SPEARTIP®

CYBER COUNTERINTELLIGENCE

Outmaneuver Your Adversary™

# ADVANCED MALWARE ANALYSIS METHODOLOGIES

## Dynamic Analysis - Memory Analysis – Trace Analysis

Malware has evolved over the decades to become what is essentially the malicious boots on the ground in a 21st century cyberwar between Cyber-Soldiers.

Kris Bleich, CISSP, EnCE, C|EH

# Malware Analysis Methodology

Over the years, the word “virus” has become broad and tends to introduce confusion into the marketplace when discussing what can commonly be referred to as “malware”. Malware typically includes adware, spyware, keyloggers, or even fake antivirus software. While malware has existed in one form or another since the 1970s, it has evolved over the decades to become what is essentially the malicious boots on the ground in a twenty-first century cyberwar between Cybersoldiers. Malware is the new malevolent Cybersoldier.

Tanks, airplanes, ships and satellites comprise just some of the platforms of modern warfare. These platforms can easily cost in the billions of dollars and in some cases, require a significant amount of resources and manpower to maintain and transport to a warzone. Many less developed countries are not able to field significant numbers of these advanced weapons platforms. Malware offers a much more advanced and flexible weapon that can be deployed in a short amount of time and without drawing obvious attention to an attacking government.

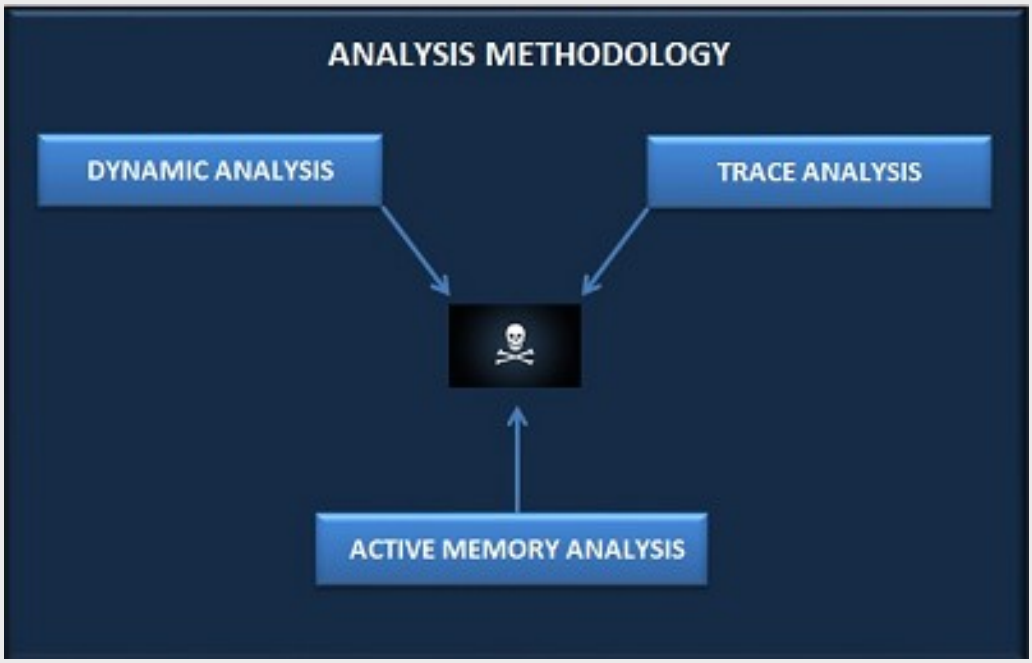
Malware often infects a computer system in multiple stages. In addition, there are many categories of malware which describe behavior, mission and functionality. While some specific malware samples may have limited capabilities and purposes, other more advanced malware families serve multiple purposes and may belong to more than one of these categories. In many instances, new malware strains are created building on older pieces of malware, utilizing code from known malware samples or code posted in the underground. This constant mutation and recompiling of new malware, along with the introduction and more frequent use of metamorphic engines and custom packers make malware detection and reverse engineering a much more challenging endeavor.

The process of Active Memory analysis allows for a detailed analysis of a malware sample as it operates within active memory, despite the use of packers, encryption or other obfuscation techniques. In addition, the analysis of active memory activity on a compromised system allows for the quick identification and analysis of malware without obtaining a “malicious file” for static analysis, which may be encrypted, obfuscated or not available. This analysis process can also detect advanced malware that may evade or compromise security software.

A “Trace Analysis”, while similar to traditional static analysis, is a dynamic analysis methodology that monitors and graphs program behavior, including new threads and processes created as a result of malware execution. This data can be analyzed in a historical context and allows for the quicker identification of what triggers observed behaviors and rapid visualization of malware execution. The Trace Analysis allows for a detailed analysis of a malware sample, despite the use of custom packers, encryption or other obfuscation techniques.

Dynamic Analysis of a malware sample’s execution and communication capability, coupled with an analysis of the hard drive of an infected system does allow for a more detailed “historical” analysis of the malware, possibly including a determination of infection vector, initial date of infection, and whether sensitive data was compromised during an incident. This process includes the monitoring of a malware sample within a virtual environment or other isolated malware analysis environment as well as the monitoring and analysis of network activity, both within an isolated analysis environment as well as within a compromised network environment.

These methodologies, used in conjunction with each other, provide a complete "big picture" of a particular malware incident, from specific malware operation and execution to communication and mission. In addition, this multi-layered analysis methodology allows for the quick analysis of malware samples as they operate within active memory, as well as how they interact with the file system on a compromised host.




The following documents the analysis of a known variant of the Tasmer trojan, SRVCP, focusing on its encrypted component. This malware utilizes an encrypted .ini file (gus.ini) to facilitate command and control communications via an IRC channel controlled by the attacker. The following analysis details the execution of this malware sample, including the decryption and implementation of the encrypted components, utilizing dynamic analysis, supported by a trace analysis of the malware sample and analysis of the active memory and hard drive of an infected system.

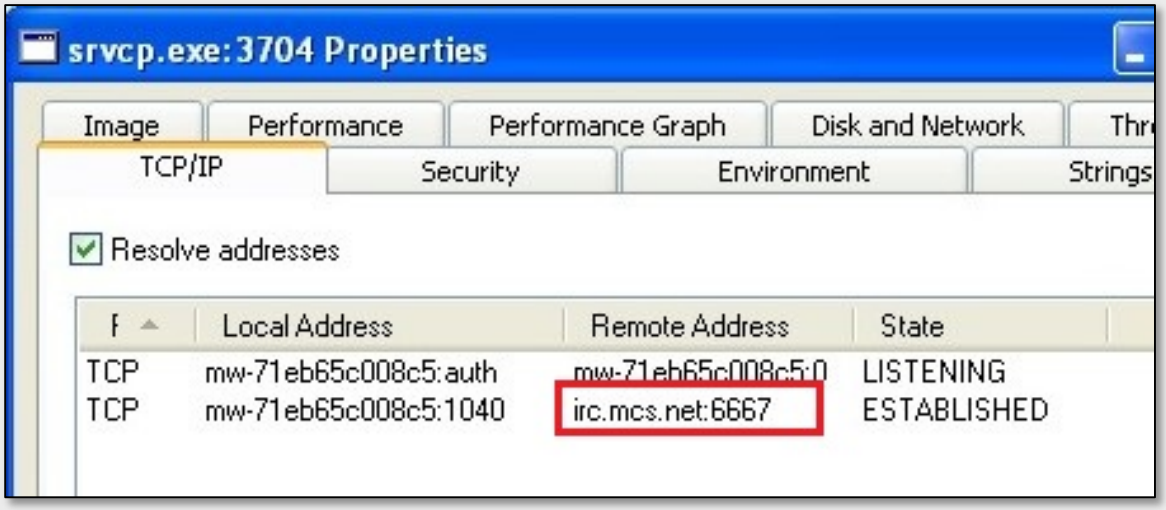
Below is a screenshot of the malware sample and the encrypted gus.ini file utilized by the malware. The gus.ini file is created in C:\Windows\system32.

| | |
|---|---|
| srvcp.exe | e9fe9148a69a1b8f70996435787609c3 |
| gus.ini | 112c42a181b298e6fd7a5b6733c44e47 |

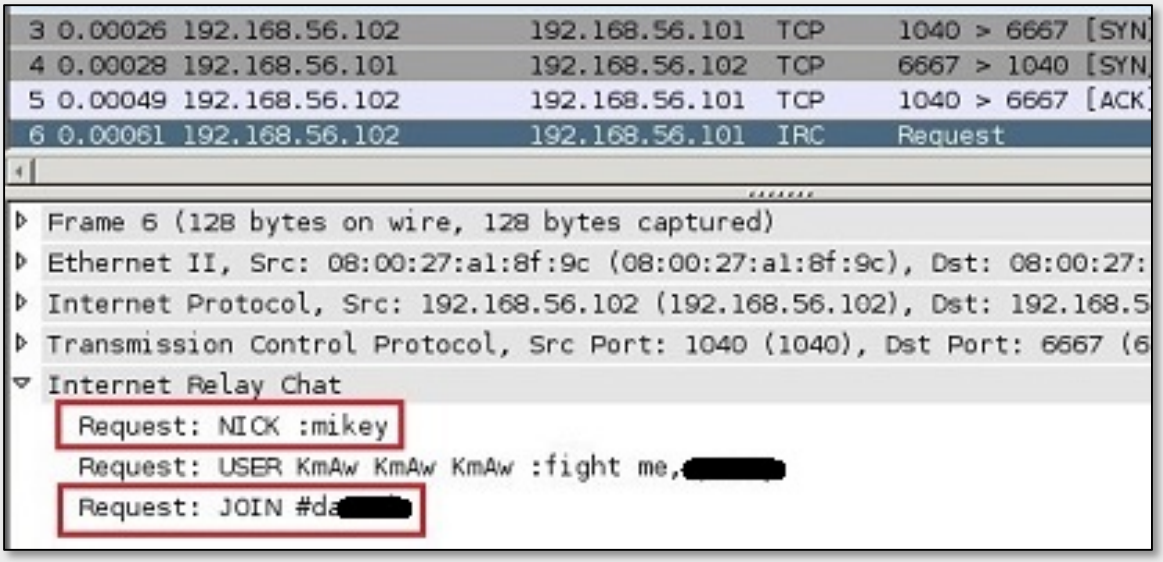
## Dynamic Analysis

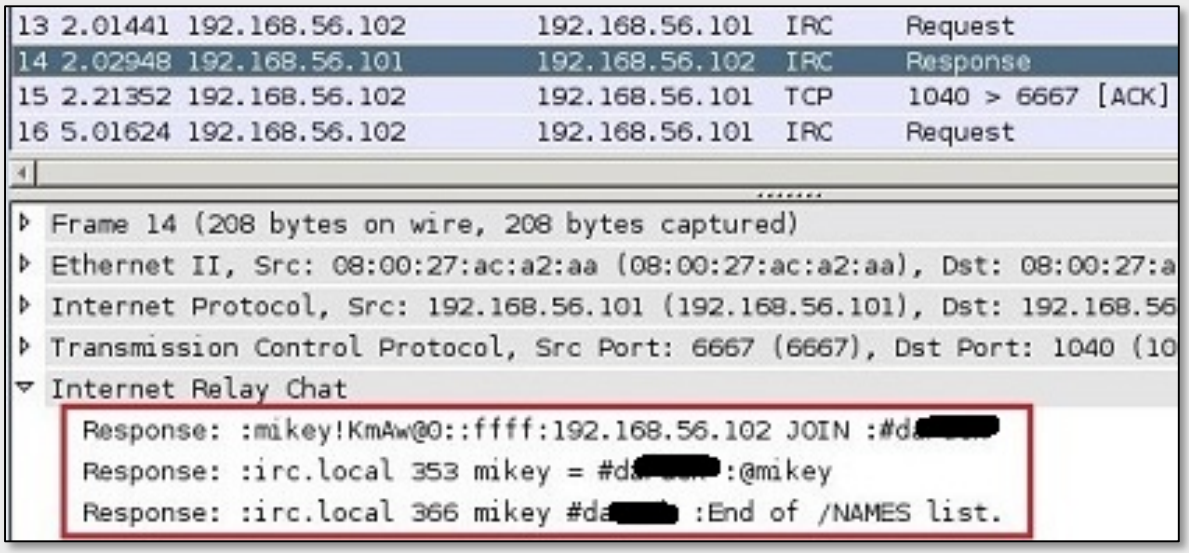
A Dynamic Analysis of this malware sample was conducted utilizing VMWare and an enclosed virtual network consisting of a Windows XP VM and a companion Linux VM. IRC services were configured on the Linux VM to provide the malware sample with IRC services while it was executed on the Windows XP VM. The hosts file on the Windows XP VM was edited to direct the communications from the malware sample to the Linux VM. In addition, an instance of Wireshark was utilized on the Linux VM to capture packet information from the infected Windows XP VM.

The malware sample was first executed without providing the encrypted gus.ini file to the malware to examine any built in functionality that the malware sample may have, independent of the encrypted .ini file. The analysis of the connection information for the malicious srvcp.exe process on the Windows XP VM found an established TCP connection on port 6667 to irc.mcs.net. Internet Relay Chat (IRC) typically uses this port for communications.



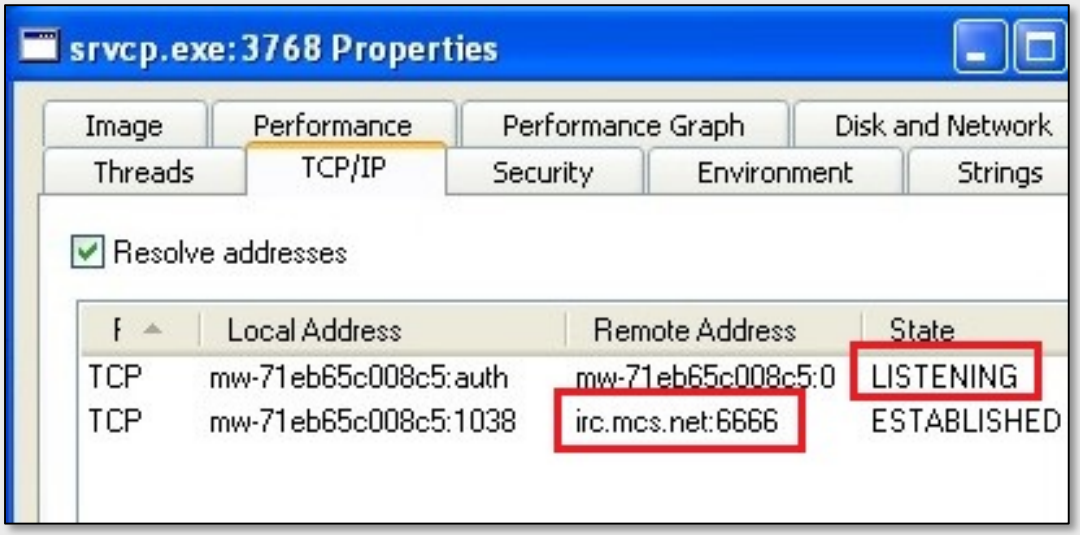

Analysis of the packet capture data found that after the malware establishes a connection, it attempts to join an IRC channel named “da////” using the nick “mikey”.



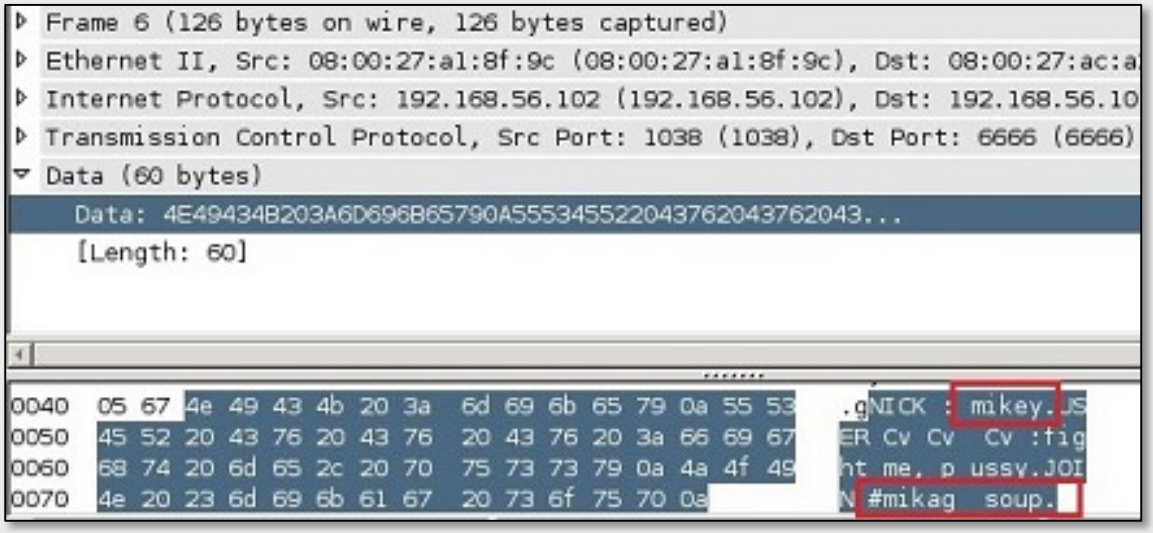



The IRC application on the Linux VM shows the user “mikey” joining the IRC channel “daXXXX”.

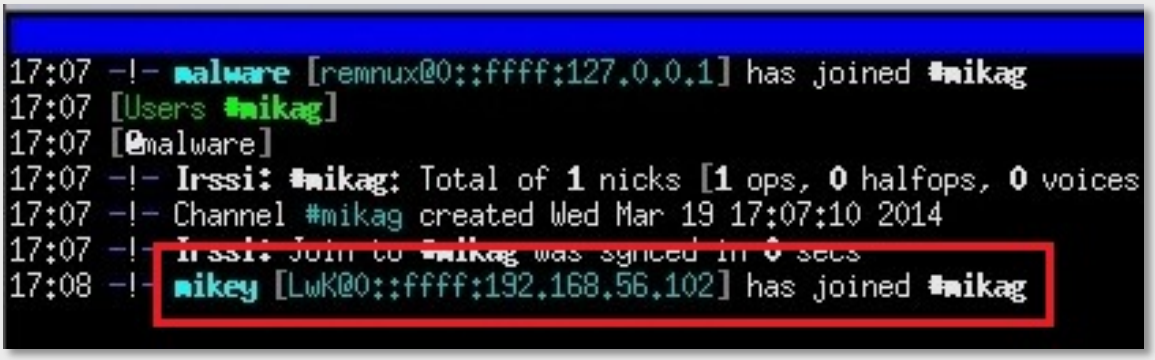

The malware sample was then executed after providing the encrypted gus.ini file to the malware by copying it to the file path location C:\Windows\system32. The analysis of the connection information for the malicious srvcp.exe process on the Windows XP VM found an established TCP connection on port 6666 to irc.mcs.net. IRC typically uses this port for communications.



Analysis of the packet capture data found that after the malware establishes a connection, it attempts to join an IRC channel named “mikag”, with a key of “soup”, using the nick “mikey”.




The IRC application on the Linux VM shows the user “mikey” joining the IRC channel “mikag”.



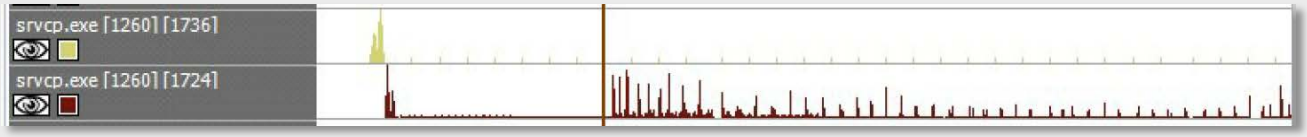

If the malware is unable to use the nick “mikey” because it has already been used, the malware appears to construct a random nick.


The malicious process then beacons to the command and control server every three seconds, attempting to change its nick to “mikey”.

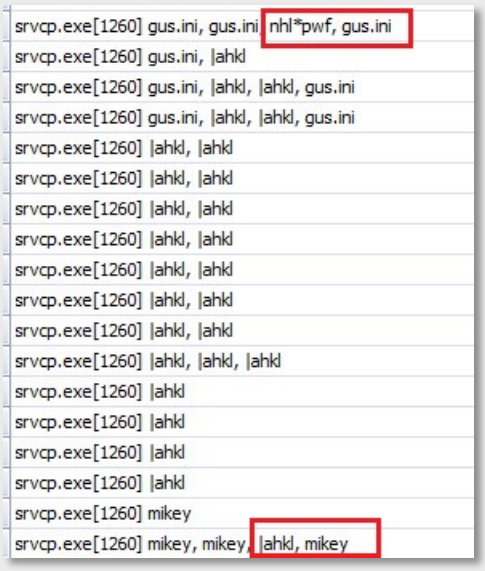
| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 18 | 7.99842 | 192.168.56.102 | 192.168.56.101 | TCP | 1038 > 6666 |
| 19 | 7.99846 | 192.168.56.101 | 192.168.56.102 | TCP | 6666 > 1038 |
| 20 | 11.0015 | 192.168.56.102 | 192.168.56.101 | TCP | 1038 > 6666 |
| 21 | 11.0015 | 192.168.56.101 | 192.168.56.102 | TCP | 6666 > 1038 |
| 22 | 14.0057 | 192.168.56.102 | 192.168.56.101 | TCP | 1038 > 6666 |
| 23 | 14.0058 | 192.168.56.101 | 192.168.56.102 | TCP | 6666 > 1038 |
| 24 | 17.0096 | 192.168.56.102 | 192.168.56.101 | TCP | 1038 > 6666 |
| 25 | 17.0097 | 192.168.56.101 | 192.168.56.102 | TCP | 6666 > 1038 |
| 26 | 20.0132 | 192.168.56.102 | 192.168.56.101 | TCP | 1038 > 6666 |

## Trace Analysis

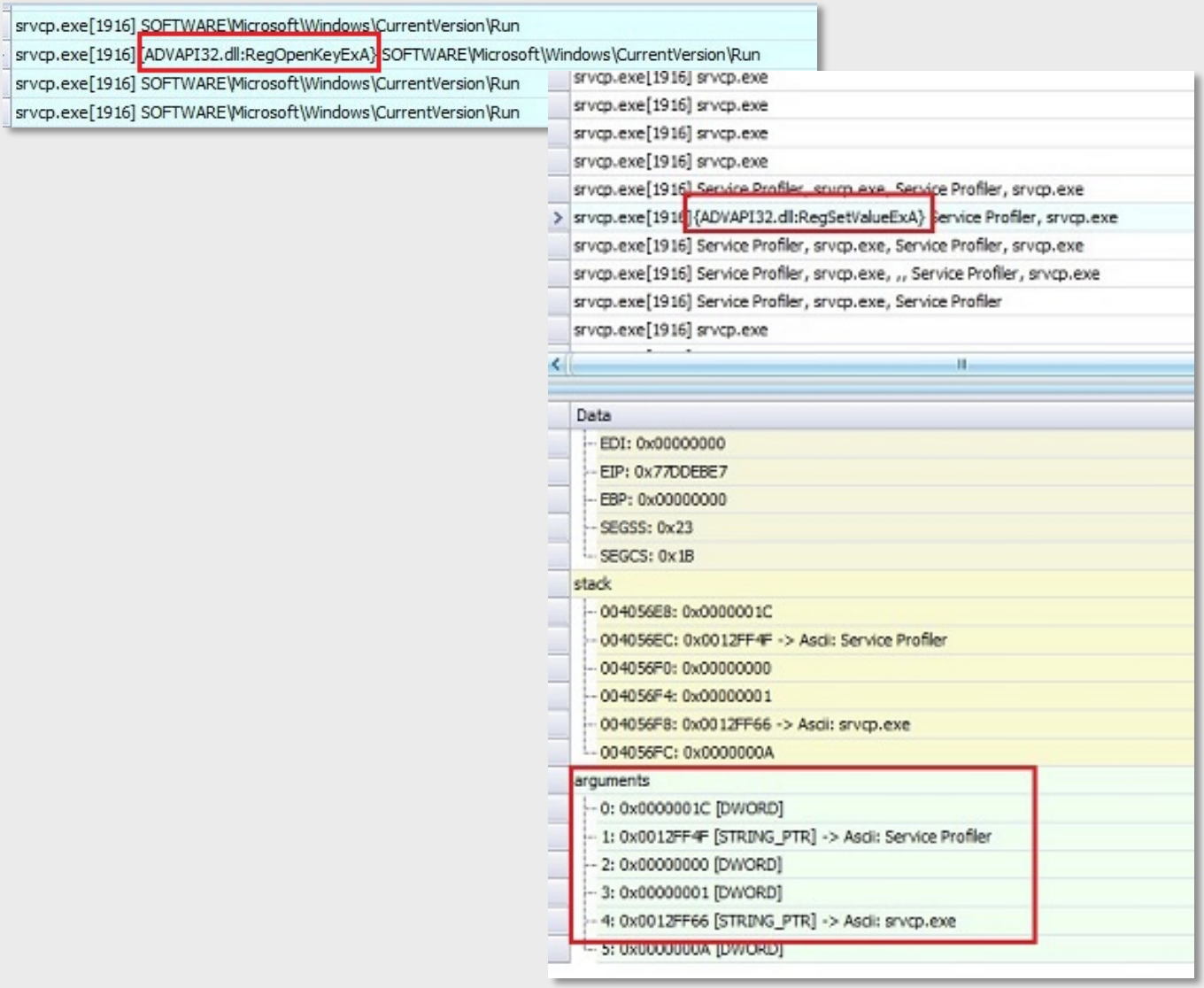
A "Trace Analysis" of the malware sample involves the execution of the sample within a VM. In this case, a Windows XP VM was utilized and the execution of the malware was traced for approximately 20 minutes within the VM. Two separate traces were obtained. The first trace was of the execution of the malware sample without providing the gus.ini file and the second trace was of the execution of the malware sample with the gus.ini file present. During the second trace, the malware was also provided an IRC channel on the companion VM to allow for the monitoring of the malware's behavior during and after its connection to the command and control IRC channel. An image of active memory and of the hard drive of the infected VM was also obtained for analysis after each trace. Below is a high-level view of the activity of the malware sample analyzed.




Using information obtained during the dynamic analysis, the first trace was analyzed. The initial analysis of the sample located what appeared to be encrypted strings being loaded into memory.



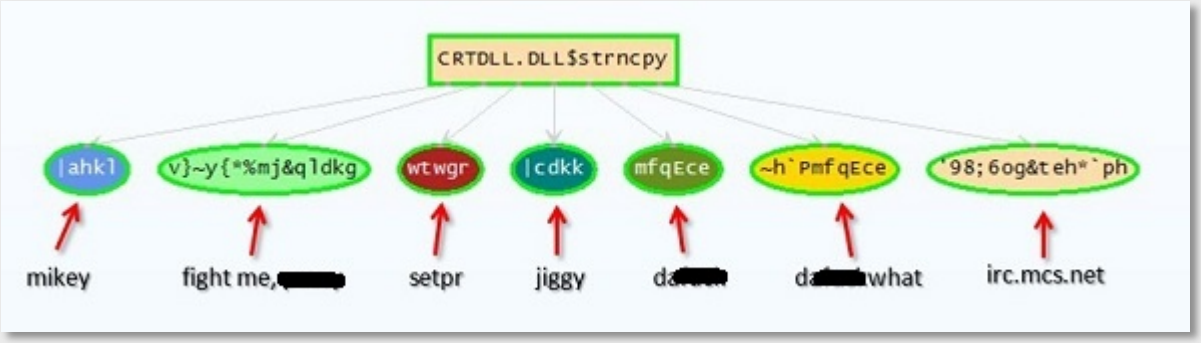
The analysis located the access and addition of an entry in the registry location \SOFTWARE\ Microsoft\Windows\ CurrentVersion\Run.



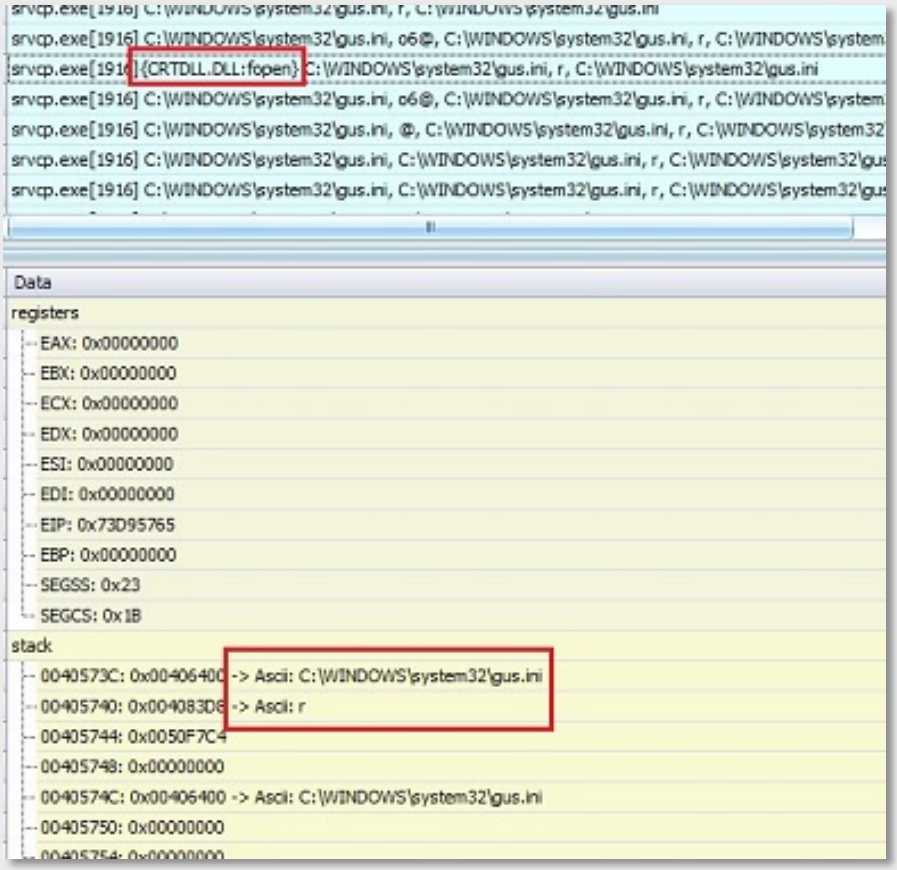

Modification of this registry entry allows the malware to persist on the compromised system following a reboot.

The Trace Analysis located encrypted IRC information hard-coded into the malware sample. Stepping through the trace allows for the identification of the decrypted values for each encrypted string.

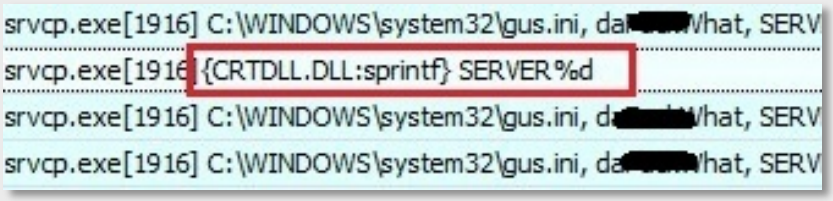
srvcp.exe[1680]{CRTDLL.DLL:strncpy} setpr
srvcp.exe[1680] C:\WINDOWS\system32\gus.ini, setpr
srvcp.exe[1680] C:\WINDOWS\system32\gus.ini, setpr
srvcp.exe[1680]{CRTDLL.DLL:strncpy} jiggy
srvcp.exe[1680] C:\WINDOWS\system32\gus.ini, T5@, jiggy
srvcp.exe[1680] C:\WINDOWS\system32\gus.ini, jiggy
srvcp.exe[1680]{CRTDLL.DLL:strncpy} da[redacted]



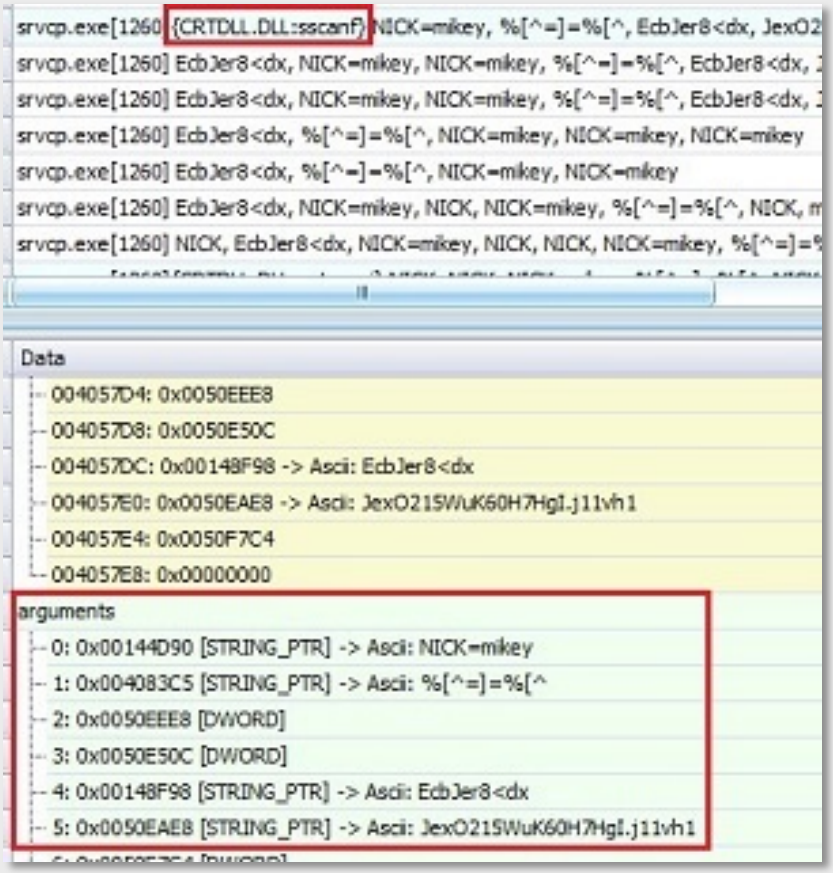

The analysis of the second trace found the malware sample accessing the gus.ini file for the purpose of reading the contents of the file.



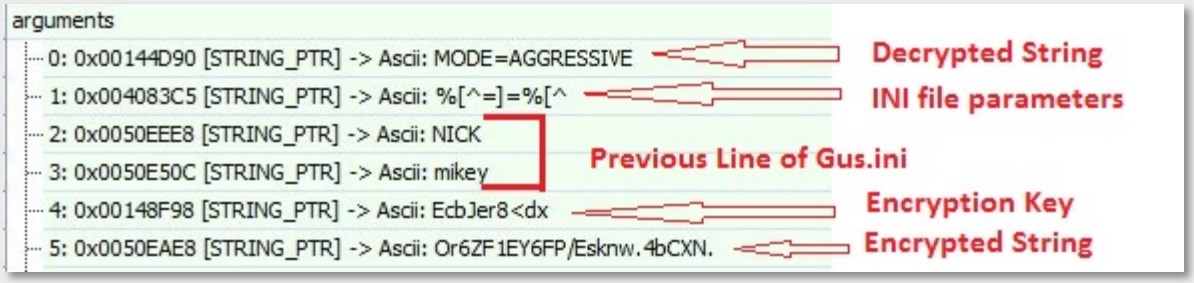
The malware uses a format string to identify additional servers, incrementing the server number for each successive server URL.




Stepping through the trace locates the decryption of the gus.ini file utilizing the sscanf function call. The sscanf function reads data from one location and stores it in another, based on format parameters provided. The screenshot below shows the decryption of the first line of the gus.ini file (NICK=mikey).



The next call to this function decrypts the second line, using the encryption key “EcbJer8<dx”.




The malware sample continues this decryption routine, labeling each successive server with an incremented server number (SERVER0, SERVER1, etc...). The result is the decryption of the encrypted gus.ini file and the loading of the data within it into active memory for use by the malware.

ENCRYPTED GUS.IMI

```
JexO215WuK6OH7HgI.j11vh1
Or6ZF1EY6FP/Esknw.4bCXN.
Rzply/OQhQ9/Dul3j1ex9J2.
jCggY1dbThf/7FoGR/5IYU/.
HBtJI.zWZtP/e1zcT/nCMAfOOsi.K.vC3lT1
ZC8YD.MBoxJ.wtPW61fAKYi1vnu6H/yPVda.
YxPgS13wXdq0m4SMh/4NhJj0hN2gw/J/L.W1
fVN.20dmo331uJaSo/CoFfs1RYrQy.lHqPM.bjDB6.d22dU.
YtqCd0dK7Ts1Ej1ZC0SplR2/pdxlr/i.KQu1L8JvE0iBK82/
aoQLZ/DVMQD0CWvM8.x1CkA0oEMAd.bf8PG13y62h0YUKEV.
pNdkb0wdFFa.mcNo21rXfue/gz6OS/jjCvK.OfNC50tvylg0
Erre71dq9e8Or/Z1k.ZxMC4/IbM24/jNtv1O0fNC50tvylg0
w1UOB1pMDRr.OqcNd.5sPlg/dFB9ZOp4z3J.AyQVA1f1nog.
SAOwF06ysjk.ATOHB/OwpNn.GBKBQ.lT9ac.j3gqyOZZj8T0
iS5p5.APaU918QMt5/rUrqf1CZq6V.HvXaO10fNC50tvylg0
IqmDK175dGq/Ow9v9/mf9gh0s9/Fh.peG6R.AyQVA1f1nog.
N5TOC/1u1bE0U2GFXOMZh5w/fMjQP/niXB313E0aH/xnUxd1
MalYj1fUwXM.uLcZ/O.TBli1KK/Ky/4m/BN.AyQVA1f1nog.
1g5tU1fq2li/wbOG1/S6wyW/8YvozONqvDQ/bjDB6.d22dU.
DSvu2/EYtmt/U7Q08.WyTaM/nZ32s1RLavG/FTYri1vWkZ31
ImlGa/nWGQiOJoaQo/q/X9V04k6mG.DkBte/B5teO.IWVT.OAyQVA1f1nog.
ZKo4Y/4NvF50g.CdC11I4Rb1ukUWMOVIUYiOWAuXO/aHA8//FTYri1vWkZ31
1Dnc519RXj5/ZKYj.1RgFD11X4yan/uhZ2u10fNC50tvylg0
PMAn9.Z2Ang/Z0AGn.BZBjX/SrsHz.sXA9C/bjDB6.d22dU.
4tp8F0fi5v90K4S321M.Qmk.WSrZw0TeLGAOFTYri1vWkZ31
ruWga099oyZ1Ip5Q5/q8X.x0SGIr812OzqVO0fNC50tvylg0
jOM3j.dmXdW.7PMpV/jsOMb1ukUWMOVIUYiOWAuXO/aHA8//FTYri1vWkZ31
te9Vc.UOv5q.T8muJ/pos1Z/YYobk/NsEF5O
aRER8.2gZ8z.pA31tOQz2RO/UI8xj.kS45FO
a9Kbt//i7wy.gkfbnOt8luZ/TUkG4OgfzWTOFTYri1vWkZ31
TTuC709yXN2.SwLbFOi/U1C/6UJKy.piaMQ.NNDAD/Y2LDO.
PTRNt/fXRae1PAI5F.wwdIr0mgv9m1L7nSH1a5wCd1/8uAA.
FnmOh0pKIfn.DZpnf1IiWbK.ZVT8I.dBehy/zpVde/NZHSL.
6BCxZ1a6a8//QGUCq/t2ooq1if/IH.VP0UyOFTYri1vWkZ31
s7Y5O12/86o15cBz41N2RTI1JZtph1ZHoyk1a5wCd1/8uAA.
Iukwg1im4L516hjV6/Qts2v/sRbJq1qOJLS.NNDAD/Y2LDO.
TKc95/h7wLG16nZXaOqfCXP0NP/n614tzX1/3y62h0YUKEV.
u1mwB.Pe54FOEzEoROjQvyP.fPzG5.fqrcc/FTYri1vWkZ31
7pwCx1eO.wz/zGSxu1d8RCc/krKD8.1FtZ2/NNDAD/Y2LDO.
98tq41asCYH/L3rcJ/Fim/z0jBSP5/013o5.NNDAD/Y2LDO.
```

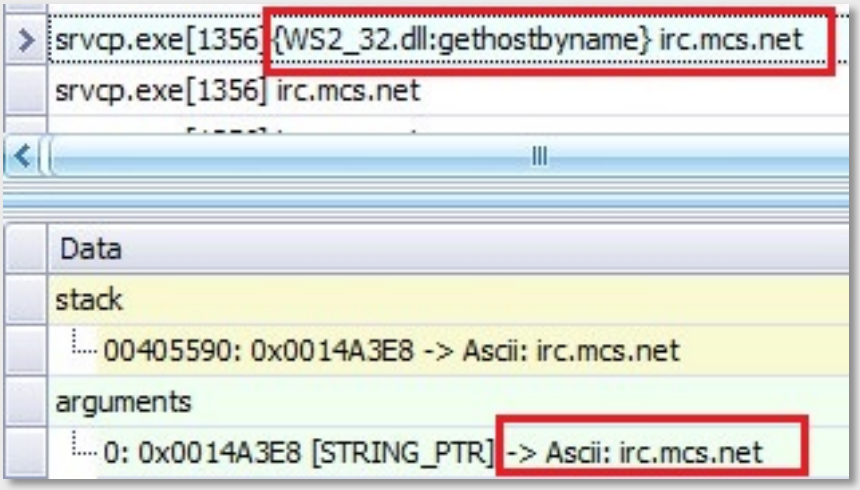
DECRYPTED GUS.INI

```
NICK=mikey
MODE=AGGRESSIVE
SETCOMMAND=setpr
COMMAND=fuckedup
CHANNEL=mikag soup
SOUPCHANNEL=alphasoup ah
SERVER0=irc.mcs.net:6666
SERVER1=efnet.cs.hut.fi:6666
SERVER2=efnet.demon.co.uk:6666
SERVER3=irc.concentric.net:6666
SERVER4=irc.etsmtl.ca:6666
SERVER5=irc.fasti.net:6666
SERVER6=irc.idle.net:6666
SERVER7=irc.powersurfr.com:6666
SERVER8=irc.total.net:6666
SERVER9=irc.core.com:6666
SERVER10=irc.inter.net.il:6666
SERVER11=irc.umn.edu:6666
SERVER12=irc.prison.net:6666
SERVER13=irc.isdnet.fr:6666
SERVER14=irc.ced.chalmers.se:6666
SERVER15=irc-e.frontiernet.net:6666
SERVER16=irc.best.net:6666
SERVER17=irc.exodus.net:6666
SERVER18=irc.enitel.no:6666
SERVER19=irc.telia.se:6666
SERVER20=irc-w.frontiernet.net:6666
SERVER21=irc.du.se:6666
SERVER22=irc.rt.ru:6666
SERVER23=irc.freei.net:6666
SERVER24=irc.homelien.no:6666
SERVER25=irc.colorado.edu:6666
SERVER26=irc.mindspring.com:6666
SERVER27=irc.umich.edu:6666
SERVER28=irc.stanford.edu:6666
SERVER29=irc.nethead.com:6666
SERVER30=irc.lightning.net:6666
SERVER31=irc.emory.edu:6666
SERVER32=irc.spynet.com:6666
SERVER33=ircd.lagged.org:6666
```

The analysis located the use of the function WSAStartup, which initiates the use of the Winsock DLL by the calling process.


Strings being loaded into active memory identified the use of Winsock 2.0 by this particular malware sample.

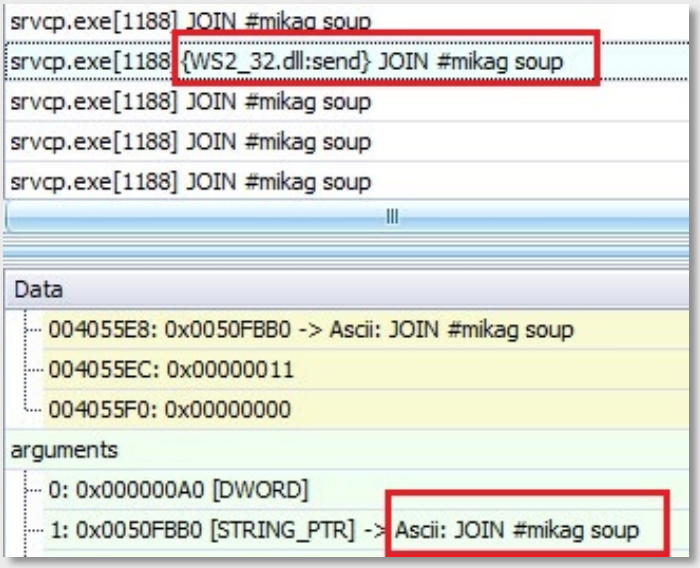

The malware sample attempts to resolve irc.mcs.net.




The malicious process establishes a socket to the resolved irc.mcs.net.


The malware sample then begins listening for an incoming connection on that socket and then accepts a connection attempt from irc.mcs.net.

Once a connection has been established, the malware attempts to join the IRC channel “mikag” using the key “soup”.

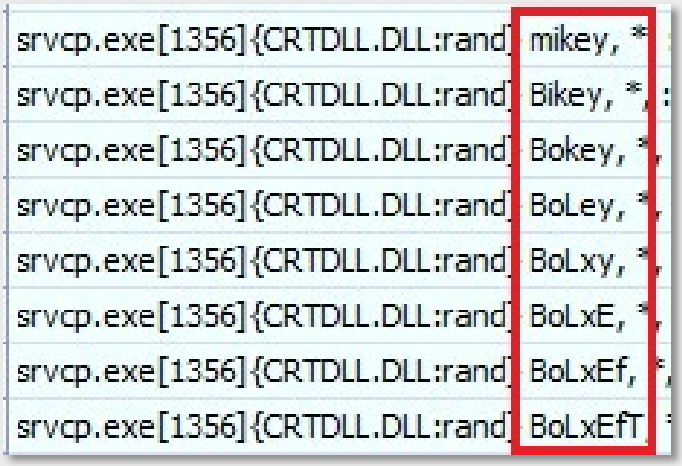




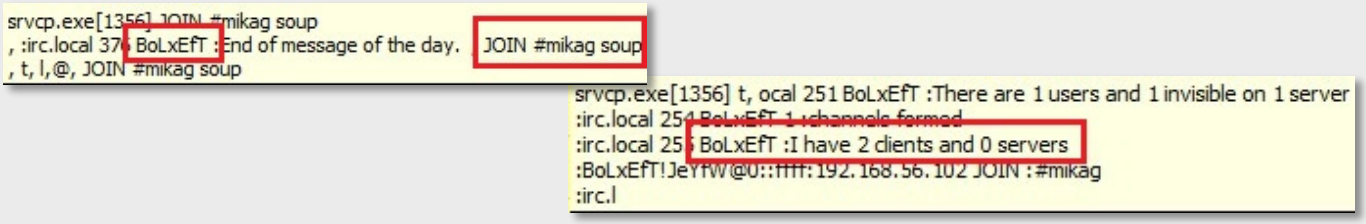
In the event that the nick “mikey” is already being used by another compromised system on the IRC channel, the malware receives notice from the IRC server that the nick is already in use.


The malware essentially transforms the nick “mikey” into a random set of characters using the rand function. First, the malware uses the GetTickCount function to obtain the number of milliseconds that have transpired since the system was started. The return value from this function is 0x00091726, or 595,750.

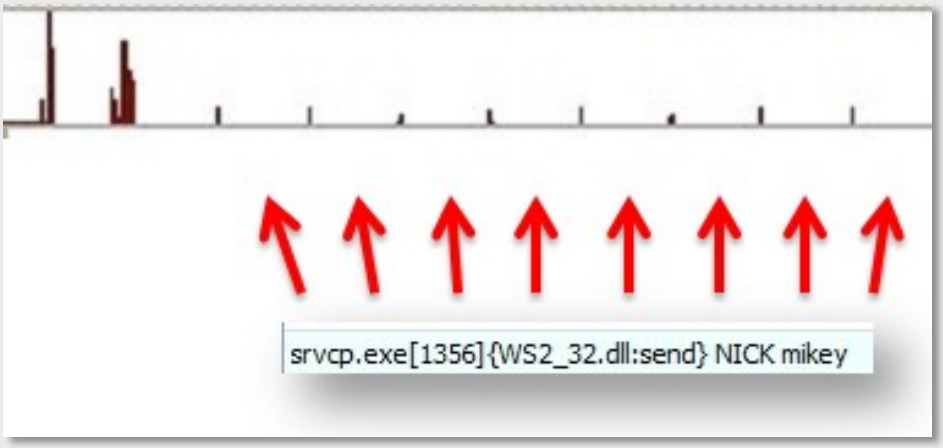

595,750 milliseconds represents approximately 9.92 minutes. The srand function is used with 595,750 as an integer seed value to generate a different succession of random results in following calls to rand. Below shows the progression of the use of the rand function to transform “mikey” to “BoLxEfT”.




Once this process is completed, the malware joins “mikag” using this nick.



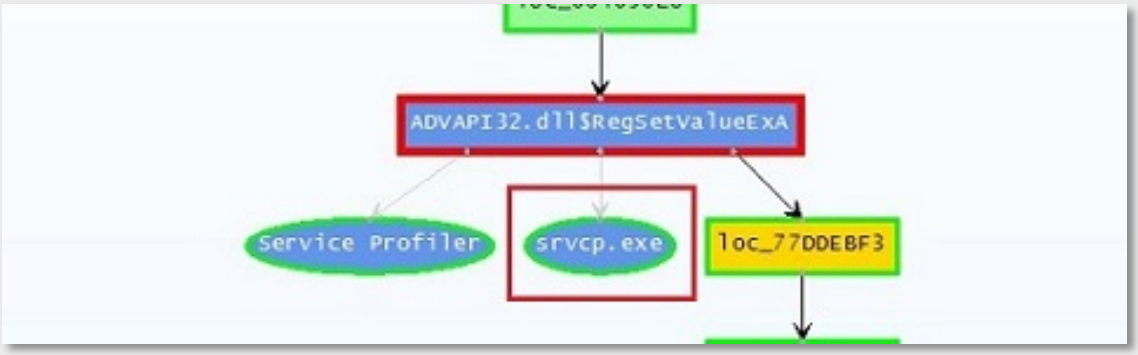

Once the malicious process successfully joins the IRC channel using the random nick, it sends the command “NICK mikey” every three seconds, attempting to change its nick to “mikey”.



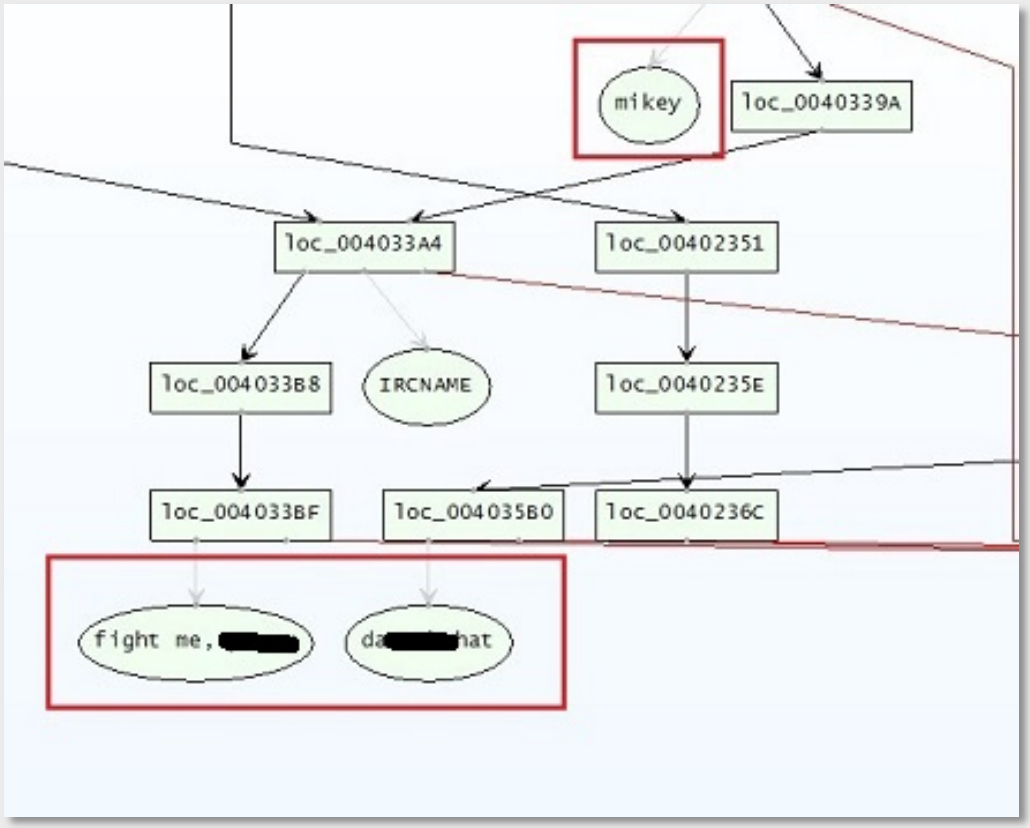
## Active Memory Analysis

An analysis of the first active memory image was conducted. This image of active memory was taken from an infected system on which the gus.ini file was not present.

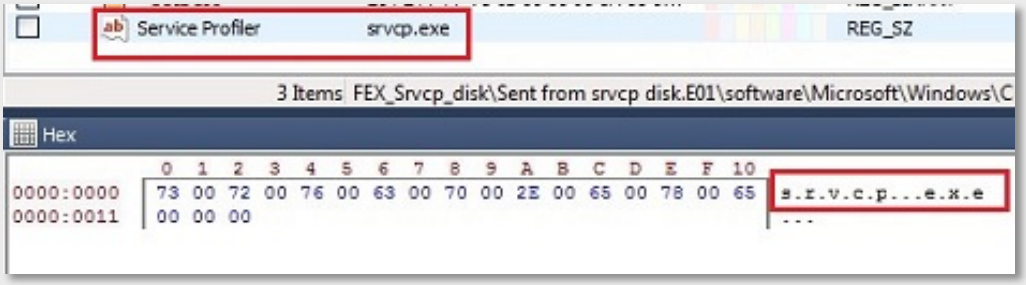
The access and modification of the registry location SOFTWARE\Microsoft\Windows\ CurrentVersion\Run, for the purposes of persistence, was located and mapped out within active memory.




The analysis of active memory also located the decrypted strings used by the malware to connect to the IRC command and control server (example shown below).

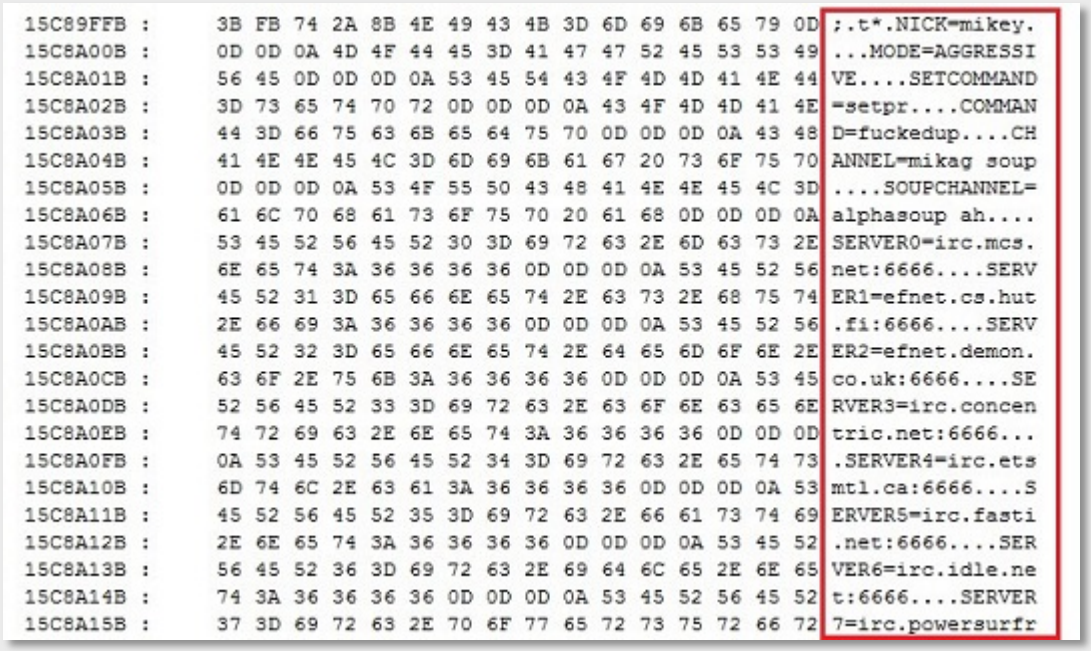


An analysis of the SOFTWARE hive from the disk image confirmed the presence of this registry entry.

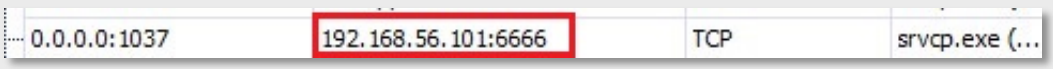



In addition, the analysis of active memory identified the malicious process SRVCP.exe listening for command and control communications on TCP port 113.


An analysis of the second active memory image was conducted. This image of active memory was taken from an infected system which was provided a connection to an IRC server and on which the gus.ini file was present in the location C:\Windows\system32. This analysis located the entire gus.ini file in active memory after it had been decrypted by the malware sample to be utilized to contact IRC command and control servers (partial screenshot below).




In addition, the analysis of active memory identified the malicious process SRVCP.exe with an established socket connection to irc.mcs.net on TCP port 6666.



## Defeating the Enemy

The idea and practice of encrypting or obfuscating malware has been around since the early to mid 1980s. Malware authors quickly realized that to be successful at introducing and spreading their creations; they would need to take steps to not only conceal the malware's behavior, but they would need to take steps to inhibit the progress of those who would reverse engineer the capabilities of these new Cybersoldiers. The longer a malware sample can remain undetected and thwart reverse engineering, the longer it has to spread and complete its mission.

Packers are often used by malware authors to compress the malicious file and thwart static analysis. When a packed file is analyzed statically, only the small packer or "wrapper" program is accessible for analysis. To statically analyze a packed executable, the packing process must be reversed. The increased use of custom packers makes this process increasingly difficult and time consuming for those attempting to statically analyze malware.

Encryption, including the use of XORing code, is also increasingly being utilized to prevent the analysis of malicious files. Custom encoding, such as layered obfuscation techniques using XOR, ROT13, BASE64 or other encryption and encoding processes in conjunction with one another, requires that the custom encoding be reverse engineered and duplicated in order for an obfuscated malicious file to be statically analyzed. In 1990, malware utilizing oligomorphism was detected. Malware using this functionality included numerous decryptors, allowing it to randomly select one to encrypt itself with as it spread to a new file.

These techniques, along with new approaches being developed in the underground, essentially act as "body armor" for this new Cybersoldier. Dynamic Analysis, Trace Analysis and Active Memory Analysis, used in conjunction with each other, offer the capability to negate and neutralize these defensive traits. Trace analysis allows for the static analysis of a "recorded" view of a malware sample after it unpacked, decrypted or de-obfuscated at runtime. In addition, Active Memory analysis allows for the examination of the execution of a malware sample on a compromised system, including legitimate processes that may have been compromised. This analysis can be performed despite the use of obfuscation or encryption, even if malicious files on the hard drive are not available. This multi-layered methodology provides an analyst with multiple avenues for analysis which complement each other and provide a means to analyze advanced malware which may employ "body armor" in the form of packers, obfuscation or other encryption techniques.