SimpleSPA

Applications Contents:

- VPNFinal.jar (this is the jar of the server side of the application)
- SPASimpleCSCompanion.jar (this is the jar of the client side of the app for Linux)
- SPASimpleCSCompanion_windows.jar (this is the jar of the client side of the app for Windows)
- sampleSPA.conf
- source code files

SimpleSPA Application Overview:

This application consists of a single packet authorization mechanism designed for the purpose of hiding semi-public services like a SSH server. There is a server side (Linux only) and a client side (Windows and Linux). This app is similar to FWKnop and more of an academic/proof of concept app as opposed to full blown commercial quality app. It has however been tested extensively and I use it regularly as I travel frequently for work. I use it in conjunction with my SSH server. I keep all ports closed on the iptables firewall and allow SimpleSPA to briefly open the SSH port to allow for a connection and then close the port to new connections.

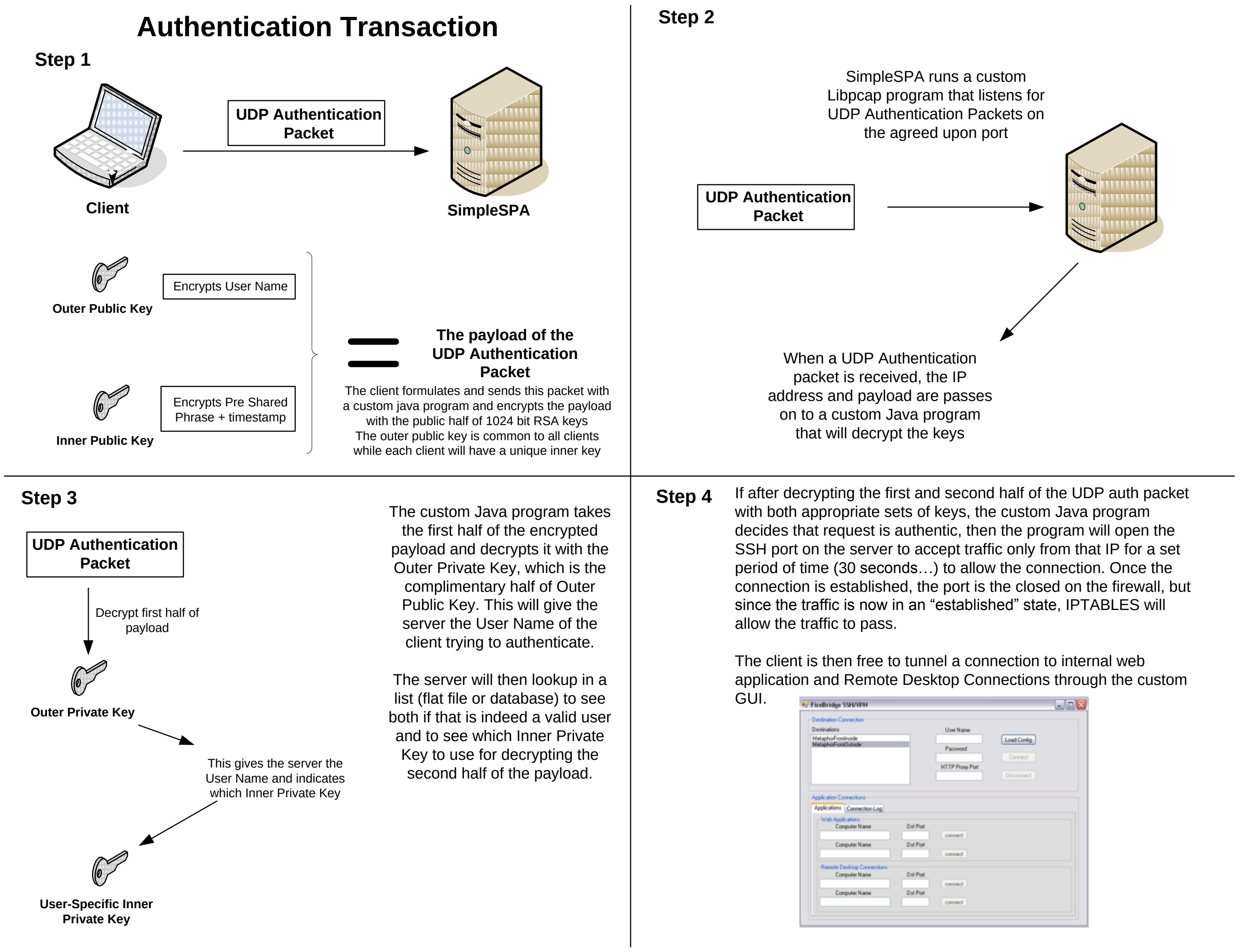
The app involves a client that creates a packet with a payload encrypted with the public half of two different RSA keys. The idea is that one key would be shared by all users and it would encrypt the user name of the individual. A second key specific to each individual user would encrypt a pre-shared key (just any old string, nothing secret about it really) and a timestamp (to counter replay attacks). The server would receive this packet and decrypt this first half of the packet…which would give us the user name of the person sending the packet. The server would then know which user specific second key to use to decrypt the pre-shared key and time stamp to evaluate them for acceptability. If all is good, then the server would open up a port for the semi-public service we were trying to conceal for a brief amount of time to allow for a connection to be made.

Upon receiving a successful packet, an iptables rule will be inserted allowing new connections for the port specified in the conf file. This assumes (probably should be parameterized) that iptables is located at /sbin/iptables. The type of rule inserted (again should also be parameterized, maybe next release) is like the following:

```
$IPTABLES –D –i eth0 –s <sourceIP> -p tcp –dport <port> -m state –
state NEW,ESTABLISHED,RELATED -j ACCEPT
```

Another assumption I make is that your iptables firewall will have some kind of rule to allow established traffic to pass. The idea is to let SimpleSPA briefly open a port, let the user make the connection and then close the port, at least for new connections. In my firewall, I have rules like the following to allow for established traffic to pass.

```
#Allow established connections:
$IPTABLES -A INPUT   -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A OUTPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```


Authentication Transaction
Step 1
UDP Authentication Packet
Client
SimpleSPA
Outer Public Key
Encrypts User Name
Inner Public Key
Encrypts Pre Shared Phrase + timestamp
The payload of the UDP Authentication Packet
Step 2
UDP Authentication Packet
Step 3
UDP Authentication Packet
Decrypt first half of payload
Outer Private Key
This gives the server the User Name and indicates which Inner Private Key
User-Specific Inner Private Key
Step 4

The client formulates and sends this packet with a custom java program and encrypts the payload with the public half of 1024 bit RSA keys
The outer public key is common to all clients while each client will have a unique inner key

SimpleSPA runs a custom Libpcap program that listens for UDP Authentication Packets on the agreed upon port

When a UDP Authentication packet is received, the IP address and payload are passes on to a custom Java program that will decrypt the keys

The custom Java program takes the first half of the encrypted payload and decrypts it with the Outer Private Key, which is the complimentary half of Outer Public Key. This will give the server the User Name of the client trying to authenticate.

The server will then lookup in a list (flat file or database) to see both if that is indeed a valid user and to see which Inner Private Key to use for decrypting the second half of the payload.

If after decrypting the first and second half of the UDP auth packet with both appropriate sets of keys, the custom Java program decides that request is authentic, then the program will open the SSH port on the server to accept traffic only from that IP for a set period of time (30 seconds…) to allow the connection. Once the connection is established, the port is the closed on the firewall, but since the traffic is now in an "established" state, IPTABLES will allow the traffic to pass.

The client is then free to tunnel a connection to internal web application and Remote Desktop Connections through the custom GUI.

Step 4: Not included in this release.

Steps for Setup and Usage:

On the server, you will need to be running iptables at /sbin….not sure if any other Linux variants run it in other places, but BT4 runs it there so its good enough for me. You will also need to install the fun Sun JDK6. The repository version is fine and the apt-get command is shown below. You will also need to install a .de file the folks who wrote Jpcap (http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/download.html), which is a Java wrapper/API for capturing packets. Yes the listener component of this app could easily have been written in C using Libpcap libraries. I did in fact do it this way initially, but quickly found that I am a better Java developer than I am a C developer.

Additionally you will need a jar from the the Bouncy Castle folks (http://www.bouncycastle.org/download/bcprov-jdk16-145.jar) who provide crypto libraries for creating the RSA keys for encrypting the payload of the single auth packet. The link for version 145 is posted above. I've been using version 138, cause I developed it a while ago. I will test sometime soon to make sure there are adverse consequences to using the latest version.

Server side config:

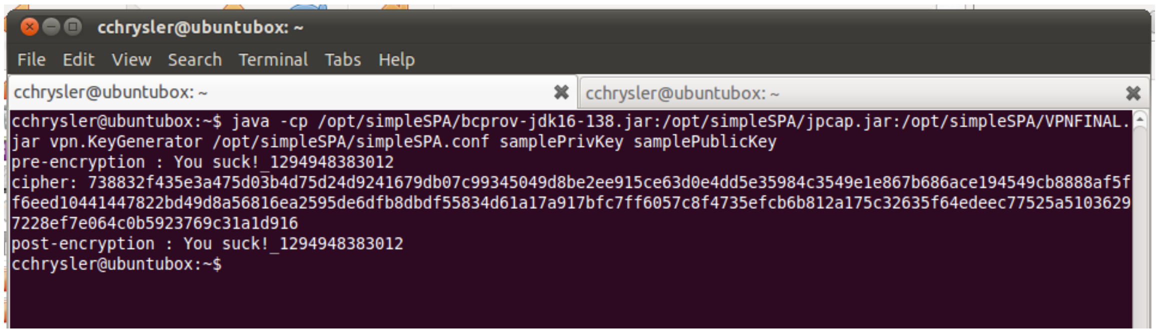
1. apt-get install sun-java6-jdk
2. dpkg --install jpcap-0.7.deb

Server side usage:

1. Create the conf file with all the appropriate variables.
2. Create a pair of keys of keys that all users will use and then a pair of keys for each specific user. Remember that the payload of single auth packet will contain strings encrypted with the public half of each of the keys.
3. Distribute the public key halves to the whoever the client will be and place the private keys in a file specified in the conf file that will contain string entries pairing up the user names to the private key file location (example. Bob.smith=/opt/simpleSPA/users/bobPrivKey)

Linux key generator call sample:

```
 $ java –cp <location>bcprov-jdk16-138.jar:<location>VPNFINAL.jar   vpn.KeyGenerator   <conf file location>  <private key name>  <public key name>
```

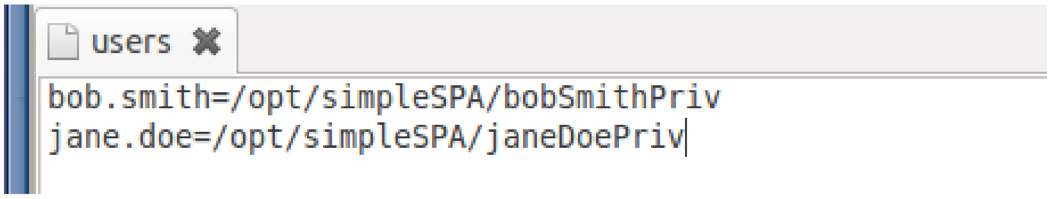


```
cchrysler@ubuntubox:~$ java -cp /opt/simpleSPA/bcprov-jdk16-138.jar:/opt/simpleSPA/jpcap.jar:/opt/simpleSPA/VPNFINAL.
jar vpn.KeyGenerator /opt/simpleSPA/simpleSPA.conf samplePrivKey samplePublicKey
pre-encryption : You suck!_1294948383012
cipher: 738832f435e3a475d03b4d75d24d9241679db07c99345049d8be2ee915ce63d0e4dd5e35984c3549e1e867b686ace194549cb8888af5f
f6eed10441447822bd49d8a56816ea2595de6dfb8dbdf55834d61a17a917bfc7ff6057c8f4735efcb6b812a175c32635f64edeec77525a5103629
7228ef7e064c0b5923769c31a1d916
post-encryption : You suck!_1294948383012
cchrysler@ubuntubox:~$
```

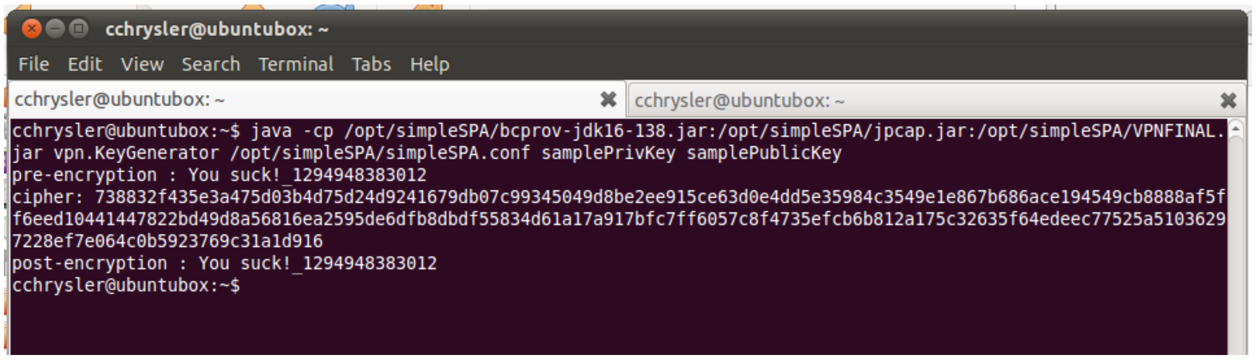
4. Setup the conf file.
5. Fire up the server. Write a script that will check if the process is running and restart it if by chance happened to crash….or else you will locked out of your SSH, or whatever service you are hiding ☺

Linux SimpleSPA call sample:

```
$ java –cp <location>bcprov-jdk16-138.jar:<location>VPNFINAL.jar   vpn.KeyGenerator   <conf file location>  <private key name>  <public key name>
```

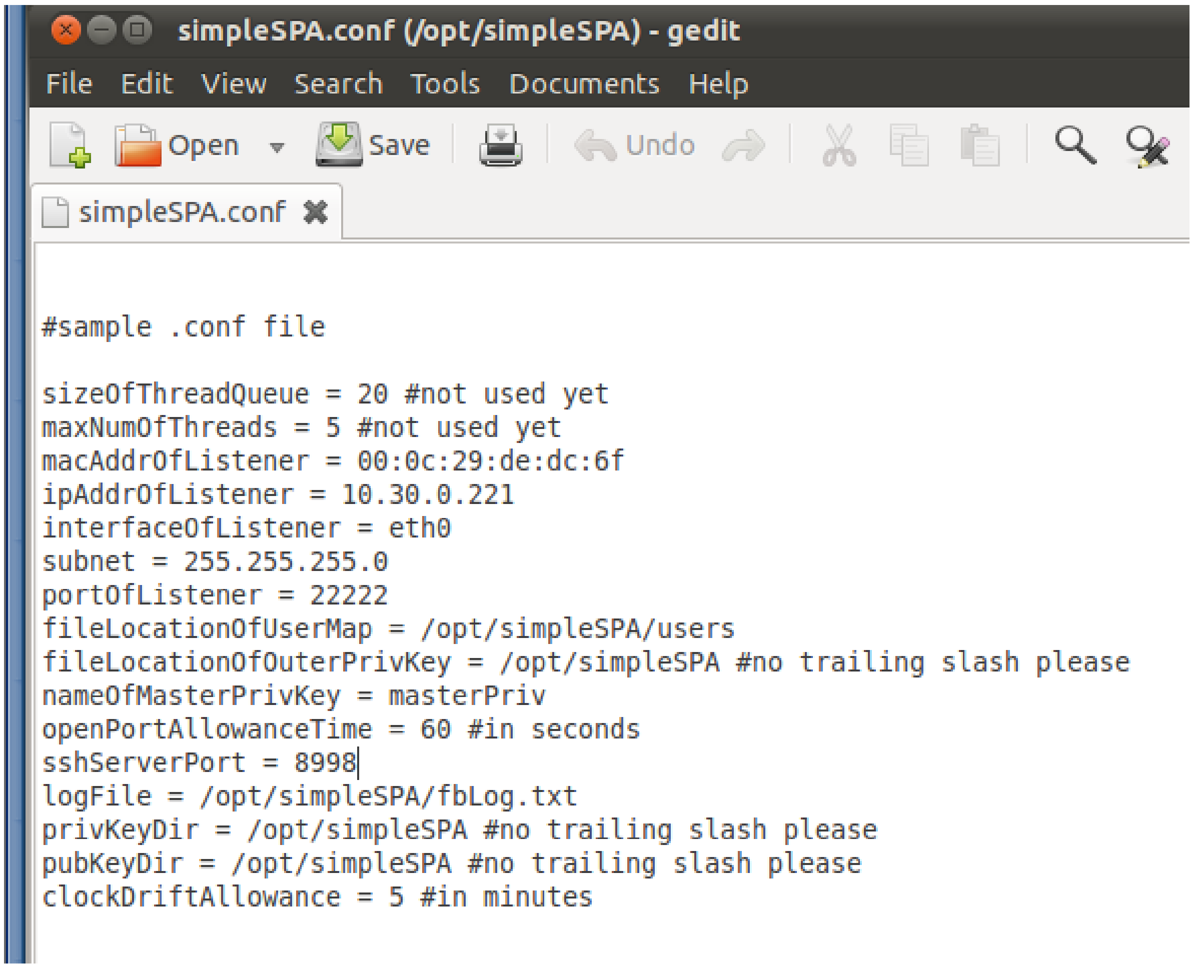


```
bob.smith=/opt/simpleSPA/bobSmithPriv
jane.doe=/opt/simpleSPA/janeDoePriv
```



```
cchrysler@ubuntubox:~$ java -cp /opt/simpleSPA/bcprov-jdk16-138.jar:/opt/simpleSPA/jpcap.jar:/opt/simpleSPA/VPNFINAL.
jar vpn.KeyGenerator /opt/simpleSPA/simpleSPA.conf samplePrivKey samplePublicKey
pre-encryption : You suck!_1294948383012
cipher: 738832f435e3a475d03b4d75d24d9241679db07c99345049d8be2ee915ce63d0e4dd5e35984c3549e1e867b686ace194549cb8888af5f
f6eed10441447822bd49d8a56816ea2595de6dfb8dbdf55834d61a17a917bfc7ff6057c8f4735efcb6b812a175c32635f64edeec77525a5103629
7228ef7e064c0b5923769c31a1d916
post-encryption : You suck!_1294948383012
cchrysler@ubuntubox:~$
```

Sample Conf file


simpleSPA.conf (/opt/simpleSPA) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
simpleSPA.conf

```
#sample .conf file

sizeOfThreadQueue = 20 #not used yet
maxNumOfThreads = 5 #not used yet
macAddrOfListener = 00:0c:29:de:dc:6f
ipAddrOfListener = 10.30.0.221
interfaceOfListener = eth0
subnet = 255.255.255.0
portOfListener = 22222
fileLocationOfUserMap = /opt/simpleSPA/users
fileLocationOfOuterPrivKey = /opt/simpleSPA #no trailing slash please
nameOfMasterPrivKey = masterPriv
openPortAllowanceTime = 60 #in seconds
sshServerPort = 8998
logFile = /opt/simpleSPA/fbLog.txt
privKeyDir = /opt/simpleSPA #no trailing slash please
pubKeyDir = /opt/simpleSPA #no trailing slash please
clockDriftAllowance = 5 #in minutes
```

Windows Client Side Usage

The Windows side is very straight forward.

Windows client side call sample:

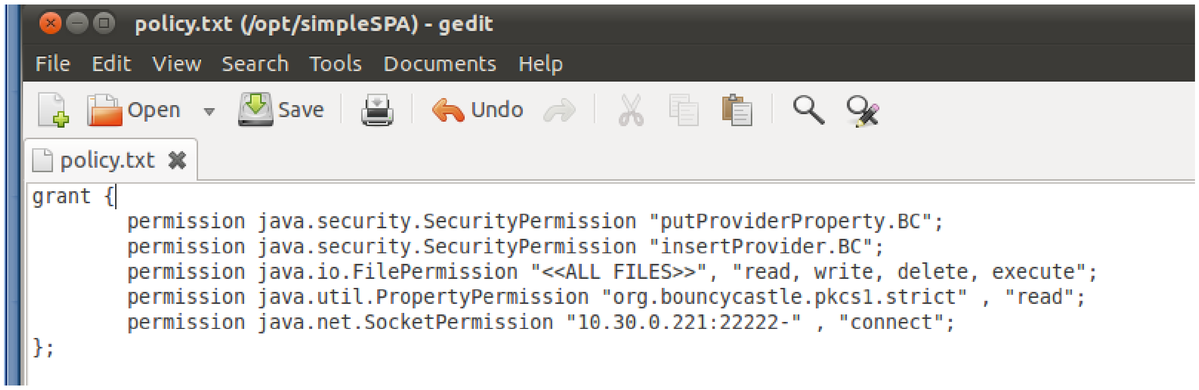
```
c:\>java –cp <location>\bcprov-jdk16-138.jar;<location>\SPASimpleCSCompanion_windows.jar
spa.UDPPacketAuthStandAlone_v2  <source IP> <master public key>  <user specific public key>
<username>  <source port>  <destination port>  <destination IP>
```

Linux Client Side Usage:

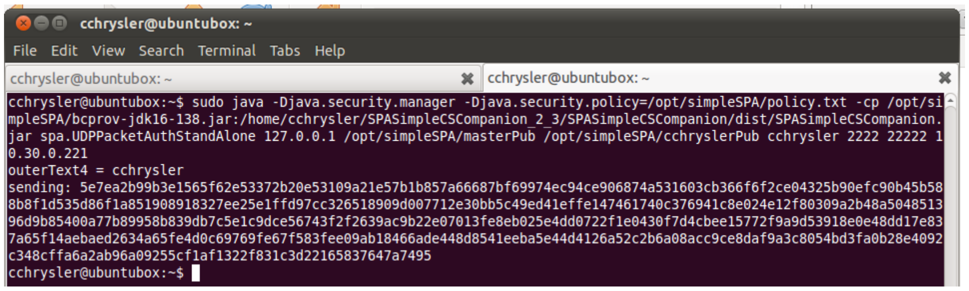
Linux is a bit more picky with the client side. You have to specify a policy explicating stating some permissions. Maybe there is a better way to do this, I don't deal with permissions and policy files very often.

Linux client side call sample:

```
 $ java –Djava.security.manager  -Djava.security.policy=<location>  –cp <location>bcprov-jdk16-
138.jar:<location>SPASimpleCSCompanion.jar   spa.UDPPacketAuthStandAlone  <source IP>
<master public key>  <user specific public key>  <username>  <source port>  <destination port>
<destination IP>
```

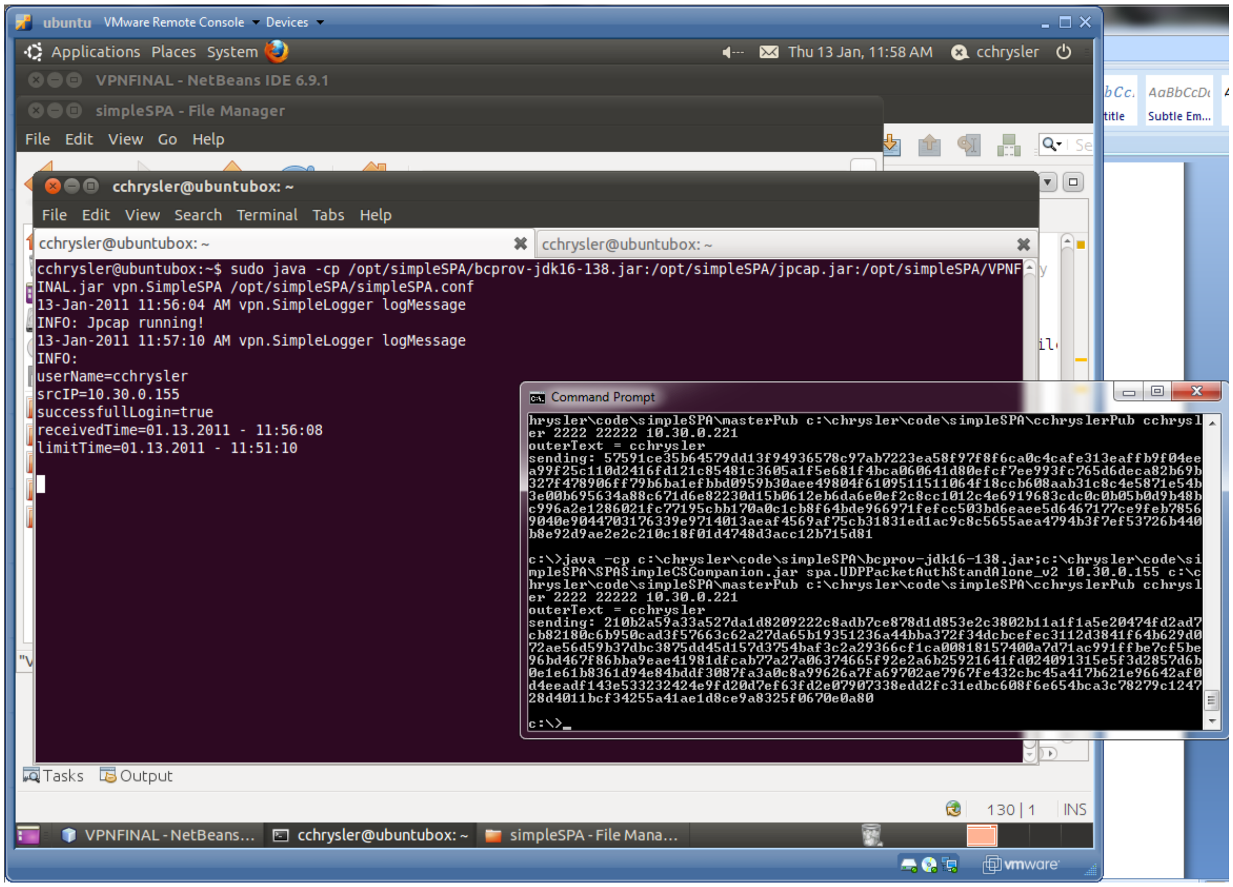

policy.txt (/opt/simpleSPA) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
policy.txt

```
grant {
        permission java.security.SecurityPermission "putProviderProperty.BC";
        permission java.security.SecurityPermission "insertProvider.BC";
        permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete, execute";
        permission java.util.PropertyPermission "org.bouncycastle.pkcs1.strict" , "read";
        permission java.net.SocketPermission "10.30.0.221:22222-" , "connect";
};
```



```
cchrysler@ubuntubox:~$ sudo java -Djava.security.manager -Djava.security.policy=/opt/simpleSPA/policy.txt -cp /opt/si
mpleSPA/bcprov-jdk16-138.jar:/home/cchrysler/SPASimpleCSCompanion_2_3/SPASimpleCSCompanion/dist/SPASimpleCSCompanion.
jar spa.UDPPacketAuthStandAlone 127.0.0.1 /opt/simpleSPA/masterPub /opt/simpleSPA/cchryslerPub cchrysler 2222 22222 1
0.30.0.221
outerText4 = cchrysler
sending: 5e7ea2b99b3e1565f62e53372b20e53109a21e57b1b857a66687bf69974ec94ce906874a531603cb366f6f2ce04325b90efc90b45b58
8b8f1d535d86f1a851908918327ee25e1ffd97cc326518909d007712e30bb5c49ed41effe147461740c376941c8e024e12f80309a2b48a5048513
96d9b85400a77b89958b839db7c5e1c9dce56743f2f2639ac9b22e07013fe8eb025e4dd0722f1e0430f7d4cbee15772f9a9d53918e0e48dd17e83
7a65f14aebaed2634a65fe4d0c69769fe67f583fee09ab18466ade448d8541eeba5e44d4126a52c2b6a08acc9ce8daf9a3c8054bd3fa0b28e4092
c348cffa6a2ab96a09255cf1af1322f831c3d22165837647a7495
cchrysler@ubuntubox:~$
```

The whole thing in action:



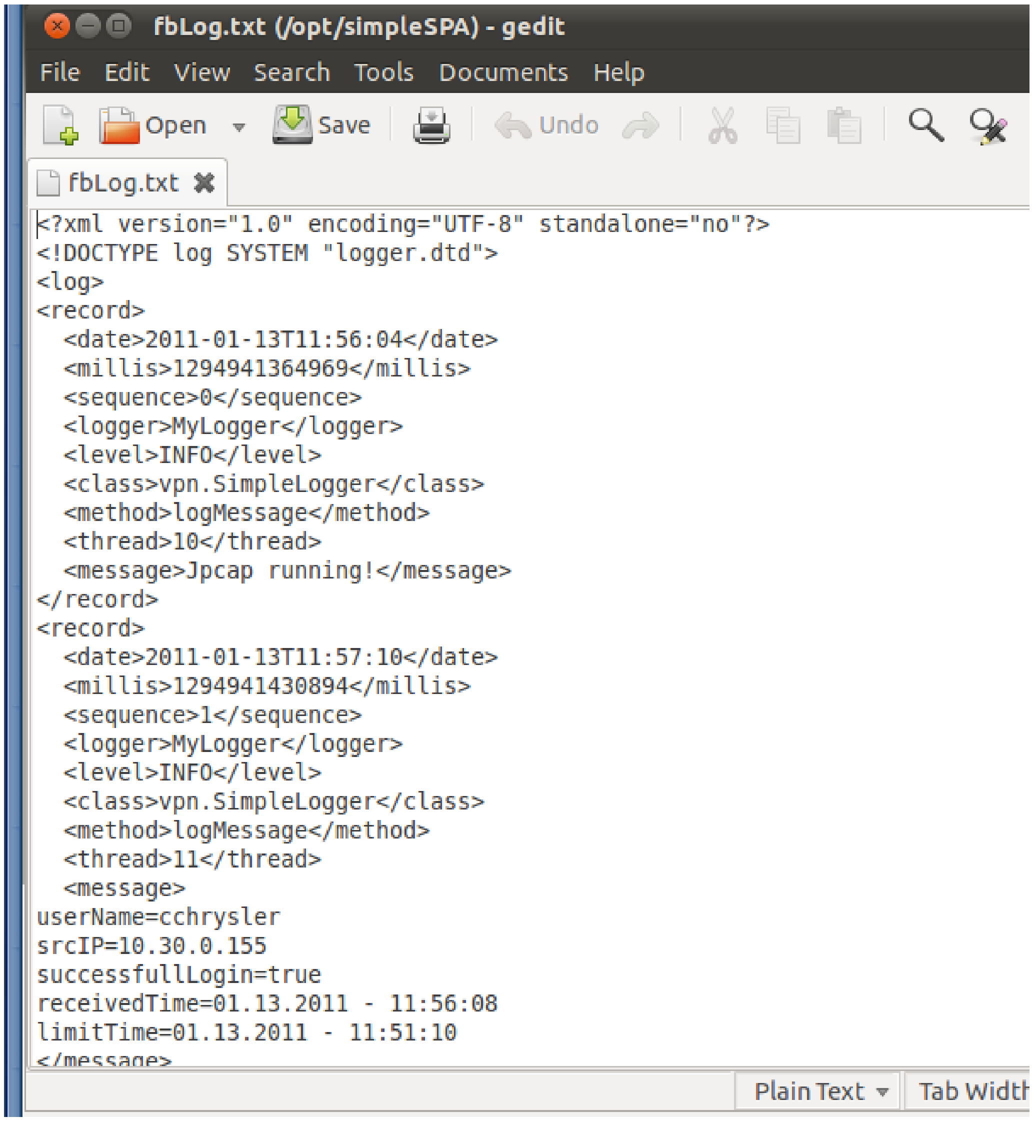
```
cchrysler@ubuntubox:~$ sudo java -cp /opt/simpleSPA/bcprov-jdk16-138.jar:/opt/simpleSPA/jpcap.jar:/opt/simpleSPA/VPNF
INAL.jar vpn.SimpleSPA /opt/simpleSPA/simpleSPA.conf
13-Jan-2011 11:56:04 AM vpn.SimpleLogger logMessage
INFO: Jpcap running!
13-Jan-2011 11:57:10 AM vpn.SimpleLogger logMessage
INFO:
userName=cchrysler
srcIP=10.30.0.155
successfulLogin=true
receivedTime=01.13.2011 - 11:56:08
limitTime=01.13.2011 - 11:51:10
```

```
hrysler\code\simpleSPA\masterPub c:\chrysler\code\simpleSPA\cchryslerPub cchrysl
er 2222 22222 10.30.0.221
outerText = cchrysler
sending: 57591ce35b64579dd13f94936578c97ab7223ea58f97f8f6ca0c4cafe313eaffb9f04ee
a99f25c110d2416fd121c85481c3605a1f5e681f4bca060641d80efcf7ee993fc765d6deca82b69b
327f478906ff79b6ba1efbbd0959b30aee49804f6109511511064f18ccb608aab31c8c4e5871e54b
3e00b695634a88c671d6e82230d15b0612eb6da6e0ef2c8cc1012c4e6919683cdc0c0b05b0d9b48b
c996a2e1286021fc77195cbb170a0c1cb8f64bde966971fefcc503bd6eaee5d6467177ce9feb7856
9040e9044703176339e9714013aeaf4569af75cb31831ed1ac9c8c5655aea4794b3f7ef53726b440
b8e92d9ae2e2c210c18f01d4748d3acc12b715d81

c:\>java -cp c:\chrysler\code\simpleSPA\bcprov-jdk16-138.jar;c:\chrysler\code\si
mpleSPA\SPASimpleCSCompanion.jar spa.UDPPacketAuthStandAlone_v2 10.30.0.155 c:\c
hrysler\code\simpleSPA\masterPub c:\chrysler\code\simpleSPA\cchryslerPub cchrysl
er 2222 22222 10.30.0.221
outerText = cchrysler
sending: 210b2a59a33a527da1d8209222c8adb7ce878d1d853e2c3802b11a1f1a5e20474fd2ad7
cb82180c6b950cad3f57663c62a27da65b19351236a44bba372f34dcbcefec3112d3841f64b629d0
72ae56d59b37dbc3875dd45d157d3754baf3c2a29366cf1ca00818157400a7d71ac991ffbe7cf5be
96bd467f86bba9eae41981dfcab77a27a06374665f92e2a6b25921641fd024091315e5f3d2857d6b
0e1e61b8361d94e84bddf3087fa3a0c8a99626a7fa69702ae7967fe432cbc45a417b621e96642af0
d4eeadf143e533232424e9fd20d7ef63fd2e07907338edd2fc31edbc608f6e654bca3c78279c1247
28d4011bcf34255a41ae1d8ce9a8325f0670e0a80

c:\>_
```

It is setup to log to a file in xml format.


fbLog.txt (/opt/simpleSPA) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
fbLog.txt
Plain Text
Tab Width

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2011-01-13T11:56:04</date>
  <millis>1294941364969</millis>
  <sequence>0</sequence>
  <logger>MyLogger</logger>
  <level>INFO</level>
  <class>vpn.SimpleLogger</class>
  <method>logMessage</method>
  <thread>10</thread>
  <message>Jpcap running!</message>
</record>
<record>
  <date>2011-01-13T11:57:10</date>
  <millis>1294941430894</millis>
  <sequence>1</sequence>
  <logger>MyLogger</logger>
  <level>INFO</level>
  <class>vpn.SimpleLogger</class>
  <method>logMessage</method>
  <thread>11</thread>
  <message>
userName=cchrysler
srcIP=10.30.0.155
successfullLogin=true
receivedTime=01.13.2011 - 11:56:08
limitTime=01.13.2011 - 11:51:10
</message>
```