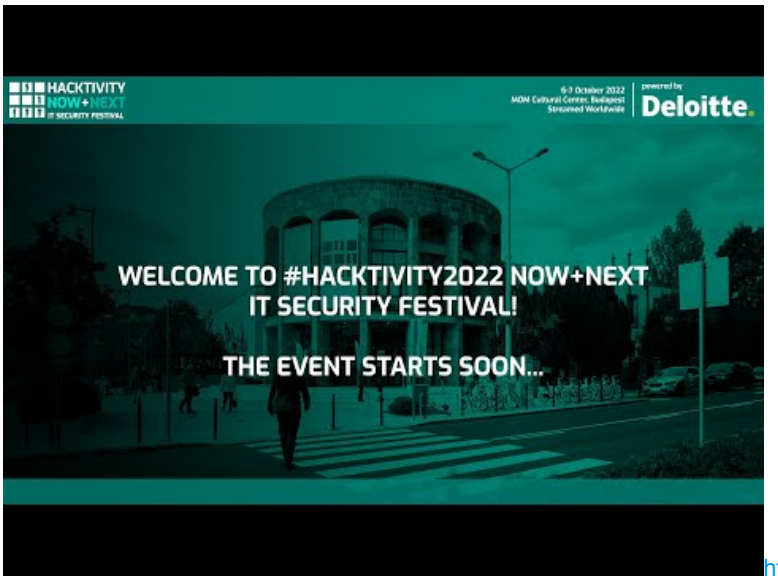
DiceyF deploys GamePlayerFramework in online casino development studio

Authors

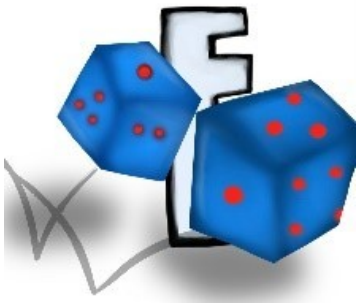
-  Kurt Baumgartner
- Expert Georgy Kucherin

The Hacktivity 2022 security festival was held at the MOM Cultural Center in Budapest, Hungary, over two days, October 6-7th 2022. One of several presentations by our GReAT researchers included an interesting set of APT activity targeting online casino development and operations environments in Southeast Asia. A recorded video of the presentation is already online. You can watch it here:



https://youtu.be/yVqALLtvkN8

All of our research, including a full set of IoCs and Yara rules, is written up in the two-part report “DiceyF Deploys GamePlayerFramework in Online Casino Development Studio”, already available to our private report subscribers. Some technical analysis from the report is provided here, along with a reference set of IoCs. You can find more information about trial and pay report subscription options at intelreports@kaspersky.com.



Who is at the table

We call this APT “DiceyF”. They have been targeting online casinos and other victims in Southeast Asia reportedly for years now. Our research shows overlap with LuckyStar PlugX, a supply chain incident privately reported. TTPs, secure messaging client abuse, malware, and targeting demonstrate that this set of activity and resources align with Earth Berberoka/GamblingPuppet activity discussed at Botconf 2022 by Trend Micro researchers, also discussed as an unknown or developing cluster by other vendors. Prior to “Operation Earth Berberoka”, Trend Micro reported on “Operation DRBControl”, which also aligns with this activity and resource set.

So, do we have another *Ocean’s Eleven* Clooney-Pitt duo targeting the largest casinos for shocking levels of criminality, revenge, and theft? No we don’t. In the related DiceyF incident that we report on, there was no evidence observed to date of immediate financial motivation or cash theft. Instead, previous incidents reported by TrendMicro researchers have exhibited customer PII database exfiltration and source code theft. Possibly we have a mix of espionage and IP theft, but the true motivations remain a mystery.

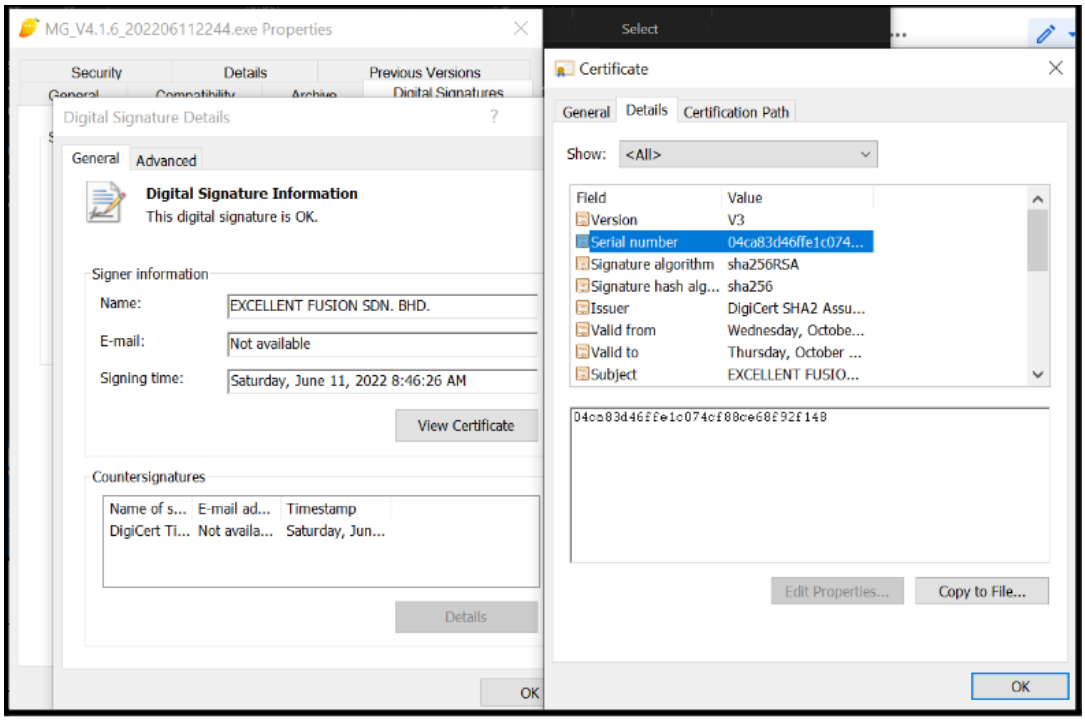
Rolling the dice

An interesting combination of detections and characteristics sparked interest in this activity. These data points included

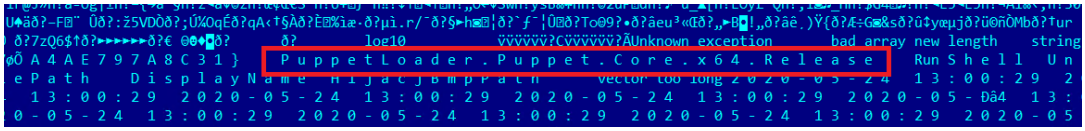
- PlugX installers signed by a potentially stolen digital certificate from a secure messaging client development studio
- Malware distribution via an employee monitoring system and a security package deployment service
- Unusual .NET code signed with the same potentially stolen certificate and calling back to the same domain as the PlugX C2

In November 2021, multiple PlugX loaders and payloads were detected in a network, which is often a wearisome topic to investigate. However, this time, the PlugX installer triad was deployed via two methods as an executable signed with a legitimate digital certificate — an employee monitoring service and a security package deployment service. This legitimate digital certificate appeared to have been stolen from a development and build studio for a secure messaging client. These PlugX payloads communicated with a C2 at apps.imangolm[.]com. Not much later, this same security package deployment service was used to push GamePlayerFramework downloaders, with these downloaders communicating with the same C2, and signed with the same digital certificate.

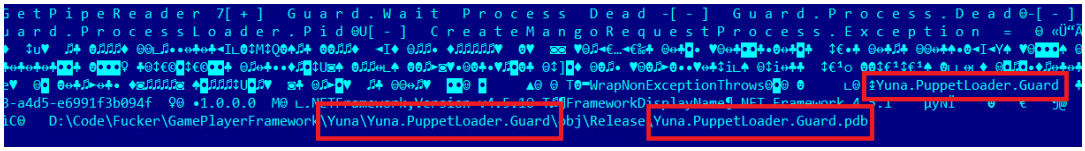
Further research revealed a targeting profile suggesting an online casino development studio, and later, recruited/outsourced development systems on disparate networks. Waves of .NET downloader deployments followed and coincided with the PlugX deployments, signed by the same digital certificate.



These downloaders maintained PDB strings with “PuppetLoader” filepaths. These PuppetLoader strings pretty clearly connected the multistage loaders with past PuppetLoader downloaders, only this time, redesigned and rewritten in C#. Past PuppetLoaders, written in C++, maintain explicit strings:



The new .NET code maintains similar strings, reflecting the previous codebase from several years ago.



While these findings were being analyzed and reported, the folks from Trend Micro reported on GamblingPuppet/Earth Berberoka at Botconf, and we are confident that this DiceyF GamePlayerFramework activity is a subsequent campaign with a newly developed core malware set. This APT, DiceyF, aligns the previously reported GamblingPuppet and Operation DRBControl resources and activity, which we also observed in earlier data as well:

- PlugX and PuppetLoader multistage loader
- Online casino targeting in Southeast Asia
- Lack of evidence presenting a financial motivation (Trend Micro observed customer database and source code exfiltration in Operation DRBControl)
- Chinese language in use, particularly GamePlayerFramework error strings and plugin names and paths
- Data theft focus for backdoors includes keystrokes and clipboard
- Stolen digital certificate re-use
- Obscure secure messaging client as delivery vehicle for malware and cloak for malicious activity

GamePlayerFramework is a complete C# rewrite of the previously mentioned PuppetLoader C++/assembly malware. This “framework” includes downloaders, launchers, and a set of plugins that provide remote access and steal keystrokes and clipboard data. The newer (summer 2022) executables are mostly all 64-bit .NET compiled with .NET v4.5.1, but some are 32-bit, or DLLs and compiled with .NET v4.0. There are at least two branches to this framework, “Tifa” and “Yuna”, and both branches maintain new modules, incrementally modified over time:

- D:\Code\Fucker\GamePlayerFramework\Tifa*.pdb
- C:\Users\fucker\Desktop\Fucker\GamePlayerFramework\Tifa*.pdb

- D:\Code\Fucker\GamePlayerFramework\Yuna*.pdb

FinalFantasy code quirks

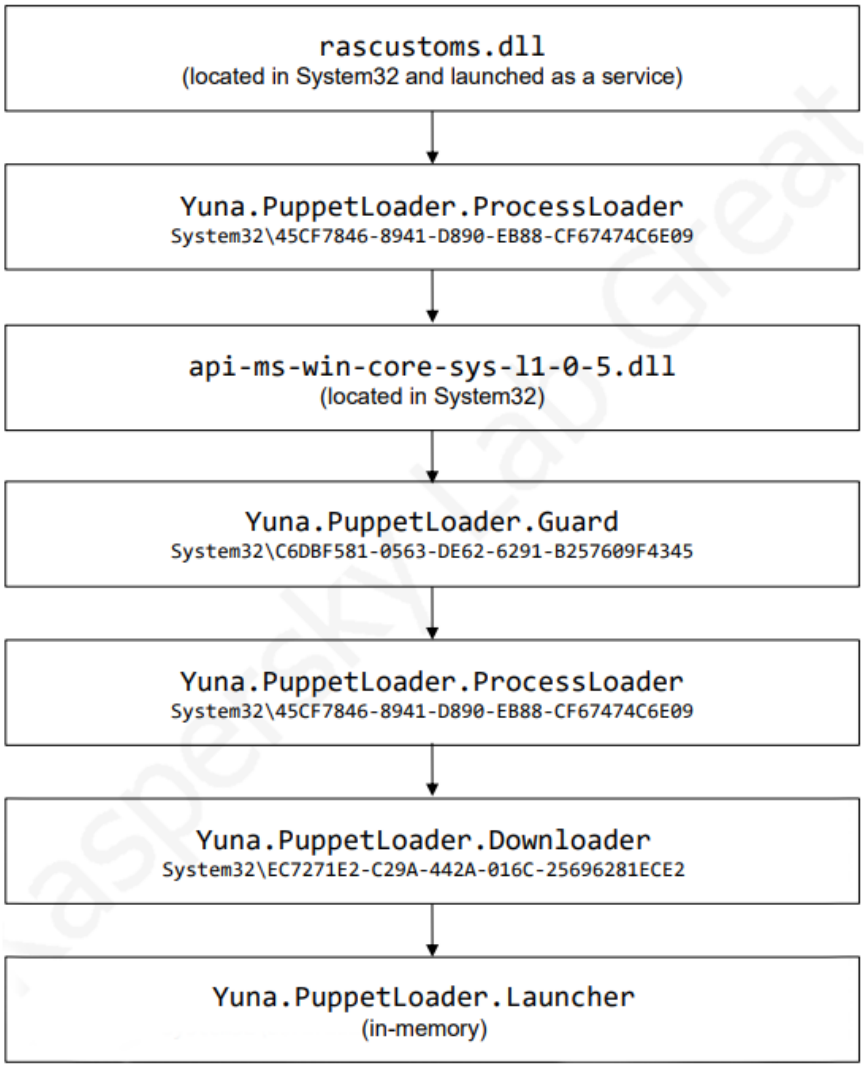
Players may be familiar with the *FinalFantasy* game series, where Tifa and Yuna are two of the main characters. The Tifa and Yuna branches are different from one another: the Tifa branch includes only a downloader and a "core" module; the Yuna branch includes a downloader, plugins, and various PuppetLoader components, at least a dozen in total. Even the downloaders are fairly different from one another. As a matter of fact, the Yuna.Downloader code changes quite a bit over time, including with JSON parsing, logging, and encryption capabilities. Ongoing code development is on display here.

The Tifa branch of code was deployed to victims first in November 2021, and these Tifa downloaders maintain more primitive functionality than the later Yuna downloaders. Additionally, it appears that code-signing coordination was not well organized in November. Except for one Tifa executable that was signed, two of the three Tifa downloaders were unsigned code, unlike the Yuna downloaders.

The initial Tifa downloaders were already using "Mango" and "Mongo" function names just like artifacts found in Yuna downloaders, along with the aforementioned apps.imangolm[.]com C2 used by the PlugX implant. Later Yuna downloaders were distributed with the filename "mango.exe". Two of the Tifa.Downloader variants introduced a "DownloaderVersion" string, likely for the attackers to maintain backwards compatibility on the server side. Some later Yuna.Downloader variants increase in functionality and complexity, but multiple early variants and the Tifa branch are quite simple.

Loading the framework

Once downloaded and persistence set up, multiple components load the framework. The overall process of loading the framework can be summarized with the following graph:



This load sequence results in running the “Launcher” component. Despite the name, the main functionality of this module is not to perform launching. Instead, it is the orchestrator of the framework, i.e. it manages all the framework components. After completing the startup process, the orchestrator starts sending heartbeat packets to the C2 server every 20 seconds. Each such packet is a XOR-encrypted JSON object that contains the following information:

- Username of the logged-in user
- Current user session status (locked or unlocked)
- Size of logs collected by the clipboard recorder plugin
- Current date and time

The C2 responds with one of fifteen commands.

Command name	Command arguments	Description
PluginKeepAlive, KeepAlive	N/A	Updates an internal timestamp with the last C2 response time
PluginDestory [sic]	N/A	Shuts down the framework
GetSystemInfo	N/A	Retrieves various system information, namely: • Local network IP addresses • Available privileges (SYSTEM, administrator or normal user) • Network protocol used for C2 communication (hardcoded to Tcpv4 in all discovered samples) • Framework version (in format yyyymmdd.xx, e.g. 20220506.00) • Downloader module version • CPU name

		• Available RAM • Operating system version • Address of the C2 server that is in use • Size of clipboard recorder logs • Installed security solution • BIOS serial number • MAC addresses • Machine boot time
FastCmd	Command: command to be executed	Allows execution of shell commands; this command creates a new cmd.exe process with redirected standard input and output and sends commands to it; the output of executed commands is sent back to the C2 server
GetDomainSetting	N/A	Sends the list of C2 servers specified in the configuration to the current C2 server
SetDomainSetting	DomainConfig: IP addresses and ports of new C2 servers	Updates the list of C2 servers in the configuration by writing new C2 server addresses to the file C:\ProgramData\NVIDIA\DConfig
GetRemotePluginInfo	PluginName: name of an installed plugin	Retrieves the version of a locally installed plugin
RunPlugin	PluginName: name of the plugin to be launched SessionId: ID of the session inside which the plugin is to be launched	Downloads a plugin from the C2 server and launches it
DeleteGuid	N/A	Removes the infection from the machine by creating a batch file that removes all files dropped by the framework installer except for rascustoms.dll; after performing removal, the batch file deletes itself
FastDownload	FilePath: path of the file to be uploaded	Uploads a file from the victim machine
CachePlugin	PluginName: name of the plugin PluginVersion: version of the plugin	Downloads a plugin from the C2 server but does not launch it
InstallPlugin	PluginName: name of the plugin to be launched WaitForExitTimeout: timeout interval	Launches a plugin on the victim machine, waiting until the plugin process finishes; in case of a timeout, the orchestrator kills the plugin process
RemoteInject	SubMsg: a string equal to either RunVirtualDesktop or DestoryVirtualDesktop [sic]	Either starts (if SubMsg is RunVirtualDesktop) or stops (if SubMsg is DestoryVirtualDesktop) the VirtualDesktop plugin
ChromeCookie	SubMsg: a string equal to either RunChromeCookie or GetCookiePath	If SubMsg is RunChromeCookie, launches the ChromeCookie plugin; if the argument string is GetCookiePath, returns the path where Chrome cookies are stored
FirefoxCookie	SubMsg: a string equal to either RunFirefoxCookie or GetCookiePath	If SubMsg is RunFirefoxCookie, launches the FirefoxCookie plugin; if the argument string is GetCookiePath, returns the path where Firefox cookies are stored

Plugins overview

Plugins are EXE files that execute most of the framework's malicious activities. Plugins can be configured to be downloaded from the C2 server when the framework starts up or is loaded at any other time using one of the commands above. During its execution, a plugin may connect to the C2 server and receive commands from it. Information about running plugins is stored in the C:\ProgramData\NVIDIA\DisplaySessionContainer1.ini file.

All plugins of the framework are stored in a fileless way. Whenever a plugin is downloaded from the C2 server, it is loaded into the framework with the following procedure:

- The orchestrator selects a random port from 10000 to 20000 and launches a local TCP socket server on it.
- The orchestrator creates a new svchost.exe process in suspended mode and injects the api-ms-win-core-sys-l1-0-5.dll library mentioned in the "Loading the Framework" section.
- The injected library loads the PuppetLoader.Downloader component with the following arguments: -LoadName <plugin name> -PacketId <internal ID of the network packet with the plugin payload> -Port <server port generated at the first step>.
- The Yuna.PuppetLoader.Downloader component downloads the plugin executable from the local TCP server and loads it using Load.

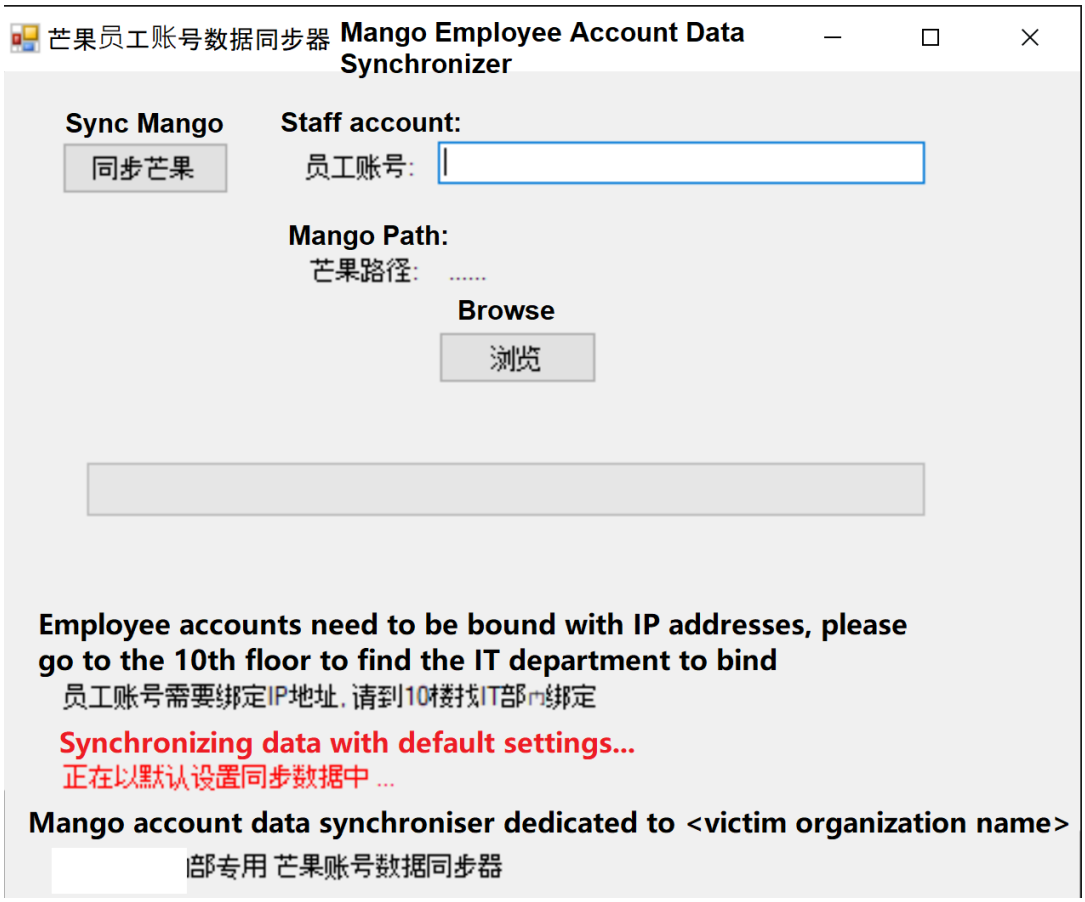
The strings of the orchestrator component reference the following plugin names:

- Plugin.采集系统 (Acquisition System)
- Plugin.隐藏进程 (Hidden Process)
- Plugin.SSH
- Plugin.常用功能插件 (General Purpose Plugin)
- Plugin.SessionCmd
- Plugin.端口转发 (Port Forwarding)
- Plugin.屏幕传输 (Screen Transfer)
- Plugin.虚拟桌面 (Virtual Desktop)
- Plugin.剪贴板 (Clipboard)
- Plugin.ChromeCookie
- Plugin.FirefoxCookie

While tracking deployments of GamePlayerFramework, we observed several plugins out of the list above being used: General Purpose Plugin, Clipboard and Virtual Desktop.

Malicious app with graphical interface

The application deployed through installation packages of security solutions was designed to mimic an application that synchronizes data of the Mango messaging application. Below is the window displayed to the victim when this application starts:



Window of the malicious "Mango Employee Account Data Synchronizer"

In order to make the victim user trust the malicious window, the attackers employed social engineering. As can be seen from the screenshot above, they included the name of the victim organization and even the floor where the organization's IT department is located. At the same time, the visible window makes this application less suspicious to security solutions.

When started, this application:

- Connects to the C2 server via a TCP socket. The address and port of the server is specified in the binary. In case connection fails, the application displays a message window with the "无法连接到芒果员工数据同步服务器! 请反馈至IT部门!" text ("Unable to connect to Mango employee data synchronization server! Please report back to the IT department").
- Sends the following information to the C2 server:
- Version of the installed Mango messenger
- Machine name
- Current username
- Operation system version
- List of local IPv4 addresses
- Receives a JSON object containing a Boolean value named IsErrorMachine. If it is set to true, the application displays a message window with the "尚未认证的机器, 请到10楼的IT部添加机器认证" text ("Machines that have not been certified, please go to the IT department on the 10th floor to add machine certification") and exits.
- Launches the exe executable located inside the same directory as the application. The internal name of this file is Yuna.Downloader.

The code is under continuous incremental change and its versioning reflects a semi-professional management of the codebase modifications. Over time, the group added Newtonsoft JSON library support, enhanced logging, and encryption for logging.

Infrastructure

Domain	IP	First seen	ASN

apps.imangolm[.]com	202.182.115.238	20211106	20473, AS-CHOOPA
quic.flashesplayer[.]com	202.182.115.238	2021-11-10	20473, AS-CHOOPA
archivess.imangoim[.]net	45.77.47.149	20220506	20473, AS-CHOOPA

As described above, much of the early implants' (both PlugX and the downloaders) communications activity calls back to infrastructure by resolving FQDN for infrastructure located in Southeast Asia. Later into April 2022, some of the Yuna.Downloaders began communicating directly with a hardcoded IP address.

Conclusion

There are many interesting characteristics of DiceyF campaigns and TTPs. The group modifies their codebase over time, and develops functionality in the code throughout their intrusions.

Organizations need to maintain solid efforts in monitoring software deployed across their organizations. The deployment systems themselves and the deployment process require heightened monitoring and maintenance: what gets deployed, when it gets deployed, and whose credentials are being used. The systems themselves need to be hardened and security products installed and updated.

GamePlayerFramework enabled DiceyF, the actor behind this framework, to perform cyberespionage activities with some level of stealth. The initial infection method is noteworthy in that the framework is distributed via installation packages deployed through security solution control centers. Furthermore, the components of this framework are signed with a digital certificate that makes the framework more trusted by security solutions. In order to further disguise the malicious components, attackers added a graphical interface to some of them. Such implants are masqueraded as components of a messenger that is used at the victim organizations. To make sure that victims did not become suspicious of the disguised implants, attackers obtained information about targeted organizations (such as the floor where the organization's IT department is located) and included it inside graphic windows displayed to victims. They also used service names, file paths, digital signing certificates, and other artifacts from NVIDIA, Mango, and other legitimate software. Plugins of GamePlayerFramework allow extensive monitoring of victim machines. For example, they are able to monitor keystrokes and the clipboard, browse websites located inside the organization's local network, or establish virtual desktop sessions. And over the course of several months, DiceyF developers added more encryption capabilities to better hide their logging and monitoring activities. In the future, we expect to see an increase in the number of plugins and observe more unusual defense evasion methods in this framework.

Finally, the deployment tactic used here isn't quite as sophisticated as infecting an external component of the supply chain itself, but can be extremely effective.

IOCs

MD5

Tifa.Downloader
ddbc9081ed2c503c5e4512a8e61b5389

Tifa.Core
49b457ee8eaa83b18cc00d2f579824c6

Yuna.Downloader
06711900cc5d7cd665bc1b6ec9d7eacf
1d59e527886e4bd72df0f609239b9d58

Yuna.Downloader and Yuna.Launcher containing legitimate Newtonsoft DLL
0c4dae01f21c3d2fa55f38314fe34958
39736c93f7d9cc62cdc00438c174f8a4

Yuna.Launcher
07d6bf2df064e97d0e635a67f083f87d
0ac4e0e08bd28e88acd4991071c98261

Yuna.Plugin.General
cb8a30fcbcb462be66462f6928c6e44a

Yuna.Plugin.ClipboardRecorder
294c22533c950d7d9d74a82729ba3841

Yuna.PuppetLoader.CodeLauncher
07ff76be283fb44ce9e9427e12e63aa6

Yuna.PuppetLoader.Guard
031466c63bba4eafb11f2966e765c0d2

Yuna.PuppetLoader.Downloader
0c4dae01f21c3d2fa55f38314fe34958
19f8809d04c06bba2ad95a937f133a89

Yuna.PuppetLoader.ProcessLoader
969ef4a64203ba2ab54a6822559600cc

Yuna.Downloader.DLL.Core
56836b19b5c35c81e006f4843ff63e51

Mango.Sync.Updater
3a1780e6fb6250b0fb63d2884788670e
4d72e573d9c4d31371c8020ba7179daf

VPN spoofing DLL
193d192ed0cec2487d18b13aedc94cb6

rascustoms.dll
2bd3b84b318beb5714cac9194078607a

Domains
apps.imangolm[.]com
archivess.imangoim[.]net

PDB paths
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.Downloader\obj\Release\Yuna.Downloader.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.Launcher\obj\Release\Yuna.Launcher.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.Plugin.ClipboardRecorder\obj\Release\Yuna.Plugin.ClipboardRecorder.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.Plugin.General\obj\Release\Yuna.Plugin.General.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.Plugin.Installer\obj\Release\Yuna.Plugin.Installer.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.PuppetLoader.CodeLauncher\obj\Release\VpnSohDesktop.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.PuppetLoader.Downloader\obj\Release\Yuna.PuppetLoader.Downloader.pd
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.PuppetLoader.Guard\obj\Release\Yuna.PuppetLoader.Guard.pdb
D:\Code\Fucker\GamePlayerFramework\Yuna\Yuna.PuppetLoader.ProcessLoader\obj\Release\Yuna.PuppetLoader.ProcessLoa