Challenge 4: VoIP (intermediate)

Submission Template

Submit your solution at http://www.honeynet.org/challenge2010/ by 17:00 EST, Wednesday, June 30th 2010. Results will be released on Wednesday, July 21st 2010.

Name (required): Franck Guénichot	Email (required): franck.guenichot@orange.fr
Country (optional): France	Profession (optional): _ ~~Student~~ _ ~~Security Professional~~ _ Other

Nota: In this document, I've use some picviz graphs and custom scripts. These elements can be found in a password-protected zip archive at http://malphx.free.fr/dotclear/public/HPFC4-stuff.zip. The secret password is: $v0ipch4lL3nge!

Section 1/ Question 1. What protocol is being used? Is it TCP or UDP?	Possible Points: 1pt
Tools Used: awk, sort, uniq, grep, SIPlogparser.rb (custom tool) Awarded Points:	

Answer

Session Initiation Protocol (SIP) message as specified in RFC 3261, must have a “Via” header field. This field must indicate the protocol and protocol version: SIP 2.0, but it specifies also the transport protocol selected (TCP or UDP).

So, to know which transport protocol was used, we just have to look at all the “Via” parameter of all the logged SIP messages.

We can quickly scan the log file with this one-liner:

```
grep "Via:" logs_v3.txt |awk '{match($0,"Via: SIP\/2.0\/(TCP|UDP)");print substr($0,1,RLENGTH);}' |sort |uniq
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ grep "Via:" logs_v3.txt |awk '{match($0,"Via: SIP\/2.0\/(TCP|UDP)");print substr($0,1,RLENGTH);}' |sort |uniq
awk: AVERTISSEMENT: séquence d'échappement « \/ » traitée simplement comme « / »
Via: SIP/2.0/UDP
```

We know that the Via header field is mandatory and is constructed like below:
Via: SIP/2.0/(transport) where transport can be TCP or UDP. So, all the messages stored in this log file have used UDP as transport protocol.

I've written a small ruby script to parse the given log file. It is far from being perfect but it makes the job ! Its name: SIPlogparser.rb

With "-z" option switch, SIPlogparser.rb is able to display general statistics about the SIP messages that have been logged in the given file:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ ruby SIPlogparser.rb -r
logs_v3.txt -z

...::: General Statistics :::...

0/4266 messages filtered
.....................................
4266 UDP messages / 0 TCP messages
.....................................
4 SIP INVITE messages
7 SIP SUBSCRIBE messages
4254 SIP REGISTER messages
1 SIP OPTIONS messages
.....................................
```

The result above also indicates that all the messages within this log files have traveled using UDP.

Section 1/ Question 2. Could this log be the result a simple nmap scan being run against the honeynet? Explain	Possible Points: 1pt
Tools Used:	

Answer

If "simple nmap scan" refers to a basic usage of this powerful tool (like a "simple" UDP scan for example), well, no, this log could not be the result of an nmap recon.
Because, an UDP scan could not have sent SIP methods (OPTIONS, REGISTER...) like those in the log.
Here the explanation from the nmap book by Fyodor:
" *UDP scan works by sending an empty (no data) UDP header to every targeted port*"

But, I think it could be possible to have similar results using the Nmap Scripting Engine (NSE) and a well-written script.

Section 1/ Question 3a. Name the scanning tools that may have been used to by the attacker.	Possible Points: 1pt
Tools Used: vi	

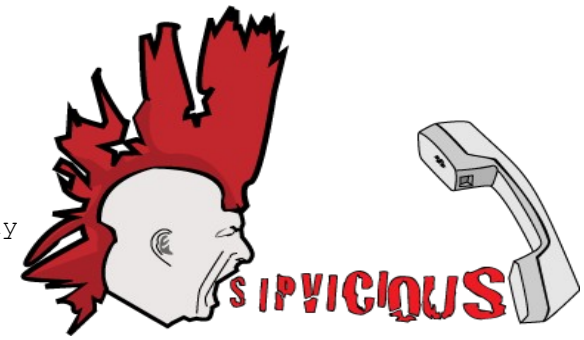
Answer

There is at least two informations which can help us guessing the tool that was used by the attacker. The first one is the name "sipvicious" and the second one is the user-agent string: "friendly-scanner"

```
Source: 210.184.X.Y:1083
Datetime: 2010-05-02 01:43:05.606584

Message:

OPTIONS sip:100@honey.pot.IP.removed SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5061;branch=z9hG4bK-2159139916;rport
Content-Length: 0
From: "sipvicious"<sip:100@1.1.1.1>; tag=X_removed
Accept: application/sdp
User-Agent: friendly-scanner
To: "sipvicious"<sip:100@1.1.1.1>
Contact: sip:100@127.0.0.1:5061
CSeq: 1 OPTIONS
Call-ID: 845752980453913316694142
Max-Forwards: 70
```

Googling for these two words sends you to this site: http://blog.sipvicious.org/
Here we learn that SIPvicious is a set of tools written by Sandro Gauci

From the README file of the SIPvicious package:



The tools:
svmap - **this is a sip scanner**. When launched against
ranges of ip address space, it will identify any SIP servers
which it finds on the way. Also has the option to scan hosts
on ranges of ports.

svwar - **identifies working extension lines on a PBX**. A working extension is one that can be registered.
Also tells you if the extension line requires authentication or not.

svcrack - a **password cracker making use of digest authentication**.
It is able to crack passwords on both registrar servers and proxy servers. Current cracking modes are either numeric ranges or words from dictionary files.

svreport - able to manage sessions created by the rest of the tools and export to pdf, xml, csv and plain text.

```
svlearnfp - allows you to generate new fingerprints by simply running
the tool against a host. It will attempt to guess most values and allow
you to save the information to the local fingerprint db. Then you can
choose to upload it to the author so that it can be added to the database.

For usage help make use of -h or --help switch.

Also check out the wiki:
http://code.google.com/p/sipvicious/w/list

And if you're stuck you're welcome to contact the author.

Sandro Gauci
sandrogauc at gmail dot com
```

(I like the reference to the well-known bass player of the punk band Sex Pistols named Sid Vicious)

Section 1/ Question 3b. What was the tool suite author's intended use of this tool suite ? Who was it designed to be used by?	Possible Points: 1pt

Tools Used: SIPvicious Project site

Answer

Well, originally, SIPvicious Tools suite had been developed to help VoIP Administrators and Security professionals to assess SIP systems security.

Quote from the author (FAQ):
http://code.google.com/p/sipvicious/wiki/FrequentlyAskedQuestions

The idea behind the tools is to aid administrators and security folks make informed decisions when evaluating the security of their SIP-based servers and devices. The tools are intended to be used for educational and demonstrational purposes. We advise people to request permission before making use of the tool suite against any network. Just like a knife, it can be used for good and bad. We hope that SIPVicious tool suite proves to be a very sharp one.

Section 1/ Question 3c. One of these tools was only used against a small subset of extensions. Which were these extensions and why were only they targeted with this tool ?	Possible Points: 2pts
Tools Used: svmap.py,svwar.py,svcrack.py source files, SIPlogparser.rb, picviz	

Answer

Before answering this question, it is important to review the usage of the different SIPvicious tools, their default behavior and signatures (if any):

- **svmap.py** => scans networks to find SIP PBX.
  - Default behavior: sends an OPTIONS message to the 100 extension (exten) on the targeted IP.
  - Signature:
    - `From and To header field set to value: “sipvicious” <sip:100@1.1.1.1>`
    - User-agent header field set to value: friendly-scanner

- **svwar.py** => scans a given SIP PBX to find valid extensions. Displays also, extension that need authentication.
  - Default behavior: Sends REGISTER messages to a given range of exten or name store in a dictionary file.
  - Evaluates the PBX response code to define if the exten exists and needs authentication. The rules below are used to verify extensions validity:
    - “200 OK” response code indicates a valid exten that doesn't need authentication
    - “404 Not found” response code means the exten is invalid.
    - “401 Unauthorized” response code means the exten is valid but needs authentication.
  - Well, these are basic rules to understand how it works, other SIP response codes are take into account by svwar.py to evaluates extension validity.
  - Signature
    - User-Agent set to: friendly-scanner

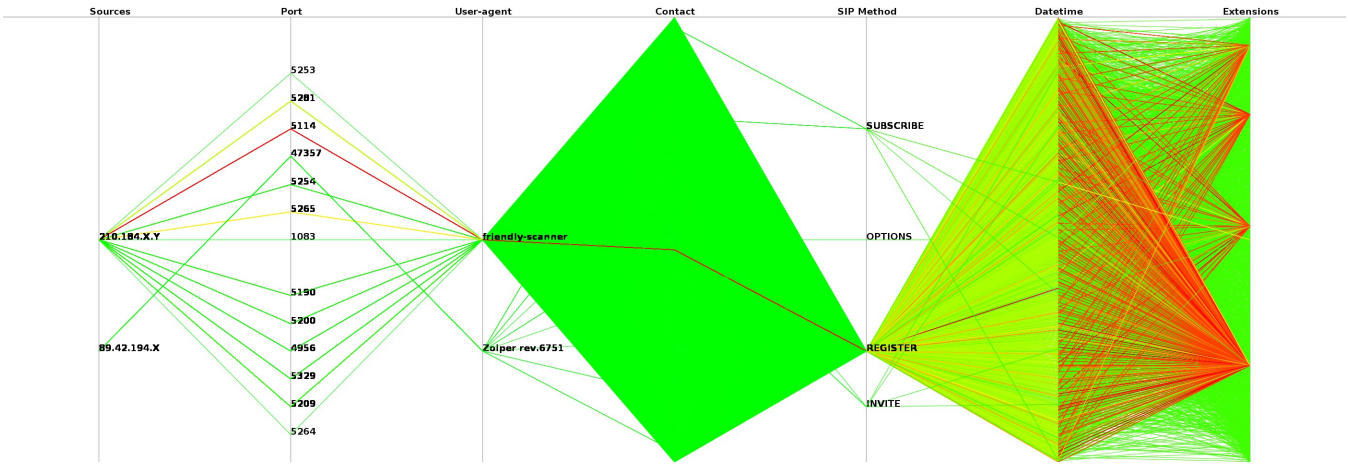
- **svcrack.py** = Tries to find secret password of a given extension (doing brute-force and/or dictionary attack)
  - Default behavior: Sends REGISTER message to sip:TARGET_IP_ADDR
  - uses brute-force or a dictionary to generate MD5 hashes in authorization header field.
  - SIP message sent by svcrack.py don't always use an authorization message header field, the reason explain by the author himself is: to force the refresh of the nonce sent by the SIP PBX.
  - Signature:
    - Request-line set to REGISTER sip:TARGET_IP_ADDR SIP/2.0
    - `Contact message header field set to: sip:123@1.1.1.1`
    - User-Agent message header field set to: friendly-scanner

With all those informations in hands, it is now possible to trace the usage of a specific tool of the SIPvicious suite against extensions on the victim's PBX. But, it is important to remember that those behaviors and signatures are only default values that will only appear if the tools are used without any specific options. For example, svmap.py and svwar.py allow the attacker to change the default SIP method used to scan.

Now it's time to answer this question, a quick analysis of the log file reveals a 4 phases attack.

1. **Phase 1**: An svmap.py scan for SIP PBX land on the fake SIP server (SIP OPTIONS To 100@1.1.1.1)
2. **Phase 2**: A scan is launched with svwar.py against a large amount of extensions.
3. **Phase 3**: svcrack.py is used against extensions that were found valid, but that need authentication.
4. **Phase 4**: unprotected extensions and password-cracked extensions are used by an attacker to call international phone numbers.

I've used picviz [http://www.wallinfire.net/picviz/index.html] to visualize the dataset. Picviz can ease visualization of what has happened. Below is the big picture:



Well this large picture doesn't fit well in this document,so you can find it here:http://malphx.free.fr/dotclear/public/HPFC4-stuff.zip/Big_picture.png
I've chosen to display the dataset based on 7 axis.

- Axe 1 => Sources (IP adresses)
- Axe 2 => Port (Source port)
- Axe 3 => SIP User-agent
- Axe 4 => SIP Contact header fiel
- Axe 5 => SIP method of the message
- Axe 6 => Datetime
- Axe 7 => Extensions (Targeted by the message)

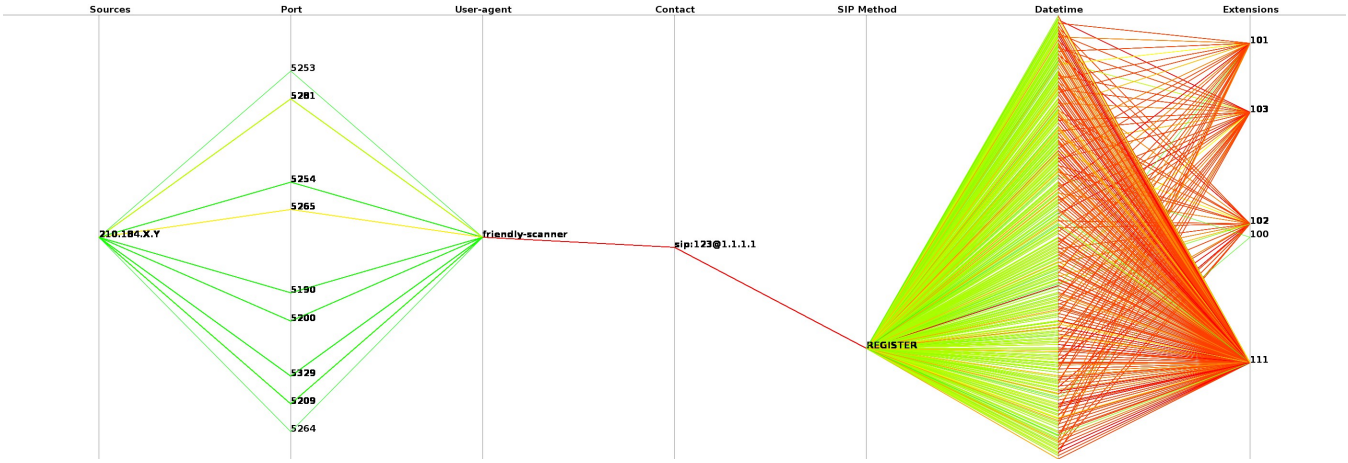
SIPlogparser.rb has an option switch (-p) to generate a PGDL file name “graph.pcv”.
This file can further be rendered with the Picviz CLI tool: pcv.
To generate the picture above, I've used the following command lines:

- `ruby SIPlogparser.rb -r logs_v3.txt -p`
- `pcv -Tpngcairo graph.pcv -rrra -o big_picture.png -Rheatline`

We can easily notice the “red lines” against a small subset of extension, in fact 4 extensions only. And a red line beginning from source 210.184.X.Y and passing through the REGISTER SIP method. Using the tool's “signatures” explained previously, let's try to filter this graph to reveal what tool was used against these 4 extensions. After some tries, here the result:

You can generate the picture below from the “graph.pcv” file and this command line:

```
pcv -Tpngcairo graph.pcv -rrra -o svcrack.png -Rheatline 'value="sip:123@1.1.1.1" on axis 4 and value="REGISTER" on axis 5'
```



You can find this picture here: http://malphx.free.fr/dotclear/public/HPFC4-stuff.zip/svcrack.png
This picture depicts the use of svcrack.py on a small subset of extensions.

Those extensions were:

- 101
- 102
- 103
- 111

All from the domain: honey.pot.IP.removed

Even if the SIP server responses are not logged in the given log file, we can guess what has happened and why those extensions were targeted with svcrack.py.

From phase 2 of the attack scenario, our attacker has a list of valid extensions (protected and unprotected) on this honeypot. So, with thess informations, he then launched a brute-force attack against extensions that were found protected by a password. (ie: Extensions for which svmap.py has received a 401 Unauthorized Status Code from the server when it has tried to register them).

Section 1/ Question 4a. How many extensions were scanned? Are they all numbered extensions, or named as well?. List them	Possible Points: 2pts

Tools Used: SIPlogparser.rb

Answer

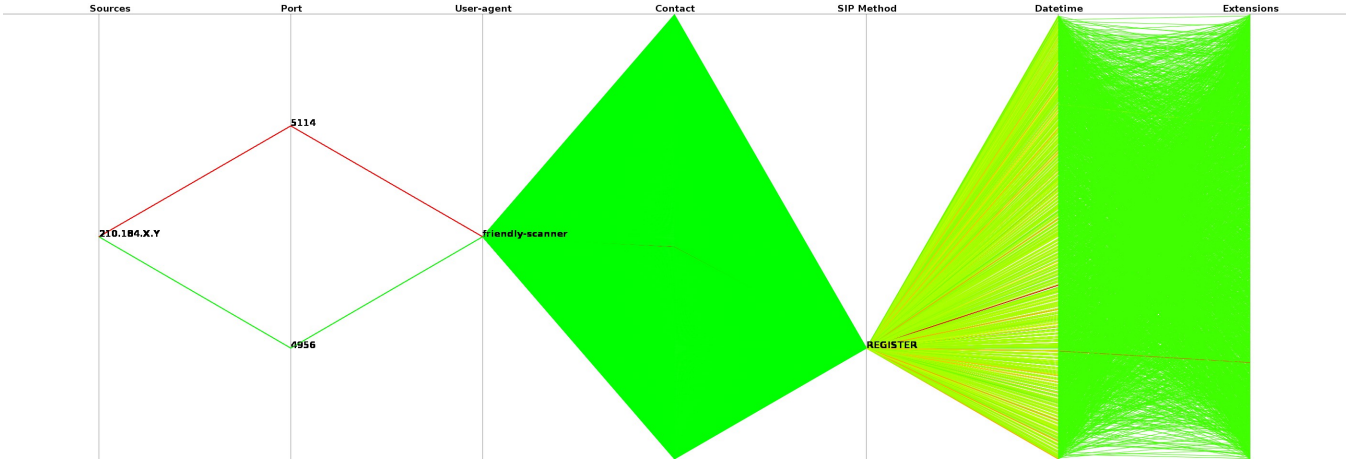
From the previously explained analysis and tools use. We can consider that the scan was made with svwar.py. Looking at the statistics given by SIPlogparser.rb, we conclude that the scan was done with the default proposed method: SIP REGISTER.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ ruby SIPlogparser.rb -r
logs_v3.txt -z
Parsing logs_v3.txt

...::: General Statistics :::...

0/4266 messages filtered
.....................................
4266 UDP messages / 0 TCP messages
.....................................

...::: Sources Statistics :::...

210.184.X.Y:5114 : 2607 messages sent
210.184.X.Y:5281 : 965 messages sent
89.42.194.X:47357 : 18 messages sent
210.184.X.Y:5329 : 94 messages sent
210.184.X.Y:5264 : 1 messages sent
210.184.X.Y:5253 : 1 messages sent
210.184.X.Y:5209 : 170 messages sent
210.184.X.Y:4956 : 45 messages sent
210.184.X.Y:5265 : 78 messages sent
210.184.X.Y:5254 : 98 messages sent
210.184.X.Y:1083 : 1 messages sent
210.184.X.Y:5200 : 94 messages sent
210.184.X.Y:5190 : 94 messages sent

...::: SIP Methods Statistics :::...

4 SIP INVITE messages
7 SIP SUBSCRIBE messages
4254 SIP REGISTER messages
1 SIP OPTIONS messages
.....................................
```

As stated previously, REGISTER message statistics don't let any doubt about the method that was used to scan. This value include extensions scanned by svwar.py, but also the brute-force attempt made with svcrack.py.

With picviz, we can view this “phase 2 scan” easily:
command line:

```
pcv -Tpngcairo graph.pcv -rrra -o phase2_svwar_scan.png -Rheatline
'value="friendly-scanner" on axis 3 and value="REGISTER" on axis 5 and value !=
"sip:123@1.1.1.1" on axis 4'
```



[http://malphx.free.fr/dotclear/public/HPFC4-stuff.zip/phase2_svwar_scan.png]

This graph depicts two interesting points:

- Only two UDP source ports were used to scan all the extensions
- The yellow lines between Axis 5 (SIP method) and Axis 6 (Datetime) reveals that multiple extensions were scanned by second (The datetime axis is rounded to second)

Further analysis will reveal that:

- UDP port 5114 was used to scan numbered extensions
- UDP port 4956 was used to scan unumbered (name) extensions

This point answer a part of the question: Scanned extensions were numbered extensions and names.

By combining -o and -m REGISTER option switches of SIPlogparser.rb we can obtain the list of all the extensions that were scanned.
Listing extensions:
cmd: **SIPlogparser.rb -r logs_v3.txt -m register -o > scanned_extensions**

This reveal that **2652 extensions** were scanned.
This number can be decomposed in:

- **2 “test” extensions**
  - "1729240413"<sip:1729240413@honey.pot.IP.removed>
    - Scanned before launching the scan against numbered extensions
  - "3428948518"<sip:3428948518@honey.pot.IP.removed>
    - Scanned before launching the scan against name extensions

These extensions doesn't appear to belong to the scan range. In fact, svwar.py automatically generate these “test extens” and scan them before launching the real scans.

- **44 named extensions**

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ ruby SIPlogparser.rb -r
logs_v3.txt -m register -N -o |more
Parsing logs_v3.txt
admin@honey.pot.IP.removed
info@honey.pot.IP.removed
test@honey.pot.IP.removed
postmaster@honey.pot.IP.removed
sales@honey.pot.IP.removed
service@honey.pot.IP.removed
support@honey.pot.IP.removed
marketing@honey.pot.IP.removed
manager@honey.pot.IP.removed
market@honey.pot.IP.removed
spam@honey.pot.IP.removed
user@honey.pot.IP.removed
data@honey.pot.IP.removed
cpanel@honey.pot.IP.removed
trixbox@honey.pot.IP.removed
news@honey.pot.IP.removed
fax@honey.pot.IP.removed
postfix@honey.pot.IP.removed
owner@honey.pot.IP.removed
client@honey.pot.IP.removed
operator@honey.pot.IP.removed
asterisk@honey.pot.IP.removed
oracle@honey.pot.IP.removed
temp@honey.pot.IP.removed
jobs@honey.pot.IP.removed
shop@honey.pot.IP.removed
help@honey.pot.IP.removed
orders@honey.pot.IP.removed
aaron@honey.pot.IP.removed
steve@honey.pot.IP.removed
andrew@honey.pot.IP.removed
jane@honey.pot.IP.removed
mike@honey.pot.IP.removed
joshua@honey.pot.IP.removed
christopher@honey.pot.IP.removed
abigail@honey.pot.IP.removed
richard@honey.pot.IP.removed
steven@honey.pot.IP.removed
sarah@honey.pot.IP.removed
heaven@honey.pot.IP.removed
freddy@honey.pot.IP.removed
samantha@honey.pot.IP.removed
sebastian@honey.pot.IP.removed
norman@honey.pot.IP.removed

44 extensions were scanned
```

- **2606 numbered extensions (all from domain honey.pot.IP.removed)**

```
          100  101  102   103  104  106  107  108
 109  110  111  112  113   114  115  116  117  118
 119  120  121  122  123   124  125  126  127  128
 129  130  131  132  133   134  135  137  152  159
 175  178  189  191  194   195  196  197  199  202
 205  210  213  216  220   225  226  227  230  233
 237  240  244  247  250   255  260  262  266  267
 269  272  275  279  282   287  290  294  298  299
 301  305  308  314  318   324  331  338  343  349
 355  360  366  373  380   385  392  398  403  408
 413  418  425  431  438   444  450  454  460  465
 472  480  485  490  494   500  504  509  513  518
 523  527  534  539  543   549  558  563  566  570
 574  577  581  586  589   593  596  599  602  606
 610  612  616  619  622   625  628  631  634  640
 643  646  649  652  654   656  659  662  665  667
 669  672  678  683  689   698  701  704  707  709
 712  715  717  721  724   727  729  732  735  738
 741  744  747  750  753   756  757  759  762  766
 769  772  775  778  782   785  790  795  799  803
 808  818  821  827  831   836  839  842  845  846
 849  852  856  859  864   868  871  875  881  886
 887  891  895  896  898   900  905  912  917  920
 923  927  931  936  940   943  946  949  951  954
 956  960  963  968  972   976  979  982  985  989
 993  996  997 1000 1003  1006 1009 1013 1016 1019
1022 1023 1026 1027 1029  1032 1034 1037 1041 1045
1048 1051 1054 1059 1063  1067 1072 1077 1081 1084
1087 1091 1094 1097 1100  1105 1109 1110 1114 1118
1119 1123 1129 1135 1139  1144 1148 1154 1159 1162
1167 1172 1179 1183 1187  1192 1195 1201 1205 1209
1214 1218 1227 1230 1233  1242 1245 1250 1253 1256
1260 1264 1267 1270 1274  1277 1280 1283 1290 1293
1298 1300 1303 1305 1310  1313 1317 1321 1325 1328
1331 1335 1338 1341 1345  1348 1350 1357 1359 1362
1365 1366 1369 1372 1375  1378 1381 1384 1387 1390
1393 1396 1399 1401 1405  1409 1411 1413 1415 1418
1423 1430 1443 1448 1449  1450 1454 1458 1462 1466
1470 1474 1477 1480 1483  1487 1489 1493 1494 1496
1499 1502 1505 1508 1511  1515 1518 1523 1526 1529
1533 1537 1540 1544 1547  1551 1555 1558 1559 1561
1565 1569 1574 1577 1580  1583 1586 1592 1596 1599
1602 1605 1609 1614 1619  1623 1627 1632 1636 1641
1645 1651 1652 1656 1657  1660 1664 1668 1673 1677
1682 1687 1692 1696 1699  1703 1707 1709 1715 1719
1723 1727 1730 1733 1737  1740 1745 1751 1753 1758
1759 1763 1766 1770 1778  1784 1786 1790 1793 1796
1799 1804 1806 1810 1813  1816 1820 1824 1827 1831
1832 1834 1838 1844 1849  1852 1855 1860 1863 1866
1869 1872 1875 1878 1883  1888 1892 1896 1900 1905
1909 1913 1916 1921 1924  1928 1932 1936 1940 1943
```

```
1948 1950 1953 1956 1960   1963 1970 1976 1979 1987
1991 1993 1997 2000 2004   2007 2010 2014 2017 2021
2024 2028 2032 2036 2040   2042 2045 2049 2053 2056
2060 2064 2067 2068 2071   2075 2079 2083 2087 2091
2095 2098 2101 2105 2108   2109 2112 2116 2121 2125
2129 2133 2140 2149 2153   2160 2164 2167 2171 2178
2182 2186 2190 2192 2195   2202 2203 2205 2210 2213
2223 2228 2232 2235 2238   2240 2243 2246 2254 2257
2261 2265 2269 2273 2277   2281 2284 2288 2292 2296
2299 2303 2307 2311 2314   2318 2328 2331 2335 2339
2350 2355 2359 2363 2367   2370 2371 2373 2374 2376
2380 2381 2384 2389 2393   2396 2397 2409 2413 2414
2417 2421 2422 2423 2427   2430 2433 2437 2438 2440
2444 2448 2450 2453 2457   2460 2463 2464 2465 2468
2471 2474 2478 2482 2486   2489 2493 2496 2499 2501
2505 2509 2513 2516 2520   2521 2522 2523 2526 2529
2533 2536 2539 2542 2545   2548 2551 2554 2558 2563
2566 2568 2571 2575 2578   2582 2584 2587 2588 2589
2590 2594 2597 2600 2603   2606 2610 2614 2617 2621
2624 2628 2631 2639 2645   2649 2652 2659 2663 2668
2672 2679 2683 2687 2692   2693 2695 2699 2704 2707
2710 2715 2719 2725 2729   2733 2737 2741 2743 2750
2763 2768 2773 2779 2782   2786 2790 2794 2797 2802
2805 2809 2813 2817 2821   2825 2828 2832 2836 2839
2843 2847 2849 2853 2856   2860 2862 2865 2870 2874
2878 2882 2885 2890 2893   2898 2902 2905 2909 2913
2917 2920 2924 2928 2931   2933 2937 2941 2943 2946
2949 2952 2955 2958 2959   2962 2964 2966 2968 2971
2975 2981 2982 2983 2997   3002 3005 3008 3010 3014
3019 3021 3025 3028 3032   3035 3038 3042 3045 3048
3051 3054 3057 3060 3065   3067 3070 3074 3077 3080
3083 3086 3090 3093 3097   3100 3103 3110 3117 3119
3123 3126 3130 3133 3138   3142 3145 3150 3153 3159
3163 3167 3172 3176 3179   3181 3183 3186 3190 3193
3195 3198 3201 3202 3203   3205 3206 3207 3212 3214
3218 3222 3225 3228 3231   3234 3237 3239 3245 3247
3250 3253 3255 3259 3262   3266 3270 3273 3276 3280
3284 3289 3292 3296 3298   3301 3303 3304 3306 3307
3309 3313 3317 3320 3323   3327 3329 3332 3334 3340
3343 3346 3349 3352 3355   3358 3362 3364 3366 3369
3372 3375 3379 3382 3384   3387 3390 3399 3402 3405
3408 3410 3413 3416 3419   3422 3424 3427 3428 3430
3432 3434 3436 3440 3441   3447 3449 3454 3457 3459
3462 3463 3465 3468 3470   3472 3474 3476 3479 3481
3483 3488 3490 3492 3494   3496 3498 3501 3503 3504
3505 3506 3509 3512 3514   3516 3519 3522 3524 3527
3529 3532 3535 3538 3540   3543 3546 3549 3551 3552
3553 3557 3559 3561 3563   3566 3568 3570 3572 3574
3577 3580 3583 3586 3588   3591 3594 3597 3601 3605
3610 3625 3629 3632 3635   3638 3641 3646 3651 3654
3658 3660 3663 3667 3669   3673 3676 3680 3683 3686
3689 3693 3695 3699 3702   3704 3706 3709 3712 3715
3718 3722 3724 3727 3729   3732 3735 3739 3742 3746
3750 3751 3754 3758 3760   3764 3767 3771 3774 3778
```

```
3781 3783 3786 3790 3794   3798 3802 3805 3808 3810
3814 3818 3819 3820 3823   3826 3829 3830 3834 3838
3841 3845 3850 3852 3853   3857 3861 3864 3867 3871
3875 3878 3881 3884 3888   3891 3894 3897 3901 3905
3910 3914 3918 3921 3925   3930 3934 3938 3943 3948
3951 3955 3958 3962 3965   3969 3975 3977 3982 3986
3990 3993 3997 4002 4005   4009 4013 4016 4020 4023
4027 4030 4033 4036 4039   4043 4049 4055 4059 4062
4066 4068 4071 4073 4076   4080 4082 4085 4086 4088
4091 4095 4097 4101 4104   4108 4115 4117 4121 4123
4126 4128 4131 4134 4136   4138 4140 4142 4144 4146
4149 4151 4154 4156 4158   4161 4164 4167 4168 4171
4174 4176 4180 4186 4189   4192 4195 4197 4199 4201
4203 4205 4209 4212 4214   4220 4222 4224 4227 4230
4232 4235 4237 4239 4241   4244 4247 4249 4251 4253
4255 4257 4259 4262 4265   4267 4270 4271 4275 4282
4289 4293 4296 4300 4303   4307 4309 4312 4315 4318
4321 4323 4326 4330 4332   4335 4337 4340 4342 4345
4346 4348 4350 4352 4354   4356 4358 4360 4363 4367
4369 4372 4374 4377 4379   4381 4383 4384 4387 4389
4392 4397 4399 4401 4403   4405 4410 4412 4414 4416
4418 4421 4424 4426 4429   4432 4434 4437 4439 4440
4441 4443 4446 4449 4452   4455 4458 4461 4464 4467
4469 4472 4475 4478 4479   4482 4485 4487 4490 4493
4496 4499 4502 4504 4507   4510 4513 4517 4520 4523
4526 4530 4535 4539 4543   4547 4551 4554 4555 4558
4561 4564 4568 4569 4573   4576 4579 4584 4588 4589
4591 4596 4601 4606 4612   4618 4625 4630 4635 4641
4645 4650 4654 4670 4678   4684 4689 4695 4707 4715
4722 4727 4733 4737 4743   4747 4752 4758 4761 4765
4769 4773 4777 4780 4784   4786 4791 4794 4798 4803
4812 4817 4822 4827 4833   4836 4840 4846 4850 4853
4858 4861 4864 4868 4871   4876 4879 4883 4887 4891
4894 4899 4903 4906 4911   4915 4919 4922 4927 4931
4935 4940 4953 4957 4962   4968 4972 4977 4983 4986
4990 4994 4999 5005 5010   5015 5020 5029 5035 5041
5046 5053 5058 5062 5068   5091 5101 5115 5119 5123
5124 5129 5133 5138 5143   5148 5152 5157 5161 5165
5170 5176 5180 5183 5188   5193 5197 5200 5205 5209
5213 5218 5224 5231 5235   5239 5243 5248 5253 5258
5264 5268 5271 5272 5275   5279 5284 5288 5291 5295
5299 5303 5307 5310 5315   5318 5322 5326 5330 5334
5338 5341 5345 5348 5353   5356 5360 5361 5364 5367
5368 5371 5374 5377 5380   5384 5386 5390 5393 5395
5399 5400 5403 5406 5409   5412 5415 5417 5420 5423
5426 5429 5433 5436 5440   5444 5448 5452 5455 5457
5460 5463 5466 5469 5472   5478 5482 5484 5487 5489
5494 5496 5498 5499 5502   5506 5508 5511 5514 5517
5519 5522 5524 5528 5530   5532 5535 5538 5541 5552
5558 5561 5568 5571 5574   5577 5580 5583 5587 5590
5594 5598 5601 5605 5609   5612 5615 5617 5625 5628
5632 5634 5640 5643 5646   5648 5651 5656 5659 5662
5666 5669 5673 5678 5682   5683 5685 5689 5693 5697
5703 5706 5710 5714 5719   5723 5729 5733 5739 5745
```

```
5748 5752 5756 5760 5764   5767 5772 5776 5779 5783
5788 5792 5796 5797 5799   5803 5807 5811 5816 5821
5824 5829 5832 5836 5842   5847 5853 5858 5862 5867
5868 5871 5878 5888 5897   5903 5916 5923 5929 5936
5941 5946 5950 5956 5961   5965 5970 5976 5981 5986
5992 5999 6003 6004 6009   6015 6021 6027 6032 6038
6041 6045 6051 6056 6062   6067 6072 6078 6083 6088
6092 6096 6101 6105 6110   6115 6120 6124 6129 6133
6138 6144 6149 6154 6159   6163 6168 6173 6179 6184
6188 6193 6198 6204 6208   6212 6216 6220 6224 6225
6229 6233 6237 6242 6248   6251 6255 6258 6262 6267
6270 6274 6279 6284 6288   6294 6300 6304 6308 6310
6316 6320 6325 6330 6335   6340 6345 6351 6356 6360
6364 6368 6371 6377 6381   6385 6389 6394 6398 6402
6405 6409 6414 6417 6421   6425 6429 6433 6436 6440
6445 6449 6455 6458 6462   6467 6472 6476 6480 6485
6498 6506 6520 6525 6529   6533 6538 6542 6545 6548
6552 6557 6564 6569 6572   6578 6589 6611 6617 6626
6632 6638 6644 6648 6654   6658 6664 6668 6673 6675
6681 6687 6693 6697 6703   6707 6712 6717 6724 6729
6734 6738 6744 6749 6756   6762 6767 6773 6779 6785
6791 6797 6802 6808 6813   6819 6824 6828 6832 6833
6835 6840 6843 6850 6854   6858 6861 6869 6870 6873
6876 6879 6882 6883 6886   6887 6888 6890 6891 6892
6898 6903 6907 6917 6920   6923 6927 6930 6935 6939
6943 6947 6950 6953 6956   6960 6964 6967 6970 6974
6978 6981 6984 6989 6992   6995 6998 7001 7002 7006
7009 7011 7013 7016 7018   7021 7024 7029 7036 7038
7041 7044 7046 7047 7049   7052 7055 7060 7061 7063
7066 7069 7073 7076 7080   7084 7088 7091 7095 7096
7098 7101 7104 7108 7112   7115 7116 7119 7123 7126
7129 7133 7135 7139 7141   7144 7148 7151 7154 7157
7160 7164 7168 7172 7175   7180 7183 7188 7192 7199
7203 7208 7212 7216 7220   7225 7230 7233 7235 7240
7243 7247 7251 7255 7260   7265 7269 7272 7276 7280
7285 7288 7293 7297 7301   7305 7309 7312 7316 7321
7325 7329 7333 7336 7340   7344 7348 7351 7356 7367
7372 7374 7386 7391 7395   7399 7403 7408 7412 7416
7424 7431 7446 7449 7454   7458 7464 7468 7472 7478
7481 7484 7487 7491 7494   7497 7500 7503 7506 7510
7514 7517 7521 7524 7527   7530 7534 7539 7541 7544
7547 7551 7554 7558 7560   7563 7566 7571 7575 7578
7582 7586 7591 7597 7601   7602 7606 7610 7614 7618
7623 7627 7632 7635 7638   7643 7646 7647 7650 7651
7655 7656 7658 7661 7662   7665 7666 7670 7674 7680
7681 7694 7697 7701 7705   7709 7711 7715 7719 7722
7726 7728 7733 7736 7739   7744 7748 7752 7755 7760
7763 7768 7772 7777 7781   7788 7792 7793 7795 7799
7803 7808 7813 7816 7820   7827 7831 7834 7837 7842
7847 7852 7857 7861 7866   7872 7877 7883 7887 7892
7896 7897 7902 7907 7912   7917 7922 7931 7936 7942
7946 7952 7958 7964 7976   7986 7991 7997 8004 8010
8017 8024 8031 8038 8044   8052 8061 8069 8078 8086
8093 8101 8110 8120 8121   8128 8137 8143 8149 8156
```

```
8161 8168 8169 8175 8180   8188 8189 8192 8198 8204
8210 8216 8222 8238 8245   8249 8255 8260 8264 8268
8276 8280 8284 8292 8296   8301 8306 8309 8310 8312
8316 8320 8324 8327 8332   8336 8338 8341 8344 8347
8350 8353 8357 8359 8363   8366 8370 8374 8377 8380
8384 8386 8390 8397 8399   8402 8404 8408 8412 8414
8417 8420 8422 8425 8429   8431 8434 8438 8442 8445
8448 8451 8453 8456 8459   8463 8467 8470 8474 8478
8481 8484 8487 8488 8491   8495 8499 8504 8508 8513
8517 8522 8527 8528 8532   8536 8541 8546 8550 8554
8559 8564 8568 8572 8577   8582 8586 8592 8596 8597
8601 8607 8612 8618 8624   8629 8630 8634 8639 8643
8647 8651 8654 8659 8663   8667 8671 8674 8677 8681
8684 8690 8693 8698 8700   8703 8706 8709 8713 8716
8719 8722 8726 8728 8731   8734 8738 8742 8745 8748
8751 8755 8758 8762 8766   8769 8772 8775 8777 8779
8782 8784 8787 8790 8793   8795 8797 8801 8804 8807
8812 8814 8817 8819 8822   8825 8831 8834 8837 8840
8843 8847 8850 8853 8856   8859 8864 8866 8870 8873
8876 8881 8886 8889 8893   8897 8902 8905 8909 8912
8916 8921 8925 8929 8934   8938 8943 8946 8950 8955
8960 8964 8970 8975 8979   8983 8988 8992 8995 8999
9003 9010 9013 9015 9018   9021 9022 9025 9028 9032
9033 9036 9040 9044 9048   9051 9056 9057 9059 9065
9069 9074 9077 9081 9085   9091 9095 9098 9102 9105
9108 9112 9113 9116 9120   9123 9128 9132 9136 9141
9146 9150 9154 9159 9163   9171 9179 9182 9191 9196
9201 9207 9211 9215 9219   9224 9229 9233 9240 9245
9250 9251 9254 9258 9262   9265 9268 9272 9275 9279
9282 9288 9294 9296 9299   9302 9306 9310 9314 9318
9321 9325 9328 9332 9335   9339 9343 9347 9350 9353
9356 9360 9363 9367 9371   9376 9380 9385 9390 9395
9400 9406 9410 9412 9416   9419 9420 9424 9427 9428
9430 9435 9438 9442 9446   9449 9452 9457 9460 9463
9467 9470 9473 9476 9479   9483 9488 9490 9493 9500
9503 9505 9518 9523 9526   9530 9532 9535 9539 9543
9547 9548 9551 9554 9558   9562 9565 9569 9573 9577
9580 9582 9586 9590 9593   9598 9602 9605 9609 9612
9616 9620 9624 9625 9629   9632 9636 9639 9642 9646
9649 9652 9654 9658 9663   9666 9670 9672 9676 9679
9683 9686 9689 9693 9697   9702 9706 9710 9715 9718
9722 9725 9726 9731 9735   9738 9741 9745 9748 9751
9756 9760 9765 9769 9772   9775 9779 9784 9788 9791
9795 9799 9803 9807 9811   9815 9818 9823 9827 9830
9835 9837 9841 9845 9849   9851 9855 9858 9861 9865
9867 9871 9874 9890 9896   9901 9906 9911 9915 9922
9928 9930 9933 9934 9939   9943 9948 9950 9954 9957
9961 9965 9970 9975 9976   9983 9991 9994
```

For a total of 2652 extensions scanned.

Section 1/ Question 4b. Categorize these extensions into the following groups, and explain to method you used: • Those that exist on the honeypot, AND require authentication • Those that exist on the honeypot, and do NOT require authentication • Those that do not exist on the honeypot	Possible Points: 6pts

Tools Used: SIPlogparser.rb, grep, awk, uniq, sort

Answer

Because the given log file doesn't reveal the SIP PBX responses, we can only base our analysis on the client (attacker) behavior. The previous analysis has taught us some of the behavior of the supposed used tools.
Here are the assumptions I've made:
In the second phase of the attack (svwar.py) REGISTER requests were sent, trying to register 2652 extensions. For each extension scanned, the attacker has received a response from the honeypot containing a status-code:

- For the valid extensions that do not require any authentication, the responses may have been:
  - Status-code: 100 Trying and 200 Ok
- For the valid extensions that require authentication, the responses may have been:
  - Status-code: 100 Trying and 401 Unauthorized
- And finally for the extensions that do not exist on the honeypot, the responses may have been:
  - Status-Code: 404 Not found.

Based on these response status-codes, extensions were categorized automatically for the attacker, and only extensions which required authentication needed to be scanned again, but this time with svcrack.py, trying to brute-force passwords. (phase 3). So, to find those extensions that require authentication, one can filter REGISTER messages containing an “Authorization” header field set. For example with this command line:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ cat logs_v3.txt |grep
"Authorization: " |awk '{split($3,fields,",");print fields[1]}' |sort |uniq
username="101"
username="102"
username="103"
username="111"
```

Then, we can analyze phase 4 of the attack (starting on 2010-05-05 10:00:08.170954 and from 89.42.194.X), the “exploitation” par. In this phase, the attacker has used 2 extensions to call real telephone numbers, these two extensions were: 100 and 101. We already know the 101 was a valid extension but require authentication, but what about 100 ?
89.42.194.X has been able to use register and then use extension number 100 without any authentication, so we can consider that this extension was a valid one that do not required authentication.

All the other extensions that were scanned were invalid on this honeypot.

Summary:

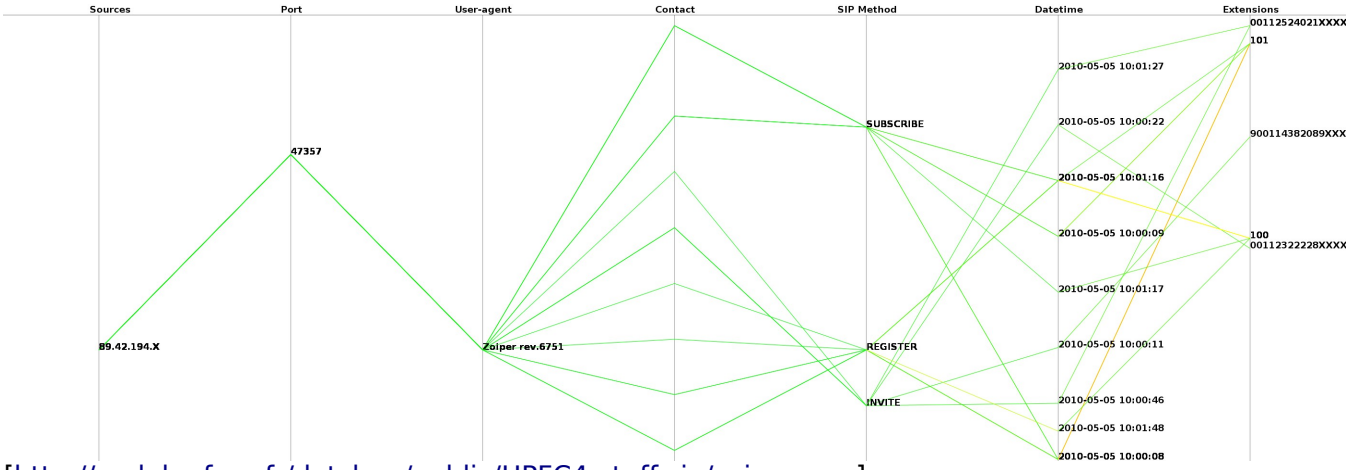
- Valid extensions without need for authentication: 100
- Valid with authentication required: 101, 102, 103, 111
- Invalid: all the other.

Section 1/ Question 5. Was a real SIP client used at any point ? If it was, what time was it used, and why ?	Possible Points: 1pt

Tools Used: SIPlogparser.rb, picviz

Answer

On the big picture.png graph we have notice that 2 different User-agents were used.
The first UA: “friendly-scanner” is known to be used by the SIPvicious tools. The second UA is **Zoiper rev.6751,** meaning that a ZoIPer SIP client was used.
Let's try to depict this with a picviz graph:



[http://malphx.free.fr/dotclear/public/HPFC4-stuff.zip/zoiper.png]
This SIP client was used by 89.42.194.X, between 2010-05-05 10:00:08.170954 UTC and 2010-05-05 10:01:48.058434 UTC, to REGISTER and then use two extensions (100 and 101) from the domain honey.pot.IP.removed.
These two extensions were used to call 3 international numbers:

- at 10:00:11.493635 UTC => 900114382089XXXX
- at 10:00:22.019093 UTC => 00112322228XXXX
- at 10:00:46.147670 UTC => 00112524021XXXX
- at 10:01:27.633156 UTC => 00112524021XXXX

Section 1/ Question 6. List the following, include geo-location information. - Source IP addresses involved - The real world phone numbers that were attempted to be dialled	Possible Points: 3pts

Tools Used: whois, geoiplookup

Answer

We know that only two sources appear in the log file: 210.184.X.Y and 89.42.194.0.
Now, we can replace X and Y by 0 and then try to gather informations on them from whois and geoiplookup databases:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ whois 210.184.0.0
% [whois.apnic.net node-3]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:        210.184.0.0 - 210.184.31.255
netname:        CPCNET-HK
descr:          CPCNet Hong Kong Ltd.
descr:          20/F, Lincoln House,
descr:          Taikoo Place,
```

```
descr:          979 King's Road,Quarry Bay,
descr:          Hong Kong
country:        HK
admin-c:        NC155-AP
tech-c:         NC154-AP
mnt-by:         APNIC-HM
mnt-lower:      MAINT-HK-CPCNET
changed:        hm-changed@apnic.net 20020823
status:         ALLOCATED PORTABLE
source:         APNIC

person:         CPCNet Hong Kong Limited NOC
address:        20/F, Lincoln House,
address:        Taikoo Place,
address:        979 King's Road,
address:        Quarry Bay,
address:        Hong Kong
country:        HK
phone:          +852-2170-7101
fax-no:         +852-2372-0287
e-mail:         hostinfo@cpcnet.com
nic-hdl:        NC154-AP
mnt-by:         MAINT-HK-CPCNET
changed:        hostinfo@cpcnet.com 20100106
source:         APNIC
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ geoiplookup 210.184.0.0
GeoIP Country Edition: HK, Hong Kong
```

The first attacker comes from Hong Kong.

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ whois 89.42.194.0
% This is the RIPE Database query service.
% The objects are in RPSL format.
%
% The RIPE Database is subject to Terms and Conditions.
% See http://www.ripe.net/db/support/db-terms-conditions.pdf

% Note: This output has been filtered.
%       To receive output for a database update, use the "-B" flag.

% Information related to '89.42.192.0 - 89.42.199.255'

inetnum:        89.42.192.0 - 89.42.199.255
netname:        SC-UNIREA-EL-NINO-SRL
descr:          SC Unirea El Nino SRL
descr:          Unirii  G1
descr:          Constanta Constanta
country:        ro
admin-c:        CDG40-RIPE
tech-c:         CDG40-RIPE
status:         ASSIGNED PA
remarks:        Registered through http://www.jump.ro/ip.html
mnt-by:         RO-MNT
mnt-lower:      RO-MNT
```

```
mnt-routes:     Unirea-El-Nino-MNT
source:         RIPE # Filtered

person:       Cristea Dragos George
address:      Str. Aviator Vasile Craiu Nr.
address:      30, Constanta,
address:      Romania
phone:        +40-722-462287
e-mail:       the_angelro@yahoo.com
nic-hdl:      CDG40-RIPE
mnt-by:       AS3233-MNT
source:       RIPE # Filtered

% Information related to '89.42.192.0/21AS41763'

route:          89.42.192.0/21
descr:          SC Unirea El Nino SRL
origin:         AS41763
mnt-by:         Unirea-El-Nino-MNT
source:         RIPE # Filtered
```

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ geoiplookup 89.42.194.0
GeoIP Country Edition: RO, Romania
```

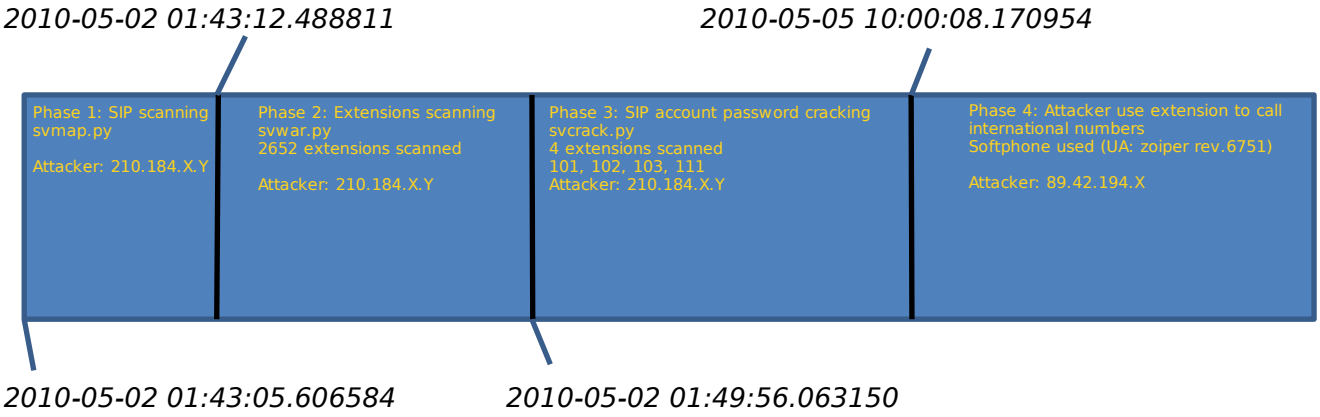
And the second, from Romania.
The Romanian attacker has tried to call these numbers:

- 900114382089XXXX
  - 9 + 0011 + 43 + 820 +89XXXX
  - Location: Austria
  - Type: service number
- 00112322228XXXX
  - 0011 + 232 + 22 + 28XXXX
  - Location: Sierra Leone
- 00112524021XXXX
  - 0011 + 252 + 40 + 21XXXX
  - Location: Somalia

Section 1/ Question 7. Draw a simple static or animated timeline of events, describing when and *where* certain phases occurred from, and what the purpose of each phase was	Possible Points: 5pts
Tools Used:	

Answer

I've decomposed this incident in 4 phases:



- **Phase 1 : Finding SIP servers**
  - Phase start: 2010-05-02 01:43:05.606584
  - Phase end: before 2010-05-02 01:43:12.488811
  - Attacker: 210.184.X.Y (Hong Kong)
  - Tool used: SIPvicious svmap.py
  - Description:
    - In this phase the attacker launch scan session on selected subnets to find valid SIP servers. In this event, the scan was done with a SIP OPTIONS message (default method used by svmap.py)
- **Phase 2: Finding valid extensions on the targeted SIP server**
  - Phase start: 2010-05-02 01:43:12.488811
  - Phase end: 2010-05-02 01:49:46.992699
  - Attacker: 210.184.X.Y (Hong Kong)
  - Tool used: SIPvicious svwar.py
  - 2652 extensions scanned (44 named extensions and 2608 numbered extensions)
  - Description:
    - The attacker tries to find valid extensions (with or without authentication required). He enumerates large range of extensions using SIP REGISTER requests. Based on the received response code from the SIP server, extensions are categorized: valid, valid with authentication, invalid.
- **Phase 3: Cracking password protected SIP accounts**
  - Phase start: 2010-05-02 01:49:56.063150
  - Phase end: 2010-05-02 01:55:11.496170
  - Attacker: 210.184.X.Y (Hong Kong)
  - Tool used: SIPvicious svcrack.py
  - 4 extensions were brute-forced: 102, 103, 101 and 111
  - Description:

        - with the informations gathered in phase 2, the attacker tries to crack extension passwords, for those which need authentication. Svcrack.py has been used for doing the job.

If you look carefully at each phase starting and ending timestamps, you can notice that the delta between the end of one phase and the beginning of the next one is particularly small for the 3 first phases. This leads to think that the attacker has automated those phases, maybe in a script which gathers SIPvicious tools outputs and launch adequate actions to do next.

- **Phase 4: Owning and Using Extensions 100 & 101**
    - Phase start: 2010-05-05 10:00:08.170954
    - Phase end: 2010-05-05 10:01:48.058434
    - Attacker: 89.42.194.X (Romania)
    - Tool used: a Softphone, User-agent: Zoiper rev6751
    - **3 international numbers were called**
        - **900114382089XXXX (on 2010-05-05 10:00:11.493635)**
        - **00112322228XXXX (on 2010-05-05 10:00:22.019093)**
        - **00112524021XXXX (on 2010-05-05 10:00:46.147670 and on 2010-05-05 10:01:27.633156)**
    - Description:
        - This is the last phase of the incident. Another host, based in Romania registers and then use extensions 101 and 100. Although, extension 101 was protected by a password, the attacker correctly authenticates itself and can use the extension to call external numbers. (the password was known to the attacker). This could lead to think that the attacker was the same as in the other phase or maybe he has bought or received SIP usernames and passwords list gathered from the previous phase.

Section 1/ Question 8a. Assuming this were a real incident, write 2 paragraphs of an Executive summary of this incident. Assume the reader does not have IT Security or VOIP experience. a) First Paragraph: Write, in the minimum detail necessary a description the nature and timings, and possible motives of the attack phases. (3 points)	Possible Points: 3pts
Tools Used:	

Answer

Our VoIP system was targeted by a four phases VoIP attack starting on Sunday 2010-05-02 01:43:05 and was initially launched from an host located in Hong Kong. The three first phases of this attack were run on Sunday May, 2nd between 01:43:05 UTC and 01:55:11 UTC; the objectives were clearly information leakage of our internals VoIP extensions. Our analysis of this incident, based on log files, has reveal that the attacker has used publicly available tools to scan and exploit SIP based VoIP systems. The used tool suite is named SIPvicious.
From the first three phases of the attack, the attacker has been able to find our VoIP system, to enumerate valid extensions and to crack at least one of our extensions password.
This information leakage has lead to an unauthorized use of two of our extensions to mahe international telephone calls in the fourth phase of this incident.

This last phase started on Wednesday May,5th at 10:00:08 UTC and was run by another host located in Romania. This attacker has successfully used extension number 100 and 101 of our telephony system to make four international calls. The callee were based in Austria (a service number), Sierra-Leone and Somalia. This last event ended on Wednesday May,5th at 10:01:48 UTC.

Possible motives of the attack may be impersonation, ability to make free international calls or any other nefarious activities.

Section 1/ Question 8b. Assuming this were a real incident, write 2 paragraphs of an Executive summary of this incident. Assume the reader does not have IT Security or VOIP experience. b) Second Paragraph: What actions would you recommend should occur following this particular incident, include any priority/urgency. Also describe any good practices that should be employed to mitigate future attacks.	Possible Points: 3pts
Tools Used:	

Answer

This incident has revealed several weakness in our actual VoIP system implementation:

- Some extensions doesn't require any authentication (e.g. extension number 100)
  - *Anyone (internal or external) can register and then use these extensions*
- Lack of publicly known SIP scanning tools detection
  - *Our IDS/IPS systems were unable to detect/prevent this attack.*
- Our actual VoIP protocols implementations don't use encryption and mutual authentication
  - *This lead to potential eavesdrops on our communications or VoIP signaling exchanges.*
  - *Critical informations can be gathered by an attacker.*

Based on these facts, our recommendations to prevent or mitigate this kind of incident are listed below:

- **Priority: Urgent**
  - Define and implement IDS signatures to detect this tool suite in our IDS/IPS
    - *Our analysis has revealed that the tools were used with their default values. Some of these values can be used to define IDS/IPS signatures and detect or prevent similar future attacks*
  - Limit the number of SIP messages (REGISTER) from external client on our perimeter devices
    - *SIP scanning and extensions enumeration, like any other network scanning technique involve sending large amount of requests. This behavior can be deteted at the perimeter, as it is generally unusual in normal usage of the VoIP system.*
    - *We must quickly define a baseline of "normal" SIP protocol requests usage (REGISTER, OPTIONS, SUBSCRIBE, INVITE...) from internal and external clients and then limit the amount of these requests adequately.*
  - Protect all our extensions with a strong password
    - *even if tools can be used to crack SIP account password, we must protect all our extension with a password. The chosen password must be sufficiently hard to guess (not less than 8 chars length, using letters (uppercase and lowercase), numbers and special chars)*
    - *This will, at least, slow brute-force attack against our system.*
  - **Priority: High**
    - We should consider using secured version of our VoIP protocols. Instead of using unencrypted SIP, we'll have to use SIP over TLS (SIPS). This will provide two-way authentication, confidentiality and messages integrity through the use of strong encryption.
    - We should consider installing a Session Border Controller. This device will protect our SIP servers and devices from various VoIP attacks.

Section 2/ Question 1. Which 4 protocols are involved in the PCAP (VOIP protocols and otherwise) ? Give a brief explanation as to their purpose.	Possible Points: 4pts
Tools Used: tshark	

Answer

As usual for me when I begin a pcap analysis, I use the Protocol Hierarchy Statistics of tshark to get a view of the protocols involved in the attack.

```
===================================================================
Protocol Hierarchy Statistics
Filter: frame

frame                                    frames:4447 bytes:1117758
  eth                                    frames:4447 bytes:1117758
    ip                                   frames:4447 bytes:1117758
      udp                                frames:3154 bytes:662385
        sip                              frames:19 bytes:11971
        rtcp                             frames:21 bytes:3734
          rtcp                           frames:21 bytes:3734
            rtcp.length_check            frames:20 bytes:3532
            rtcp                         frames:1 bytes:202
              rtcp.length_check          frames:1 bytes:202
        rtp                              frames:3113 bytes:646634
          rtpevent                       frames:125 bytes:7250
        data                             frames:1 bytes:46
      tcp                                frames:1291 bytes:454993
        http                             frames:166 bytes:85025
          data-text-lines                frames:9 bytes:7085
          xml                            frames:3 bytes:1512
          png                            frames:7 bytes:6412
          image-gif                      frames:1 bytes:355
        tcp.segments                     frames:18 bytes:12536
          http                           frames:18 bytes:12536
            data-text-lines              frames:16 bytes:11450
            png                          frames:1 bytes:296
            image-jfif                   frames:1 bytes:790
      icmp                               frames:2 bytes:380
===================================================================
```

SIP (Session Initiation Protocol)

SIP is a signaling protocol used for controlling multimedia communication sessions, like voice or video calls over IP. The latest version of SIP is defined in RFC 3261. The protocol can be used for creating, modifying and terminating two-party or multiparty sessions consisting of one or several media streams.
The SIP protocol is an Application Layer protocol designed to be independent of the underlying transport layer; it can run on Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or Stream Control Transmission Protocol (SCTP).[3] It is a text-based protocol, incorporating many elements of the Hypertext Transfer Protocol (HTTP) and the Simple Mail Transfer Protocol (SMTP), [4] allowing for direct inspection by administrators.
Source en.wikipedia.org

Default port: 5060.

In this capture file, SIP is used to create and tear down VoIP sessions.

RTP (Real-time Transport Protocol)

The Real-time Transport Protocol (RTP) defines a standardized packet format for delivering audio and video over the Internet. It was developed by the Audio-Video Transport Working Group of the IETF and first published in 1996 as RFC 1889, and superseded by RFC 3550 in 2003.
RTP is used extensively in communication and entertainment systems that involve streaming media, such as telephony, video teleconference applications and web-based push to talk features. For these it carries media streams controlled by H.323, MGCP, Megaco, SCCP, or Session Initiation Protocol (SIP) signaling protocols, making it one of the technical foundations of the Voice over IP industry.
RTP is usually used in conjunction with the RTP Control Protocol (RTCP). While RTP carries the media streams (e.g., audio and video) or out-of-band events signaling (DTMF in separate payload type), RTCP is used to monitor transmission statistics and quality of service (QoS) information. When both protocols are used in conjunction, RTP is usually originated and received on even port numbers, whereas RTCP uses the next higher odd port number.

Source en.wikipedia.org

In this capture file, RTP is used as the media protocol to transport voice.

RTCP (Real-time Transport Control Protocol)

The Real-Time Transport Control Protocol (RTCP) is a sister protocol of the Real-time Transport Protocol (RTP). Its basic functionality and packet structure is defined in the RTP specification RFC 3550,[1] superseding its original standardization in 1996 (RFC 1889).
RTCP provides out-of-band statistics and control information for an RTP flow. It partners RTP in the delivery and packaging of multimedia data, but does not transport any media streams itself. Typically RTP will be sent on an even-numbered UDP port, with RTCP messages being sent over the next highest odd-numbered port[2]. The primary function of RTCP is to provide feedback on the quality of service (QoS) in media distribution by periodically sending statistics information to participants in a streaming multimedia session.
RTCP gathers statistics for a media connection and information such as transmitted octet and packet counts, lost packet counts, jitter, and round-trip delay time. An application may use this information to control quality of service parameters, perhaps by limiting flow, or using a different codec.
RTCP itself does not provide any flow encryption or authentication methods. Such mechanisms may be implemented, for example, with the Secure Real-time Transport Protocol (SRTP) defined in RFC 3711.

Source en.wikipedia.org

HTTP (HyperText Transport Protocol)

The Hypertext Transfer Protocol (HTTP) is an Application Layer protocol for distributed, collaborative, hypermedia information systems.[1]
HTTP is a request-response protocol standard for client-server computing. In HTTP, a web browser, for example, acts as a client, while an application running on a computer hosting the web site acts as a server. The client submits HTTP requests to the responding server by sending messages to it. The server, which stores content (or resources) such as HTML files and images, or generates such content on the fly, sends messages back to the client in response. These returned messages may contain the content requested by the client or may contain other kinds of response indications. A client is also referred to as a user agent (or 'UA' for short). A web crawler (or 'spider') is another example of a common type of client or user agent.
In between the client and server there may be several intermediaries, such as proxies, web caches

or gateways. In such a case, the client communicates with the server indirectly, and only converses directly with the first intermediary in the chain. A server may be called the origin server to reflect the fact that this is where content ultimately originates from.
HTTP is not constrained in principle to using TCP/IP, although this is its most popular implementation platform. Indeed HTTP can be "implemented on top of any other protocol on the Internet, or on other networks." HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.[2]

Source en.wikipedia.org

In this capture file, HTTP is used to communicate with the GUI frontend of the SIP PBX.

Section 2/ Question 2a. Which codec does the RTP stream use?	Possible Points: 1pt

Tools Used: tshark

Answer

Tshark can give useful informations on RTP streams within a PCAP file:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ tshark -r Forensic_challenge_4.pcap -qz rtp,streams
========================= RTP Streams =========================
    Src IP addr  Port    Dest IP addr  Port       SSRC          Payload  Pkts         Lost   Max
Delta(ms)  Max Jitter(ms) Mean Jitter(ms) Problems?
   172.25.105.3 63184   172.25.105.40 18150 0xA254E017 ITU-T G.711 PCMU  1811   -30 (-1.7%)
1940.06          122.24           11.28 X
  172.25.105.40 18150    172.25.105.3 63184 0x42AFE59B ITU-T G.711 PCMU  1302     0 (0.0%)
56.05            3.43             0.32 X
================================================================
```

These stats indicate that the G.711 µ-law (or u-law) codec was used for the VoIP call. Some good infos on G.711 codecs can be found here: (http://www.en.voipforo.com/codec/codecs-g711-alaw.php)

Section 2/ Question 2b. How long is the sampling time (in milliseconds)?	Possible Points: 1pt
Tools Used:	
Answer G.711 family of codecs use a sampling frequency of 8kHz (8000 Hz). Meaning, the voice or audio stream is sampled 8000 times in one (1) second. So, the sampling time or length of one sample is 1/8000 s = 0.000125 s = 0.125 ms	

Section 2/ Question 3. How did the attacker gain access to the server? List ways this could have been prevented.	Possible Points: 2pts
Tools Used: tshark	

Answer

At the beginning of the attack, the attacker (172.25.105.43) sent a SIP OPTIONS request for extension 100 at 172.25.105.40. (packet #1).
The tool used by the attacker seems to be svmap.py from the SIPvicious suite.
Luckily, 172.25.105.40 responded to the request with a 200 OK response and a lot of information on the targeted extension. (packet #2)
Here an extract of the informations returned:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ tshark -r
Forensic_challenge_4.pcap -VR "frame.number==2"
Frame 2 (560 bytes on wire, 560 bytes captured)
    Arrival Time: May  1, 2010 20:13:00.948226000
    [Time delta from previous captured frame: 0.000353000 seconds]
    [Time delta from previous displayed frame: 0.000353000 seconds]
    [Time since reference or first frame: 0.000353000 seconds]
    Frame Number: 2
    Frame Length: 560 bytes
    Capture Length: 560 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:sip]
Ethernet II, Src: IntelCor_87:cf:96 (00:21:6a:87:cf:96), Dst: IntelCor_9f:78:c6
(00:13:ce:9f:78:c6)
    Destination: IntelCor_9f:78:c6 (00:13:ce:9f:78:c6)
        Address: IntelCor_9f:78:c6 (00:13:ce:9f:78:c6)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
    Source: IntelCor_87:cf:96 (00:21:6a:87:cf:96)
        Address: IntelCor_87:cf:96 (00:21:6a:87:cf:96)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory
default)
    Type: IP (0x0800)
Internet Protocol, Src: 172.25.105.40 (172.25.105.40), Dst: 172.25.105.43
(172.25.105.43)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x60 (DSCP 0x18: Class Selector 3; ECN: 0x00)
        0110 00.. = Differentiated Services Codepoint: Class Selector 3 (0x18)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
```

```
    Total Length: 546
    Identification: 0xca73 (51827)
    Flags: 0x00
        0.. = Reserved bit: Not Set
        .0. = Don't fragment: Not Set
        ..0 = More fragments: Not Set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
    Header checksum: 0x8371 [correct]
        [Good: True]
        [Bad : False]
    Source: 172.25.105.40 (172.25.105.40)
    Destination: 172.25.105.43 (172.25.105.43)
User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
    Source port: 5060 (5060)
    Destination port: 5060 (5060)
    Length: 526
    Checksum: 0x57f4 [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
        Status-Code: 200
        [Resent Packet: False]
        [Request Frame: 1]
        [Response Time (ms): 0]
    Message Header
        Via: SIP/2.0/UDP 127.0.1.1:5060;branch=z9hG4bK-
1453809699;received=172.25.105.43;rport=5060
            Transport: UDP
            Sent-by Address: 127.0.1.1
            Sent-by port: 5060
            Branch: z9hG4bK-1453809699
            Received: 172.25.105.43
            RPort: 5060
        From: "sipvicious"<sip:100@1.1.1.1>;
tag=6163313936393238313363340131323031353530343335
            SIP Display info: "sipvicious"
            SIP from address: sip:100@1.1.1.1
                SIP from address User Part: 100
                SIP from address Host Part: 1.1.1.1
            SIP tag: 6163313936393238313363340131323031353530343335
        To: "sipvicious"<sip:100@1.1.1.1>;tag=as18cdb0c9
            SIP Display info: "sipvicious"
            SIP to address: sip:100@1.1.1.1
                SIP to address User Part: 100
                SIP to address Host Part: 1.1.1.1
            SIP tag: as18cdb0c9
        Call-ID: 6112707879346995719413l
        CSeq: 1 OPTIONS
            Sequence Number: 1
            Method: OPTIONS
        User-Agent: Asterisk PBX 1.6.0.10-FONCORE-r40
        Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
        Supported: replaces, timer
```

```
        Contact: <sip:172.25.105.40>
            Contact Binding: <sip:172.25.105.40>
                URI: <sip:172.25.105.40>
                    SIP contact address: sip:172.25.105.40
        Accept: application/sdp
        Content-Length: 0
```

One information that is particularly useful for the attacker is the User-Agent message header field of the response (highlighted in orange)
Given this information, the attacker now knows that:

- He is facing an Asterisk PBX. (http://www.asterisk.org/)
- the "FONCORE" mention in the UA string could indicate that this is a Tribox distribution (http://www.trixbox.org/)

Tribox systems comes with an easy to use web-based GUI called FreePBX (http://www.freepbx.org/), using it one can controls and manages an entire asterisk-based PBX.

With these clues in hands, our attacker tried to connect to the box with HTTP. (starting at packet #3)
Quickly, he tried to access the "Admin" pages via the /maint URI (packet #6). But, the access to these pages is protected by a login/password pair.
Our attacker guessed that he was facing a poorly secured PBX and tried the default administrative credentials to access /maint (packet #60).

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ tshark -r
Forensic_challenge_4.pcap -VR "frame.number==60"

==== OUTPUT CUT====

[Number of bytes in flight: 519]
Hypertext Transfer Protocol
    GET /maint HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /maint HTTP/1.1\r\n]
            [Message: GET /maint HTTP/1.1\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Request Method: GET
        Request URI: /maint
        Request Version: HTTP/1.1
    Host: 172.25.105.40\r\n
    User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.9) Gecko/20100401
Ubuntu/9.10 (karmic) Firefox/3.5.9\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://172.25.105.40/user/\r\n
    Cookie: lng=en; PHPSESSID=gl8n9gi8r09lke1vb0hpf97na1\r\n
    Authorization: Basic bWFpbnQ6cGFzc3dvcmQ=\r\n
        Credentials: maint:password
    \r\n
```

Basic Authentication was used and the decoding of the base64 encoded string:
bWFpbnQ6cGFzc3dvcmQ=
reveals that the default login/password for a tribox system were tried by the attacker.
We can see in packet #62 that he was right and was given access to the system. At this point, the attacker had total control over the victim's PBX system.

Well, this kind of "attack" could have been prevented by:

- At least, changing the user's "maint" default password !
- Filtering HTTP access to the box.

Section 2/ Question 4. What information was gained by the attacker ?	Possible Points: 2pts

Tools Used: tshark, httpdumper

Answer

At this point of the analysis, the attacker knows:

- The kind of SIP PBX => Asterisk
- The distribution: Tribox family.

Now, it's time to analyze the entire HTTP conversations to evaluate what informations were gained by the attacker.

To do this analysis, I've use one of my own tool called httpdumper (http://malphx.free.fr/dotclear/public/nfpc3/httpdumper). It's a simple ruby that gives some informations on HTTP flows. Httpdumper has a specific option to list all the requested-uri:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ ruby httpdumper -r Forensic_challenge_4.pcap -s uri
Reading file Forensic_challenge_4.pcap
Parsing packets...
4447 packets read in 9.165 sec.

---------------------------------
Listing URI requested ALL clients
---------------------------------
---------------------------------
Requested to
---------------------------------
[conv: 68] [flow: 2]
[conv: 68] [flow: 3]
[conv: 72] [flow: 1]
[conv: 72] [flow: 2]
---------------------------------
Requested to 172.25.105.40
---------------------------------
[conv: 0] [flow: 0] /maint
[conv: 1] [flow: 0] /
[conv: 2] [flow: 0] /user/
[conv: 3] [flow: 0] /maint
[conv: 4] [flow: 0] /maint
[conv: 5] [flow: 0] /maint/
[conv: 6] [flow: 0] /js/scriptaculous.js
[conv: 7] [flow: 0] /maint/js/submitSignup.js
[conv: 8] [flow: 0] /maint/js/iframeSizing.js
[conv: 9] [flow: 0] /maint/js/iframeSizing_freepbx.js
[conv: 10] [flow: 0] /maint/js/main.js
[conv: 11] [flow: 0] /maint/skin/default/css/index_tpl.css
[conv: 12] [flow: 0] /maint/skin/default/cssJavascriptWindows/default.css
[conv: 13] [flow: 0] /maint/skin/default/cssJavascriptWindows/alert.css
[conv: 14] [flow: 0] /maint/js/DHTMLAPI.js
[conv: 15] [flow: 0] /maint/js/javascriptsWindows/window.js
```

```
[conv: 16] [flow: 0] /maint/js/javascriptsWindows/effects.js
[conv: 17] [flow: 0] /maint/js/javascriptsWindows/window_effects.js
[conv: 18] [flow: 0] /maint/js/javascriptsWindows/debug.js
[conv: 19] [flow: 0] /maint/js/javascriptsWindows/popUps.js
[conv: 20] [flow: 0] /maint/js/chromejs/chrome.js
[conv: 21] [flow: 0] /maint/js/packages.js
[conv: 22] [flow: 0] /maint/js/initAnime.js
[conv: 23] [flow: 0] /maint/includes/xajax_js/xajax.js
[conv: 24] [flow: 0] /maint/skin/default/css/style.css
[conv: 25] [flow: 0] /maint/skin/default/css/chrometheme/chromestyle.css
[conv: 26] [flow: 0] /maint/skin/default/css/header.css
[conv: 27] [flow: 0] /maint/skin/default/css/footer.css
[conv: 28] [flow: 0] /maint/skin/default/cssJavascriptWindows/mac_os_x.css
[conv: 29] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube.css
[conv: 30] [flow: 0] /js/prototype.js
[conv: 31] [flow: 0] /js/builder.js
[conv: 32] [flow: 0] /js/effects.js
[conv: 33] [flow: 0] /js/slider.js
[conv: 34] [flow: 0] /js/dragdrop.js
[conv: 35] [flow: 0] /js/controls.js
[conv: 36] [flow: 0] /js/sound.js
[conv: 37] [flow: 0] /maint/skin/default/trixbox_logo.gif
[conv: 38] [flow: 0] /maint/
[conv: 39] [flow: 0] /maint/skin/default/arrow_top.gif
[conv: 40] [flow: 0] /maint/skin/default/emailIcon.gif
[conv: 41] [flow: 0] /maint/skin/default/menu_bar.gif
[conv: 42] [flow: 0] /maint/modules/home/index.php?lang=english
[conv: 43] [flow: 0] /maint/skin/default/barS.jpg
[conv: 44] [flow: 0] /maint/skin/default/loading_image.gif
[conv: 45] [flow: 0] /maint/modules/home/templates/classic/classic.css
[conv: 46] [flow: 0] /maint/modules/home/templates/classic/images/bar_right.gif
[conv: 47] [flow: 0] /maint/modules/home/templates/classic/images/bar_left.gif
[conv: 48] [flow: 0] /maint/modules/home/templates/classic/images/bar_middle.gif
[conv: 49] [flow: 0] /maint/modules/home/templates/classic/images/barS.jpg
[conv: 50] [flow: 0] /maint/modules/registrationTool/index.php
[conv: 51] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/top-middle.gif
[conv: 52] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/bottom-middle.gif
[conv: 53] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/frame-right.gif
[conv: 54] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/frame-left.gif
[conv: 55] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/button-close-focus.gif
[conv: 56] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/button-max-focus.gif
[conv: 57] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/button-min-focus.gif
[conv: 58] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/left-top.gif
[conv: 59] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/right-top.gif
[conv: 60] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/bottom-left-c.gif
[conv: 61] [flow: 0] /maint/modules/registrationTool/js/tb_reg.js
[conv: 62] [flow: 0] /maint/skin/default/cssJavascriptWindows/alphacube/bottom-right-c.gif
[conv: 63] [flow: 0] /maint/modules/registrationTool/images/registration.png
[conv: 64] [flow: 0] /maint/index.php?freepbx
[conv: 65] [flow: 0] /maint/index.php?freepbx
[conv: 66] [flow: 0] /admin
[conv: 67] [flow: 0] /admin/
[conv: 68] [flow: 0] /admin/config.php
[conv: 69] [flow: 0] /admin/common/mainstyle.css
[conv: 70] [flow: 0] /admin/config.php?handler=file&module=dashboard&file=dashboard.css
[conv: 71] [flow: 0] /admin/common/script.js.php
[conv: 72] [flow: 0] /admin/common/libfreepbx.javascripts.js
[conv: 73] [flow: 0] /admin/common/stylesheet_custom.css
[conv: 74] [flow: 0] /admin/images/
[conv: 75] [flow: 0] /admin/images/logo.png
[conv: 76] [flow: 0] /admin/images/tab.png
[conv: 77] [flow: 0] /admin/images/shadow-side-background.png
[conv: 78] [flow: 0] /admin/images/notify_update.png
[conv: 79] [flow: 0] /admin/images/notify_delete.png
[conv: 80] [flow: 0] /admin/images/notify_error.png
[conv: 81] [flow: 0] /admin/images/notify_notice.png
[conv: 82] [flow: 0] /admin/images/cancel.png
[conv: 83] [flow: 0] /admin/images/modules-hover1.png
[conv: 84] [flow: 0] /admin/config.php?
```

```
type=tool&display=index&quietmode=1&info=stats&restrictmods=core/dashboard
[conv: 85] [flow: 0] /admin/config.php?
type=tool&display=index&quietmode=1&info=stats&restrictmods=core/dashboard
[conv: 86] [flow: 0] /maint/index.php?configEdit
[conv: 87] [flow: 0] /maint/index.php?configEdit
[conv: 88] [flow: 0] /maint/modules/configedit/phpconfig.php
[conv: 89] [flow: 0] /maint/modules/configedit/images/spacer.gif
[conv: 90] [flow: 0] /maint/modules/configedit/images/barS.jpg
[conv: 91] [flow: 0] /maint/modules/configedit/phpconfig.php?
file=sip_custom.conf&section=sip_custom.conf
```

The listing above list the request-uri in the order where they have been requested. We could analyze the progression of the attacker in the FreePBX GUI. One interesting thing is the last requested URI (conv 91).
This URI is: /maint/modules/configedit/phpconfig.php?
file=sip_custom.conf&section=sip_custom.conf
By requesting this URI, the attacker is trying to access the informations stored in this configuration file.

sip_custom.conf is a configuration file that can be used on asterisk system to define SIP extensions.

Let's try to view what information the attacker has gained by dumping the PBX response in HTTP conversation #91:

```
franck@ODIN:~/Analysis/Sources/Honeynet/Challenge 4$ ruby httpdumper -r Forensic_challenge_4.pcap -c 91
Reading file Forensic_challenge_4.pcap
Parsing packets...
4447 packets read in 9.504 sec.

FLOWS TABLE
+------------+-----------------------------------------+-----------------+-----------------------------------------------------------------------------------------+--------------------+
| Flow Index | Hosts                                   | HTTP message type | HTTP Request or Content type                                                          | HTTP Content Length |
+------------+-----------------------------------------+-----------------+-----------------------------------------------------------------------------------------+--------------------+
| 0          | 172.25.105.43:57177 -> 172.25.105.40:80 | REQUEST           | /maint/modules/configedit/phpconfig.php?file=sip_custom.conf&section=sip_custom.conf | 0                  |
| 1          | 172.25.105.40:80 -> 172.25.105.43:57177 | RESPONSE          | text/html;                                                                            | 7000               |
+------------+-----------------------------------------+-----------------+-----------------------------------------------------------------------------------------+--------------------+
```

A 7 KB HTML file was receive. Obviously, this file contains all the HTML stuff to render the page in the browser, but more interesting, we can also view, what information the attacker has accessed:
(see next page)

The attacker has gained some valuable information on 2 SIP extensions: 555 and 556
The most valuable informations being:

- Extensions numbers : 555 and 556
- Username and password used to register extensions : 555/1234

After having taken control over the SIP PBX by guessing the administrator credentials, our attacker has now (at least) all the informations he needs to register and use maliciously the 2 extensions listed above. But he also has total control on the Asterisk system and can potentially do anything he wants to.

Section 2/ Question 5a. The PCAP includes a (not so) hidden bonus! [hint1: You can't read it in the pcap, hint2: It's a city with an active honeynet chapter] a) Describe it, and explain how you found it.	Possible Points: 10pts
Tools Used: wireshark	

Answer

Quick answer: “the secret password is: MEXICO”

Following the HTTP flows between the attacker and the Asterisk PBX. A new host 172.25.105.3 successfully registered the 555 extension. (packet #1294 - #1297). (It can be the attacker again, registering the extension from a new host, an accomplice of the attacker or a legitimate user.)

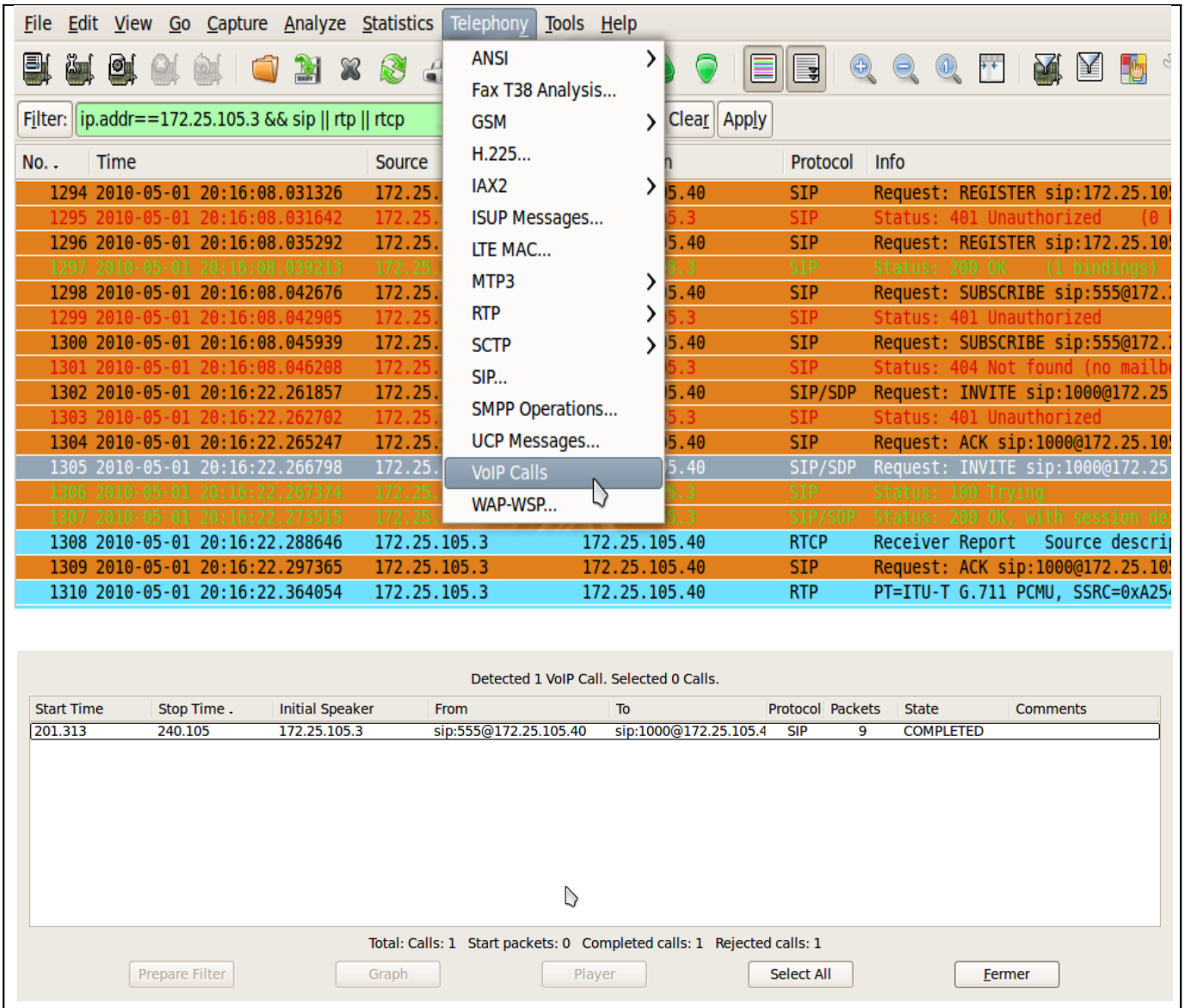
No. .	Time	Source	Destination	Protocol	Info
1294	2010-05-01 20:16:08.031326	172.25.105.3	172.25.105.40	SIP	Request: REGISTER sip:172.25.105.40
1295	2010-05-01 20:16:08.031642	172.25.105.40	172.25.105.3	SIP	Status: 401 Unauthorized (0 bindings)
1296	2010-05-01 20:16:08.035292	172.25.105.3	172.25.105.40	SIP	Request: REGISTER sip:172.25.105.40
1297	2010-05-01 20:16:08.039213	172.25.105.40	172.25.105.3	SIP	Status: 200 OK (1 bindings)
1298	2010-05-01 20:16:08.042676	172.25.105.3	172.25.105.40	SIP	Request: SUBSCRIBE sip:555@172.25.105.40
1299	2010-05-01 20:16:08.042905	172.25.105.40	172.25.105.3	SIP	Status: 401 Unauthorized
1300	2010-05-01 20:16:08.045939	172.25.105.3	172.25.105.40	SIP	Request: SUBSCRIBE sip:555@172.25.105.40
1301	2010-05-01 20:16:08.046208	172.25.105.40	172.25.105.3	SIP	Status: 404 Not found (no mailbox)
1302	2010-05-01 20:16:22.261857	172.25.105.3	172.25.105.40	SIP/SDP	Request: INVITE sip:1000@172.25.105.40, with session description
1303	2010-05-01 20:16:22.262702	172.25.105.40	172.25.105.3	SIP	Status: 401 Unauthorized
1304	2010-05-01 20:16:22.265247	172.25.105.3	172.25.105.40	SIP	Request: ACK sip:1000@172.25.105.40
1305	2010-05-01 20:16:22.266798	172.25.105.3	172.25.105.40	SIP/SDP	Request: INVITE sip:1000@172.25.105.40, with session description
1306	2010-05-01 20:16:22.267374	172.25.105.40	172.25.105.3	SIP	Status: 100 Trying
1307	2010-05-01 20:16:22.273515	172.25.105.40	172.25.105.3	SIP/SDP	Status: 200 OK, with session description
1309	2010-05-01 20:16:22.297365	172.25.105.3	172.25.105.40	SIP	Request: ACK sip:1000@172.25.105.40
4444	2010-05-01 20:17:01.052671	172.25.105.3	172.25.105.40	SIP	Request: BYE sip:1000@172.25.105.40
4446	2010-05-01 20:17:01.053304	172.25.105.40	172.25.105.3	SIP	Status: 200 OK

Then 172.27.105.3 tries to establish a call to 1000@172.25.105.40 via a SIP INVITE message (packet #1302 and #1305)

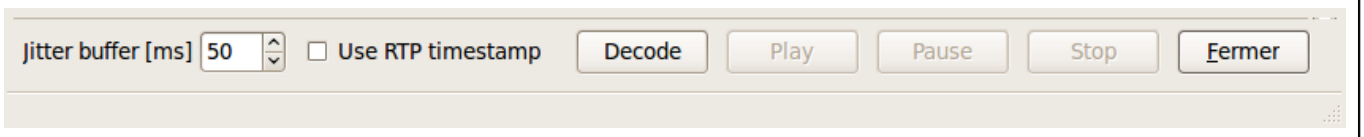
The call confirmation is done by the IPBX, first by sending a 100 Trying response code (packet #1306) and then a 200 OK response with SDP informations in packet #1307

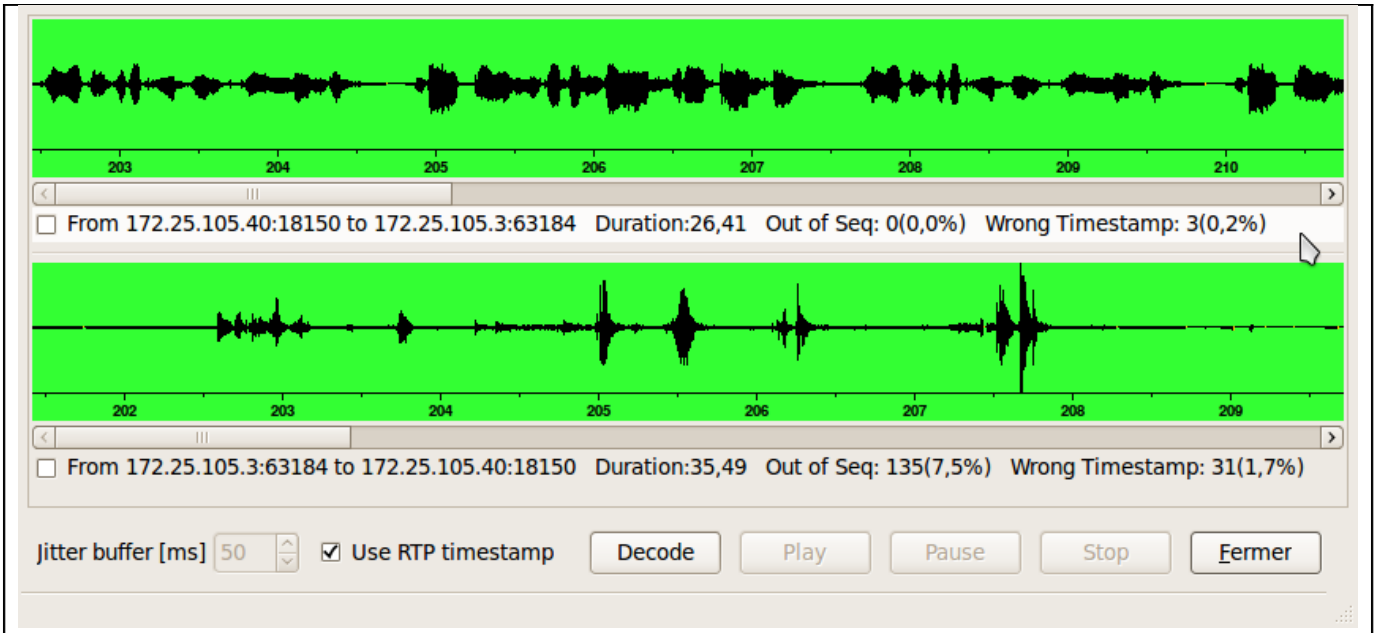
Informations exchanged in SDP (Session Description Protocol) in packet #1305 and #1307 define all the parameters of the audio communication that will start between 555@172.25.105.40 and 1000@172.25.105.40.

Using the VoIP Call option of the Wireshark's Telephony menu, it is possible to list all the VoIP call contained in a pcap file.



Wireshark offer a simple player capable of reading unencrypted RTP audio streams. This player can be launched by selecting a VoIP call and pressing the “Player”button.
The next window is used to adjust the jitter buffer manually or to use RTP timestamps stored in RTP packets to decode the RTP streams. Select “Use RTP timestamp” and press “Decode”





The RTP player's window appears and now to listen to the audio stream, just select one or both of the audio streams and press "Play".

The audio stream reveals that the user is trying to access an audio conference room (1000) and after having typed 2 bad pin numbers he finally accessed the conference. The user was the only person in this conference.

After some times in the conference room, the user said:

"Congratulations, you're listening to an unencrypted VoIP call. The secret password is Mexico, so write it down and submit your challenge. Good job, thank you!"

Section 2/ Question 5b. If VOIP packets between the two calling parties traverse an untrusted network (eg the wireless/internet) and a similar PCAP was captured by a malicious party, would you think this a security problem? why?	Possible Points: 3pts
Tools Used:	

Answer

Yes it is.
For obvious reason, if the audio stream is not encrypted , an attacker can eavesdrop on the conversation and some business-critical informations can be recorded or stolen. DTMF tones = passwords/pin numbers can also be stolen (heard).

The signaling protocol messages (here SIP) also travel unencrypted (in our case) and can give to an attacker some valuable informations on the calling parties, like:

- Extension numbers
- Call-ID values
- Cseq values
- Authentication digest
  - That can be brute-forced off-line with tools like SIPcrack (http://www.darknet.org.uk/2008/08/sipcrack-sip-login-dumper-hashpassword-cracker/)
  - That can be used in attacks which need some packets to be replayed, like the "SIP

unregister attack" (http://www.idc.ac.il/publications/files/238.pdf)

So, If VoIP packets need to traverse an untrusted network, security must be used for the signaling protocol (SIP over TLS / Secure SIP) and for the media stream (SRTP, ZRTP...)

Section 2/ Question 5c. Wireshark has an option "Use RTP timestamp". What is the function of this option?	Possible Points: 2pts
Tools Used:	

Answer

This is an option of the Wireshark's RTP Player. It is used to decode and play an RTP stream based on RTP timing stored in RTP packet instead of on packet arrival time. Each RTP packets includes a timestamp which define the sampling instant of the first octet of the data packet.
Quote from RFC 3550 (RTP): *The timestamp reflects the sampling instant of the first octet in the RTP data packet.*

This feature is useful when the original IP/RTP packets have been encapsulated or tunneled and original timing is lost. In this case Wireshark will use RTP timestamp values to order and decode the audio stream.
When using this feature Wireshark cannot simulate the jitter buffer and so, this option is grayed out.

One drawback of this feature is that the RTP player doesn't render the audio as the calling parties have heard it.

Section 2/ Question 6. What technologies or protocols can be used to protect confidentiality of RTP traffic as it traverses untrusted networks.	Possible Points: 3pts
Tools Used:	

Answer

Multiples solutions exists to protect and secure RTP exchanges. They mostly rely on message authentication, encryption. Here are some examples of such technologies:

- SRTP (RFC 3711) can be used to protect RTP traffic. It's an RTP profile which provides confidentiality (through encryption), message authentication and replay protection to the RTP and RTCP traffic.

- ZRTP (http://tools.ietf.org/html/draft-zimmermann-avt-zrtp-21) which provides a key-agreement protocol to exchange key informations (using Diffie-Hellman exchange) between calling parties in RTP packets (in-band). Then ZRTP uses SRTP to secure the data stream.

- Using a protocol like RTSP which provides a way to multiplex data and control in a single stream (RTSP + RTP data) supported by an unique TCP connection. This connection can then be secured using TLS hence offering the expected confidentiality.

- RTP can also be protected with the security offered by the network layer (by the use of IPsec for example).

A good reference on this subject can be find in this document:
http://tools.ietf.org/html/draft-ietf-avt-srtp-not-mandatory-05

Section 3/ Question 1. What is "RTP injection" and describe how it functions. What conditions are required to allow this?	Possible Points: 2pts
Tools Used:	

Answer

RTP injection is a kind of attack where the attacker is able to inject or mix RTP packets in an on-going call between two parties. One objective of this attack can be to diffuse “SPIT” (SPams over Internet Telephony) by injecting a pre-recorded audio message in an established VoIP call. This attack targets only the media protocol (RTP) and hence is totally independent of the signaling protocol used to setup the call.

Nevertheless, RTP injection is only possible when some specific conditions are met:

- The targeted RTP stream must be unencrypted.
- The use of UDP protocol as transport protocol for RTP
- The attacker must be able to capture at least one valid RTP packet from the stream. This packet will be used as a template to construct the spoofed RTP packets that will be later injected in the stream.
- From this packet, the attacker has to get critical informations on the stream to successfully inject RTP packets. These informations are:
  - The payload type
    - needed to send correctly encoded audio data.
  - The RTP Sequence number
    - Will be set in the spoofed packets to a higher value than the legitimate RTP packets. This will force the receiver “thinking” they are older than the spoofed ones and hence will be dropped.
  - The RTP timestamps
    - in the same fashion as sequence numbers, timestamp will be set to an higher value than the legitimate packets.
  - Synchronization Source Identifier (SSRC)
    - This value remain the same during all the call. So the attacker has just to set SSRC with the same value as the captured packet's SSRC field.
  - IP ID
    - Again the IP ID will be set to a higher value than the legitimate RTP/IP packets.

If all this conditions are met, the attacker should be able to correctly craft RTP packets and to inject them in the on-going call.

Using a secure media protocol, like SRTP or ZRTP, prevent this kind of attack.

Section 3/ Question 2. Explain how a SIP password digest could be intercepted or stolen. Is this a security issue? why or why not.	Possible Points: 2pts
Tools Used:	

Answer

At least two ways can be used to intercept and steal SIP password digest:

- By Sniffing SIP traffic
  - Attacker can take control of a poorly secure switch (password brute-force, exploit, social engineering...) and the configure traffic mirroring
  - Attacker can have previously attacked a poorly secure Wireless LAN and then can sniff traffic “over the air”.
  - Intrusive: attacker can insert a hub or a Pc with two NIC cards on the traffic path ...

- By Redirecting SIP traffic flowing between a client and a server to the attacker using Man-in-the-Middle (MitM) attack.
  - On today's switched networks, an attacker cannot easily eavesdrops on the traffic not destined to him. So, he has to use traffic diversion by launching a MitM attack against a SIP client and a SIP proxy for example.
    - DNS entry modification to divert traffic to the attacker host
    - ARP spoofing (gratuitous ARP) directed to a client and associating the IP address of the SIP server to the attacker's MAC address
    - Flooding switches with lot of unknown MAC address to exhaust CAM table and force the switch to broadcast all the packets over all the ports...

  - If attack is successful, the attacker will be able to eavesdrop on SIP traffic (and surely other kind of network traffic flowing between the two parties) and steal password digest.

In the case where SIP traffic travels unencrypted on the network, these two ways can give to an attacker the possibility of stealing SIP password digest. Then, he will be able to use them in some “replay attacks” or try to brute-force them with specialized offline cracking tools (e.g sipcrack). So, this can be considered as a security issue.

However, use of the secured version of SIP, SIP over TLS, may thwart these attacks.

Section 3/ Question 3. Is DDoS a threat to VOIP systems? Are there any general functional requirements of telephony systems that would be impaired by a DDoS?	Possible Points: 2pts
Tools Used:	

Answer

DDoS stands for Distributed Deny of Service.

It is an attempt to make a particular service, offered by a server, unavailable to its legitimate users by denying it or bringing the server that offers the service down (Crash, reboot-loop...).
“Distributed” involves the use of a medium to large amount of previously compromised computers (e.g Zombies and BotNets) to launch a synchronized attack against an unique target. The primary objective is to exhaust server resources, thus making it unable to process legitimate user's requests.

VoIP systems like any other “servers” need resources to make their jobs (Calls handling for example), these resources are CPU, memory, network bandwidth...
This is making them vulnerable to DDoS attacks.

Some examples of such attacks are:

- Flooding a SIP proxy with SIP REGISTER or SIP INVITE messages, making him unable to process legitimate calls or user's requests.
- Exhausting resources by sending large amount of SIP REGISTER messages to extensions that need authentication. (Database lookup)