Voznak, "Speech Bandwith Requirements in IPsec and TLS Environment," *13th WSEAS International Conference on Computers,* Rodos, July, 2009, pp.217-220.

[10] M. Voznak and J. Rozhon, "SIP Infrastructure Performance Testing," *In WSEAS 9th International Conference on Telecommunications an Informatics,* Catania, Italy, 2010, pp. 153-158.

[11] I. Pravda and J. Vodrazka, "Voice quality planning for NGN including mobile networks", *12th International Conference on Personal Wireless Communications (PWC 2007)*, 2007, Prague.

[12] M. Voznak and F. Rezac, "The implementation of SPAM over Internet telephony and defence against this attack," *The 32nd International Conference on Telecommunications and Signal Processing, Dunakiliti,* HUNGARY, Aug 26-27, 2009, pp. 200-203.

[13] M. Voznak, F. Rezac and M. Halas,"Speech Quality Evaluation in IPsec Environment," *In Proceedings of the 12th Inter. Conference on Networking, VLSI and Signal Processing - ICNVS*

2010, University of Cambridge, February 2010, pp. 49-53.

[14] M. Voznak, F. Rezac and J. Zdralek, "VoIP Based System for the Message Distribution," *In Proceedings of the 9th International Conference on Telecommunications and Informatics*, Catania, Italy, May 2010, pp. 49-53.

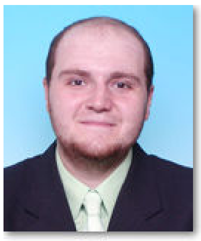
[15] M. Voznak and J. Rudinsky,"Alternative Methods of Intelligent Network Service Implementation in IP Telephony," *The 14th WSEAS International Conference on Communications*,Corfu, Greece, July 2010, pp.204-207.

[16] M. Voznak, "Advanced implementation of IP telephony at Czech universities", *WSEAS Transactions on Communications, Volume 9, Issue 10,* October 2010, pp. 679-693

[17] M. Voznak, J.Rozhon,"Methodology for SIP infrastructure performance testing," *WSEAS Transactions on Computers, Volume 9, Issue 9,* September 2010, pp. 1012-1021

About Auhors

Miroslav Voznak holds position as an associate professor with Department of telecommunications, Technical University of Ostrava. He received his M.S. and Ph.D. degrees in telecommunications, dissertation thesis “Voice traffic optimization with regard to speech quality in network with VoIP technology” from the Technical University of Ostrava, in 1995 and 2002, respectively. Topics of his research interests are Next Generation Networks, IP telephony, speech quality and network security.



Filip Rezac received his M.S. degree in telecommunications from VSB - Technical University of Ostrava, Czech Republic, in 2009 and now he continues with Department of Telecommunication as assistant and also continues in studying Ph.D. degree on the same department. His research is focused on Voice over IP Security, Networks and Speech Quality.