三米前有蕉皮

人不折腾枉少年

Metasploit后渗透模块开发

作者:	三米前有蕉皮
邮箱:	root@kali-team.cn
哔哩哔哩:	https://www.bilibili.com/video/bv1Za4y147af
博客:	https://www.cnblogs.com/Kali-Team/
CC协议	 知识共享许可协议本作品采用知识共享署名-非商业性使用-相同方式共享 4.0 国际许可协议进行许可。

前言

- 接着上几个视频都是演示添加用户的，然后突然想起Metasploit里面的添加用户是调用命令行的，我就想能不能把它改为调用API的，给Metasploit提交了一点点代码，这些都是我用过之后得到的一些函数，所以不一定全都是对的，而且存在对某一个知识领域不是很了解，可能会有地方讲错，请大家指正。

结构

文件结构

external：扩展文件，比如zsh的命令行自动补全，里面有一个source文件夹

data：放一些exploits要用的二进制文件，字典，配置等等，一般是上传到目标主机上执行或者在本地的一些辅助文件，里面有一个meterpreter文件放的是留后门时用到的文件。

scripts：独立脚本，可以学习里面的套路，自动化脚本。

tools：开发辅助参考等等

plugins：和其他工具的联动接口，rpc等等

modules结构

auxiliary：可以理解打点的时候用的辅助模块，端口扫描，指纹识别，漏洞验证，登录密码爆破等等

encoders：编码混淆

exploits：漏洞利用，先按照操作系统分类，里面再是各种应用协议分类

payloads：一共有三个不同的payload：**Singles**，**Stagers**和**Stages**。

1. **Singles**是独立的payload，就是一个单独个功能，比如添加一个用户，执行一条命令，生成出来就不依赖Metasploit这个框架了，可以理解为shellcode；
2. **Stagers**是需要依赖到**Metasploit**框架和目标主机建立网络连接，但是依赖较少，功能也比较单一，比如弹回一个shell；
3. **Stages**就是我们常用的**Meterpreter**(Meta-Interpreter的缩写)这个高级payload，功能强大，DLL反射

post：后渗透模块

Post Exploitation

打印信息

```
class.instance_variables.map{|v|v.to_s[1..-1]}
class.methods.map &:to_s
https://rapid7.github.io/metasploit-framework/api/
https://www.rubydoc.info/github/rapid7/metasploit-framework
pry调试
```

最新随笔

1.Gather xshell and xftp passwords

2.Vue实现前端标签绑定搜索

3.axios拦截器

4.Metasploit Web Service

5.Meterpreter Window Enhanced

Kali-Team

哔哩哔哩

Github

KeyBase

友情链接

Akkuman'Blog

倾旋'Blog

gh0StKey'Blog

少年游~'Blog

Lz1y'Blog

阅读排行榜

1. Metasploit后渗透模块开发(462)

2. Write-up-CH4INRULZ_v1.0.1(423)

3. Write-up-Violator(210)

4. Genymotion设置代理至BurpSuite和Charles(199)

函数	描述
print_line	打印普通信息
print_good	向终端输出绿色信息，成功，好消息
print_error，print_bad	向终端输出红色信息，失败，坏消息
print_warning	向终端输出黄色信息，警告
print_status	向终端输出绿色信息，状态
print_blank_line	打印空行

```
print_line("---")
print_good("successful")
print_error("error")
print_warning("warning")
print_status("status")
print_blank_line
```

当前session信息

```
meterpreter > sysinfo
Computer        : WIN-A18RNMNL9C2
OS              : Windows 2008 R2 (6.1 Build 7601, Service Pack 1).
Architecture    : x64
System Language : zh_CN
Domain          : KALI-TEAM
Logged On Users : 2
Meterpreter     : x86/windows
```

函数	描述
session.platform	获取目标操作系统平台，返回windows或其他操作系统平台等等
session.type	获取session的类型。返回meterpreter或其他session类型等等
session.tunnel_to_s	隧道
session.arch	获取目标平台架构，x86或者x64，常量（ARCH_X64，ARCH_X86）
session.info	获取主机名和用户名
session.run_cmd	相当于在msf控制台敲命令
session.session_host	获取目标连接通信IP地址
session.session_port	获取目标连接通信端口
session.session_type	类型
session.payload_uuid	payload的UUID，在调用API的时候要用到
session.exploit_uuid	exploit的UUID，在调用API的时候要用到
session.uuid	UUID，在调用API的时候要用到
session.lookup_error(5)	Windows的错误常量
session.exploit_datastore	exploit选项

```
print_good(session.platform.to_s)
print_good(session.type.to_s)
print_good(session.tunnel_to_s.to_s)
print_good(session.arch.to_s)
print_good(session.info.to_s)
print_good(session.session_host.to_s)
print_good(session.session_port.to_s)
print_good(session.session_type.to_s)
print_good(session.lookup_error(5).to_s)
print_good(session.exploit_datastore['payload'].to_s)
```

目标网络信息

- **session.net.config.**

函数	描述

函数	描述
interfaces	获取网卡信息
each_interface	枚举网卡
arp_table	arp表对象
get_routes	获取路由信息
remove_route	移除路由
netstat	netstat
each_route	枚举路由
add_route	添加路由
routes	routes表
get_netstat	get_netstat
get_proxy_config	获取代理配置
get_arp_table	获取ARP表

- interfaces对象

```
[#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x0000562efff8bbd0 @index=10,
@mac_addr="\x00\f)Rr\xD0", @mac_name="Intel(R) PRO/1000 MT Network Connection", @mtu=1500, @flags=nil, @addrs=
["fe80::4c6f:11ed:581f:c274", "192.168.76.132"], @netmasks=["ffff:ffff:ffff:ffff::", "255.255.255.0"], @scopes=
["\n\x00\x00\x00"]>
```

- arp_table对象

```
[#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Arp:0x00007f3d38463030 @ip_addr="224.0.0.22",
@mac_addr="00:00:00:00:00:00", @interface="1">
```

- get_route对象

```
[#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Route:0x00007f3d385a0b28 @subnet="0.0.0.0",
@netmask="0.0.0.0", @gateway="192.168.76.2", @interface="10", @metric=266>
```

- netstat对象

```
[#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Netstat:0x0000562f00520168 @local_addr="::",
@remote_addr="::", @local_port=54538, @remote_port=0, @protocol="udp6", @state="", @uid=0, @inode=0,
@pid_name="1344/dns.exe", @local_addr_str=":::54538", @remote_addr_str=":::*">
```

```
print_good(session.net.config.interfaces[0].mac_name.to_s)
session.net.config.each_interface do |interface|
    print_good(interface.addrs.to_s)
end
print_good(session.net.config.arp_table[0].ip_addr.to_s)
print_good(session.net.config.get_routes[0].gateway.to_s)
print_good(session.net.config.netstat[0].pid_name.to_s)
print_good(session.net.config.get_proxy_config.to_s)
session.net.config.add_route(subnet, netmask, gateway)   # Add route
```

核心功能

```
https://github.com/rapid7/metasploit-framework/wiki/Meterpreter-Transport-Control
lib/rex/post/meterpreter/client_core.rb
```

模块名称	描述
session.core.use	加载扩展插件
session.core.migrate	迁移进程
session.core.load_library	加载DLL
session.core.machine_id	机器ID
session.core.get_loaded_extension_commands('stdapi')	获取已加载扩展命令
session.core.secure	secure
session.core.transport_sleep	传输休眠
session.core.transport_add	添加传输

模块名称	描述
session.core.transport_change	reverse_tcp, reverse_http, bind_tcp
session.core.set_transport_timeouts	设置传输超时
session.core.transport_remove	移除传输
session.core.transport_next	关闭当前传输，切换到下一个传输
session.core.transport_prev	关闭当前传输，切换到上一个传输
session.core.transport_list	列出传输
session.core.create_named_pipe_pivot	创建命名管道

```
session.core.use("extapi")
```

注册表模块

```
lib/msf/core/post/windows/registry.rb
Msf::Post::Windows::Registry
```

根键		
HKEY_CLASSES_ROOT	用于存储一些文档类型,类,类的关联属性	
HKEY_CURRENT_CONFIG	用户存储有关本地计算机系统的当前硬件配置文件信息	
HKEY_CURRENT_USER	用于存储当前用户配置项	
HKEY_PERFORMANCE_DATA	用于存储当前用户对计算机的配置项	
HKEY_LOCAL_MACHINE	用于存储当前用户物理状态	
HKEY_USERS	用于存储新用户的默认配置项	
HKEY_DYN_DATA	一个特别的根键	

键操作

模块名称	描述
registry_hive_lookup	通过缩写注册根键
registry_createkey	创建键
registry_deletekey	删除键
registry_enumkeys	枚举键

```
print_good("#{registry_createkey(hkey+'X')}")
print_good("#{registry_deletekey(hkey+'X')}")
print_good(registry_enumkeys('HKEY_CURRENT_USER\\Software').to_s)
```

值操作

模块名称	描述
registry_getvaldata	获取值数据
registry_deleteval	删除值
registry_enumvals	枚举值
registry_getvalinfo	获取值信息（key，val），还返回值都类型
registry_setvaldata	设置值数据

```
reg_data_types = 'REG_SZ|REG_MULTI_SZ|REG_DWORD_BIG_ENDIAN|REG_DWORD|REG_BINARY|'
'REG_DWORD_LITTLE_ENDIAN|REG_NONE|REG_EXPAND_SZ|REG_LINK|REG_FULL_RESOURCE_DESCRIPTOR'

print_good(registry_enumvals('HKEY_CURRENT_USER\\Software\\TeamViewer\\').to_s)
print_good(registry_getvalinfo('HKEY_CURRENT_USER\\Software\\TeamViewer','SelectedLanguage').to_s)
print_good(registry_setvaldata('HKEY_CURRENT_USER\\Software\\TeamViewer','SelectedLanguageX', 'KT','REG_SZ').to_s)
print_good(registry_getvaldata('HKEY_CURRENT_USER\\Software\\TeamViewer','SelectedLanguage').to_s)
print_good(registry_deleteval('HKEY_CURRENT_USER\\Software\\TeamViewer','SelectedLanguageX').to_s)
```

Teamviewer主窗口句柄

用户账号管理

```
lib/msf/core/post/windows/accounts.rb
Msf::Post::Windows::Accounts
```

模块名称	描述
get_domain	获取域名
delete_user	删除用户
resolve_sid	处理sid，**e.g.**('S-1-5-18')
check_dir_perms	检查目录权限
add_user	添加用户
add_localgroup	添加本地组
add_group	添加域组
add_members_localgroup	添加用户到本地组
add_members_group	添加用户到域组
get_members_from_group	获取域组里面的用户
get_members_from_localgroup	获取本地组里面的用户
enum_user	枚举用户
enum_localgroup	枚举本地组
enum_group	枚举域组
net_server_enum	枚举网络服务
net_session_enum	枚举网络会话

API和错误常量

```
lib/msf/core/post/windows/error.rb
Msf::Post::Windows::Error
lib/rex/post/meterpreter/extensions/stdapi/railgun/def/windows/api_constants.rb
```

```
print_good(session.railgun.const('ERROR_ACCESS_DENIED').to_s)
```

日志事件

```
lib/rex/post/meterpreter/extensions/stdapi/sys/event_log.rb
lib/msf/core/post/windows/eventlog.rb
scripts/meterpreter/event_manager.rb
include Msf::Post::Windows::Eventlog
```

模块名称	描述
eventlog_list	列出日志
eventlog_clear	清除日志

- **event_manager**插件

PowerShell模块

```
lib/msf/core/post/windows/powershell.rb
Msf::Post::Windows::Powershell
```

模块名称	描述
read_script	读入一个脚本
execute_script	执行脚本返回输出内容，文本
have_powershell?	判断是否存在powershell

模块名称	描述
get_powershell_version	获取powershell的版本
psh_exec	执行PowerShell文本

```
base_script = File.read(File.join(Msf::Config.data_directory, "post", "powershell", "NTDSgrab.ps1"))
execute_script(base_script)
```

系统

```
lib/msf/core/post/windows/priv.rb
lib/rex/post/meterpreter/extensions/stdapi/sys
lib/rex/post/meterpreter/extensions/stdapi/stdapi.rb
Msf::Post::Windows::Priv
```

- 以前是有提权模块的，但是现在全部归到local漏洞那边了。所以就剩下这些辅助函数了。

函数	描述
session.sys.config.sysinfo['OS']	获取sysinfo里面的值
session.sys.config.getprivs	权限标识
session.sys.config.getenv	获取环境变量
session.sys.config.is_system?	是不是系统权限
session.sys.config.steal_token	偷进程token
session.sys.config.getuid	获取用户名
session.sys.config.revert_to_self	返回自己的token
session.sys.config.getsid	获取sid标示
session.sys.config.getdrivers	枚举驱动，枚举类型
session.sys.config.drop_token	丢弃当前token
is_admin?	判断是不是admin
steal_current_user_token	偷当前用户的token
is_in_admin_group?	判断是不是在admin组
is_uac_enabled?	UAC是否开启
get_uac_level	获取UAC等级
session.priv.getsystem	getsystem

```
print_good(session.sys.config.sysinfo.to_s)
print_good(session.sys.config.getprivs.to_s)
print_good(session.sys.config.getenv("windir").to_s)
print_good(session.sys.config.getuid.to_s)
print_good(session.sys.config.getsid.to_s)
print_good(session.sys.config.getdrivers[0].to_s)
print_good(session.sys.config.is_system?.to_s)
print_good(is_admin?.to_s)
print_good(steal_current_user_token.to_s)
print_good(is_in_admin_group?.to_s)
print_good(is_uac_enabled?.to_s)
print_good(get_uac_level.to_s)
```

反射DLL

```
lib/msf/core/post/windows/reflective_dll_injection.rb
modules/post/windows/manage/shellcode_inject.rb
Msf::Post::Windows::ReflectiveDLLInjection
```

模块名称	描述
inject_into_process	注入shellcode到进程
inject_dll_into_process	注入dll到进程
inject_dll_data_into_process	注入反射性dll数据到进程

服务管理

```
lib/rex/post/meterpreter/extensions/extapi/service/service.rb
lib/msf/core/post/windows/services.rb
include Msf::Post::Windows::Services
```

模块名称	描述
each_service	枚举服务，枚举类型
service_list	列举服务
service_change_startup	修改启动方式
service_change_config	修改服务配置
service_create	创建服务
service_start	启动服务
service_stop	停止服务
service_delete	删除服务
service_status	服务状态
service_restart	重启服务
service_info	获取服务信息

```
each_service do |service|
    # print_good("#{service}")
    if service[:display] == 'TeamViewer'
        print_good(service_info(service[:name]).to_s)
        print_good(service_status(service[:name]).to_s)
    end
end
# ["Boot","System","Auto","Manual","Disabled"]
service_change_startup('TeamViewer', START_TYPE_DISABLED)
service_create("TeamViewerX", { path: 'C:\\Program Files (x86)\\TeamViewer\\TeamViewer_Service.exe', display:
"TEXT" })
service_start("TeamViewerX")
service_stop("TeamViewerX")
service_delete("TeamViewerX")
```

```
modules/exploits/windows/local/service_permissions.rb
```

用户设置

```
lib/msf/core/post/windows/user_profiles.rb
Msf::Post::Windows::UserProfiles
```

模块名称	描述
grab_user_profiles	获取用户配置

进程模块

```
lib/msf/core/post/windows/process.rb
Msf::Post::Windows::Process
```

函数	描述
session.sys.process.getpid	获取当前进程pid
session.sys.process.open(pid.to_i, PROCESS_ALL_ACCESS)	打开一个进程，返回一个进程句柄
session.sys.process.processes	枚举所有进程信息
session.sys.process.execute	执行程序
session.sys.process.kill	杀掉一个进程
session.sys.process.each_process	枚举类型
session.sys.process.get_processes.keep_if	枚举类型
execute_shellcode	执行shellcode

函数	描述
inject_unhook	注入释放钩子
has_pid	pid是否存在

执行shellcode

1. 获取当前的pid：session.sys.process.getpid
2. 打开进程得到进程句柄：host = session.sys.process.open(pid.to_i, PROCESS_ALL_ACCESS)
3. 申请内存：shell_addr = host.memory.allocate(shellcode.length)
4. 保护当前地址：host.memory.protect(shell_addr)
5. 向地址写入shellcode：host.memory.write(shell_addr, shellcode)
6. 执行shellcode：host.thread.create(shell_addr,0)

文件系统操作

```
include Msf::Post::File
lib/rex/post/dir.rb
lib/rex/post/file_stat.rb
lib/rex/post/file.rb
lib/rex/post/meterpreter/extensions/stdapi/fs/file_stat.rb
```

文件操作

- **session.fs.file.**

```
lib/rex/post/meterpreter/extensions/stdapi/fs/file.rb
```

函数	描述
separator	获取系统目录路径的分隔符\
expand_path	解析环境变量形式的文件路径'%appdata%'
rm，delete	删除文件
new	新建文件返回一个句柄
rename，mv	重命名文件
download	下载远程文件到本地
stat	文件信息
move，mv	移动文件
search	搜索文件
chmod	修改文件属性
exist	文件是否存在
open	打开文件返回一个句柄
download_file	下载远程单文件到本地
copy，cp	拷贝文件
file_local_write	写文件到本地
upload_file	上传单文件到远程
write_file	写文件到远程
sha1	获取文件的sha1
md5	获取文件的md5

文件夹操作

- **session.fs.dir**

```
lib/rex/post/meterpreter/extensions/stdapi/fs/dir.rb
```

函数	描述
entries，ls	列出当前文件夹里的文件

函数	描述
entries_with_info	列出当前文件夹里的文件，带文件的详细信息
mkdir	新建目录
match	匹配文件
foreach	枚举文件夹
chdir	切换到目录，就是cd
pwd，getwd	显示当前目录路径
delete，rmdir，unlink	删除文件夹
download	递归下载远程文件夹到本地
upload	递归上传本地文件夹到远程

- 获取文件的详细信息

```
session.fs.file.stat
```

剪切板管理

```
lib/rex/post/meterpreter/extensions/extapi/clipboard/clipboard.rb
lib/rex/post/meterpreter/ui/console/command_dispatcher/extapi/clipboard.rb
include Msf::Post::Windows::ExtAPI
```

模块名称	描述
session.extapi.clipboard.set_text	设置剪切板文本
session.extapi.clipboard.get_data	获取剪切板数据 `-d` 下载非文本数据
monitor_start	开始监控
monitor_pause	暂停监控
monitor_dump	导出监控内容
monitor_resume	重新监控
monitor_purge	清除监控
monitor_stop	停止监控

WMIC

include Msf::Post::Windows::WMIC

模块名称	描述
wmic_query	查询wmic
wmic_command	执行wmic命令
wmic_user_pass_string	smbexec

Runas

Msf::Post::Windows::Runas

模块名称	描述
shell_execute_exe	执行exe
shell_execute_psh	执行PowerShell
shell_exec	执行命令
create_process_with_logon	以登录用户创建进程
create_process_as_user	以指定用户创建进程

Kiwi

Msf::Post::Windows::Kiwi

模块名称	描述
password_change	修改密码
dcsync	同步域控
dcsync_ntlm	同步域控NTLM
lsa_dump_secrets	导出secrets
lsa_dump_sam	导出sam
lsa_dump_cache	导出cache
creds_all	获取全部凭证
kerberos_ticket_list	列出kerberos票据
kerberos_ticket_use	使用kerberos票据
kerberos_ticket_purge	清除kerberos票据
golden_ticket_create	创建黄金票据
wifi_list	列出WiFi凭证

ShadowCopy

Msf::Post::Windows::ShadowCopy

模块名称	描述
vss_list	列出卷影备份
vss_get_ids	获取卷影备份的id
vss_get_storage	获取卷影备份储存的参数
get_sc_details	列出指定id卷影备份的详细信息
get_sc_param	获取制定id卷影备份的参数信息
vss_get_storage_param	获取卷影备份储存的指定参数
vss_set_storage	设置卷影备份储存
create_shadowcopy	创建卷影备份
start_vss	启动卷影备份

LDAP

Msf::Post::Windows::LDAP

- 这个我不会！下次一定！

RailGun

```
lib/msf/core/post/windows/railgun.rb
lib/rex/post/meterpreter/extensions/stdapi/railgun/library.rb
lib/rex/post/meterpreter/extensions/stdapi/railgun/library_function.rb
```

模块名称	描述
known_library_names	列出可用DLL
memread	读内存
memwrite	写内存
add_function	添加函数

模块名称	描述
add_library，add_dll	添加库
get_library，get_dll	获取库，判断存不存在
multi	执行多个函数，数组形式
const	获取常量
lookup_error	lookup_error
pointer_size	获取指针大小，x86与x64的差别

数据类型

```
  @@allowed_datatypes = {
    'VOID'   => ['return'],
    'BOOL'   => ['in', 'return'],
    'DWORD'  => ['in', 'return'],
    'WORD'   => ['in', 'return'],
    'BYTE'   => ['in', 'return'],
    'LPVOID' => ['in', 'return'], # sf: for specifying a memory address (e.g. VirtualAlloc/HeapAlloc/...) where we
don't want to back it up with actual mem ala PBLOB
    'HANDLE' => ['in', 'return'],
    'SIZE_T' => ['in', 'return'],
    'PDWORD' => ['in', 'out', 'inout'], # todo: support for functions that return pointers to strings
    'PWCHAR' => ['in', 'out', 'inout'],
    'PCHAR'  => ['in', 'out', 'inout'],
    'PBLOB'  => ['in', 'out', 'inout'],
  }.freeze

  @@allowed_convs = ['stdcall', 'cdecl']

  @@directions = ['in', 'out', 'inout', 'return'].freeze
```

- 在下面文件有详细的转换对应关系

```
lib/rex/post/meterpreter/extensions/stdapi/railgun/util.rb
VOID, BOOL, DWORD, WORD, BYTE, LPVOID, HANDLE, PDWORD, PWCHAR, PCHAR, PBLOB
```

- 如果是指针数据类型的要使用：PBLOB类型，返回来的使用unpack解析，pack之后作为参数传进。

C语言	Railgun	描述
LPCWSTR，LPWSTR	PWCHAR	
DWORD	DWORD	
LPCSTR	PCHAR	
*LPVOID	PBLOB	指针各种奇怪的数据类型
*DWORD，LPDWORD	PDWORD	一般是一个指针地址DWORD，返回值存储变量的地址
VOID	VOID	VOID返回类型
BOOL	BOOL	
WORD	WORD	一般是常量，bits
BYTE	BYTE	
PSID	LPVOID	指针内存
HANDLE	HANDLE	句柄

```
#process return value
case function.return_type
  when 'LPVOID', 'HANDLE'
    if( @native == 'Q<' )
      return_hash['return'] = rec_return_value
    else
      return_hash['return'] = rec_return_value % 4294967296
    end
  when 'DWORD'
    return_hash['return'] = rec_return_value % 4294967296
  when 'WORD'
    return_hash['return'] = rec_return_value % 65536
  when 'BYTE'
    return_hash['return'] = rec_return_value % 256
  when 'BOOL'
    return_hash['return'] = (rec_return_value != 0)
  when 'VOID'
```

```
    return_hash['return'] = nil
  else
    raise "unexpected return type: #{function.return_type}"
end
```

- 定义函数

```
lib/rex/post/meterpreter/extensions/stdapi/railgun/railgun.rb
```

- 添加函数

```
 Example:
   add_function("MessageBoxW",   # name
     "DWORD",                    # return value
      [                           # params
          ["DWORD","hWnd","in"],
      ["PWCHAR","lpText","in"],
      ["PWCHAR","lpCaption","in"],
      ["DWORD","uType","in"],
  ])
```

- 添加DLL库

模块名称	用途	举例
session.railgun.netapi32	用户相关	添加用户等等
session.railgun.util	工具函数	各种实用函数
known_library_names	已知可用DLL	

```
https://docs.microsoft.com/en-us/previous-versions//aa383749(v=vs.85)?redirectedfrom=MSDN
```

添加用户例子

- C代码

> https://docs.microsoft.com/en-us/windows/win32/api/lmaccess/nf-lmaccess-netuseradd

```
NET_API_STATUS NET_API_FUNCTION NetUserAdd(
  LPCWSTR servername,
  DWORD   level,
  LPBYTE  buf,
  LPDWORD parm_err
);
```

- 在文件 `lib/rex/post/meterpreter/extensions/stdapi/railgun/def/windows/def_netapi32.rb` 添加函数。

```
dll.add_function('NetUserAdd', 'DWORD', [
    ["PWCHAR","servername","in"],
    ["DWORD","level","in"],
    ["PBLOB","buf","in"],
    ["PDWORD","parm_err","out"]
    ])
```

- 你会发现 `level` 这个参数他是一个结构体，和上面介绍的几种数据类型都对不上，怎么把 `level` 传给 `NetUserAdd` 呢？Ruby里有一个pack可以封装结构体，pack就是告诉railgun在内存中怎么解析这串东西。

> https://docs.microsoft.com/en-us/windows/win32/api/Lmaccess/ns-lmaccess-user_info_1

- 结构体

```
typedef struct _USER_INFO_1 {
  LPWSTR usri1_name;
  LPWSTR usri1_password;
  DWORD  usri1_password_age;
  DWORD  usri1_priv;
  LPWSTR usri1_home_dir;
  LPWSTR usri1_comment;
  DWORD  usri1_flags;
  LPWSTR usri1_script_path;
} USER_INFO_1, *PUSER_INFO_1, *LPUSER_INFO_1;
```

- 上面有5个LPWSTR类型的变量，因为它是一个指针类型，在x86架构里的指针寻址32位，用pack封装的时候全部使用V就可以了，V在ruby的pack中表示：小端字节顺序的unsigned long (32bit 无符号整数)；在x64架构里寻址位数就不一样了，所以这个结构体在内存就会不一样，所以封装的时候就要使用Q，Q在ruby的pack中表示：小端字节顺序unsigned long long(64bit 无符号整数)。这个问题我用x64dbg调了一天[捂脸]。

```
def add_user(username, password, server_name = nil)
    addr_username = session.railgun.util.alloc_and_write_wstring(username)
    addr_password = session.railgun.util.alloc_and_write_wstring(password)
    #  Set up the USER_INFO_1 structure.
    #  https://docs.microsoft.com/en-us/windows/win32/api/Lmaccess/ns-lmaccess-user_info_1
    user_info = [
        addr_username,
        addr_password,
```

```
        0x0,
        0x1,
        0x0,
        0x0,
        client.railgun.const('UF_SCRIPT | UF_NORMAL_ACCOUNT|UF_DONT_EXPIRE_PASSWD'),
        0x0
        ].pack(client.arch == "x86" ? "VVVVVVVV" : "QQVVQQVQ")
    result = client.railgun.netapi32.NetUserAdd(server_name, 1, user_info, 4)
    client.railgun.multi([
        ["kernel32", "VirtualFree", [addr_username, 0, MEM_RELEASE]], #  addr_username
        ["kernel32", "VirtualFree", [addr_password, 0, MEM_RELEASE]], #  addr_password
        ])
    return result
end
```

解析返回数据

- 上面的传参一个解决了，解析数据可以使用unpack，或者有别人在util写好的函数

枚举用户

- C代码

> https://docs.microsoft.com/en-us/windows/win32/api/lmaccess/nf-lmaccess-netuserenum

```
NET_API_STATUS NET_API_FUNCTION NetUserEnum(
  LPCWSTR servername,
  DWORD   level,
  DWORD   filter,
  LPBYTE  *bufptr,
  DWORD   prefmaxlen,
  LPDWORD entriesread,
  LPDWORD totalentries,
  PDWORD  resume_handle
);
```

- 添加函数

```
dll.add_function('NetUserEnum', 'DWORD', [
    ["PWCHAR","servername","in"],
    ["DWORD","level","in"],
    ["DWORD","filter","in"],
    ["PBLOB","bufptr","out"],
    ["DWORD","prefmaxlen","in"],
    ["PDWORD","entriesread","out"],
    ["PDWORD","totalentries","out"],
    ["PDWORD","ResumeHandle","inout"],
    ])
```

```
def enum_user(server_name = nil)
    users = []
    filter = 'FILTER_NORMAL_ACCOUNT|FILTER_TEMP_DUPLICATE_ACCOUNT'
    result = client.railgun.netapi32.NetUserEnum(server_name, 0, client.railgun.const(filter), 4, 4096, 4, 4, 0)
    if (result['return'] == 0) && ((result['totalentries'] % 4294967296) != 0)
        begin
            user_info_addr = result['bufptr'].unpack1("V")
            unless user_info_addr == 0
                user_info = session.railgun.util.read_array(USER_INFO, (result['totalentries'] % 4294967296),
user_info_addr)
                for member in user_info
                    users << member["usri0_name"]
                end
                return users
            end
        end
    else
        return users
    end
    ensure
    session.railgun.netapi32.NetApiBufferFree(user_info_addr)
end
```

- 可以看一下我提交的PR，上面Railgun的用法我在这个推送请求都用上了

分类： Metasploit

三米前有蕉皮
关注 - 1
粉丝 - 0
+加关注

« 上一篇： Get TeamViewer ID and Password
» 下一篇： Caesar cipher

posted @ 2020-03-28 21:46 三米前有蕉皮 阅读(462) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。