ShadowPad: the Masterpiece of Privately Sold Malware in Chinese Espionage

Yi-Jhen Hsieh, Joey Chen
Outline

- From PlugX to ShadowPad: the history
- Technical overview
- The business model of ShadowPad
- Which threat actors are using ShadowPad?
- Landscape shift: from developing backdoors to acquiring backdoors
- Mitigation advice
From PlugX to ShadowPad: the history
What is ShadowPad?

- A modular backdoor in shellcode format.
- The plugins can be plugged or unplugged remotely during runtime.
- It is used in several well-known campaigns as the primary backdoor for intrusion.
Which threat actors are using ShadowPad?

- WINNTI/APT41
- Tick and Tonto Team
- Fishmonger
- Operation Redbonus
- Operation Redkanku
- ShadowHammer
- NetSarang
- JP JAXA
- CCleaner
- Tropic Trooper
- LuckyMouse
## ShadowPad: A successor to PlugX?

<table>
<thead>
<tr>
<th>Feature</th>
<th>ShadowPad</th>
<th>PlugX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shellcode</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Plugin-based</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Remote plugin management</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Distribution</td>
<td>Privately shared</td>
<td>Widely used</td>
</tr>
</tbody>
</table>
The relationship between ShadowPad and PlugX

- Comparisons on their codes have been discussed[1]
- They share the same project name “SC” according to the PDB strings of their controllers
  - PlugX Controller: D:\My2014\SController(5.6)(天道)(匙)\SC\n  - ShadowPad Controller: X:\My2015\SC(1.1)x64\Release\SoSvr.pdb

The alleged author of PlugX: whg

- whg aka 无花果, based in the Sichuan province of China
- His nickname was found in some of the PlugX samples[2]:
  - C:\Documents and Settings\whg\桌面\Plug\FastGui(LYT)\Shell\Release\Shell.pdb
  - C:\Users\whg\Desktop\Plug\FastGui(LYT)\Shell\Release\Shell.pdb

- He had a solid track record of developing backdoors and hacking tools, while he claimed to sell some “shared software”.
  - Also, he had deep connection with Rose, one of the threat actors in APT41.

Technical overview
The modular design
The shellcode loader

```
loc_45F26F:
push ebp
mov ebp, esp
sub esp, 430h
push ebx
push esi
push edi
jns short near ptr loc_45F27F+1
js short near ptr loc_45F27F+1

loc_45F27F:
call near ptr

; db 0
; db 0
mov eax, [eax+0Ch]
mov ebx, [eax+0Ch]
xor edi, edi
jmp short loc_45F200

loc_45F290:
mov esi, [ebp+30h]
xor eax, eax
mov [ebp-4], eax
cmp [esi], di
jz short loc_45F2CF

loc_45F29D:
jge short near ptr loc_45F2A1+1
jl short near ptr loc_45F2A1+1
```

```
loc_45F280:
mov eax, large fs:30h
mov ebx, [eax+0Ch]
mov eax, [eax+0Ch]
mov ebx, [eax+0Ch]
xor edi, edi
jmp short loc_45F200

loc_45F290:
mov esi, [ebp+30h]
xor eax, eax
mov [ebp-4], eax
mov [edi-4], eax
cmp [esi], di
jz short loc_45F2CF

loc_45F29D:
jge short near ptr loc_45F2A2
jl short near ptr loc_45F2A2
```
The plugins

- Every plugin is decrypted, loaded into memory, and referenced in a LinkedList.
- Additional plugins could be uploaded from the C&C servers.

```
struct plugin_node {
    plugin_node* previous_node;
    plugin_node* next_node;
    DWORD referenced_count;
    DWORD plugin_timestamp;
    DWORD plugin_id;
    DWORD field_0;
    DWORD field_1;
    DWORD field_2;
    DWORD field_3;
    DWORD plugin_size;
    LPVOID plugin_base_addr;
    LPVOID plugin_export_function_table_addr;
};
```
The start function of a plugin

- 0x01: Setup the export function table of the plugin
- 0x64: Setup the plugin
- 0x66: Return the plugin ID
- 0x67: Return the plugin name
- 0x68: Return the address of the export function table

```c
int __stdcall start(_BYTE *plugin_base, int command, DWORD *return_value) {
    DWORD *p_str_obj; // eax
    DWORD str_obj[2]; // [esp+0H] [ebp-10H] BYREF
    if ( command ) {
        if ( command == 1 ) // setup plugin export function table
            export_func_table = main_function;
        else if ( command == 0x64 ) // plugin installation
            function_table_base = (int)return_value;
        else if ( command != 0x65 ) {
            switch ( command ) {
                case 0x66: // return plugin ID
                    *return_value = 0x137;
                    break;
                case 0x67: // return plugin name
                    p_str_obj = str_decrypt(encrypted_plugin_name, str_obj);
                    strcpy(*return_value, p_str_obj[2]);
                    str_destroy_obj(str_obj);
                    break;
                case 0x68: // return the address of plugin export table
                    *return_value = 0x6E4000;
                    break;
            }
        }
        return 1;
    }
}```
22 unique plugins

Basic Set
- Root
- Plugins
- Config
- Install
- Online
- TCP
- HTTP
- UDP
- DNS

Utilities
- ImpUser
- PIPE
- Disk
- Process
- Service
- Register
- Shell
- PortMap
- Keylogger

- Screen
- Software
- Hardware
- RecentFiles
The configuration

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data Type</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>WORD</td>
<td>offset_product_key</td>
</tr>
<tr>
<td>0x02</td>
<td>WORD</td>
<td>offset_note</td>
</tr>
<tr>
<td>0x04</td>
<td>WORD</td>
<td>offset_binary_path</td>
</tr>
<tr>
<td>0x06</td>
<td>WORD</td>
<td>offset_service_name (default: MyTest)</td>
</tr>
<tr>
<td>0x08</td>
<td>WORD</td>
<td>offset_service_display_name (default: MyTest)</td>
</tr>
<tr>
<td>0x0A</td>
<td>WORD</td>
<td>offset_service_description (default: MyTest)</td>
</tr>
<tr>
<td>0x0C</td>
<td>WORD</td>
<td>offset_registry_key</td>
</tr>
<tr>
<td>0x0E</td>
<td>WORD</td>
<td>offset_registry_value</td>
</tr>
<tr>
<td>0x10 - 0x17</td>
<td>WORD</td>
<td>offset_process_spawn_and_inject 1-4</td>
</tr>
<tr>
<td>0x18 - 0x37</td>
<td>WORD</td>
<td>offset_c2 1-16</td>
</tr>
<tr>
<td>0x38 - 0x3F</td>
<td>WORD</td>
<td>offset_proxy_type 1-4</td>
</tr>
<tr>
<td>0x40 - 0x4F</td>
<td>DWORD</td>
<td>DNS 1-4</td>
</tr>
<tr>
<td>0x50</td>
<td>DWORD</td>
<td>timeout_multiplier</td>
</tr>
</tbody>
</table>
New version: the infection chain

- First found in 2020 by PT security[^3]

New version, more obfuscation

- First found in 2020 by PT security\textsuperscript{[3]}
- The control flow is flattened by instruction
The configuration of the new version

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data Type</th>
<th>Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>WORD</td>
<td>offset_date (product key)</td>
</tr>
<tr>
<td>0x02</td>
<td>WORD</td>
<td>offset_note</td>
</tr>
<tr>
<td>0x04</td>
<td>WORD</td>
<td>offset_install_directory</td>
</tr>
<tr>
<td>0x06</td>
<td>WORD</td>
<td>offset_executable_name</td>
</tr>
<tr>
<td>0x08</td>
<td>WORD</td>
<td>offset_loader_name</td>
</tr>
<tr>
<td>0x0A</td>
<td>WORD</td>
<td>offset_payload_name</td>
</tr>
<tr>
<td>0x0C</td>
<td>WORD</td>
<td>offset_service_name</td>
</tr>
<tr>
<td>0x0E</td>
<td>WORD</td>
<td>offset_service_display_name</td>
</tr>
<tr>
<td>0x10</td>
<td>WORD</td>
<td>offset_service_description</td>
</tr>
<tr>
<td>0x12</td>
<td>WORD</td>
<td>offset_reg_key</td>
</tr>
<tr>
<td>0x14</td>
<td>WORD</td>
<td>offset_reg_value</td>
</tr>
<tr>
<td>0x16 - 0x1D</td>
<td>WORD</td>
<td>offset_process_spawn_and_inject 1-4</td>
</tr>
<tr>
<td>0x1E - 0x4F</td>
<td>WORD</td>
<td>offset_c2 1-16</td>
</tr>
<tr>
<td>0x50 - 0x57</td>
<td>WORD</td>
<td>offset_proxy_type 1-4</td>
</tr>
<tr>
<td>0x58 - 0x67</td>
<td>DWORD</td>
<td>DNS 1-4</td>
</tr>
<tr>
<td>0x68</td>
<td>DWORD</td>
<td>timeout_multiplier</td>
</tr>
</tbody>
</table>
The controller

- Version 1.0, 2015
- Written in Delphi
- All-in-one: builder + C&C listener
The management console

- Default plugins v.s. uploaded plugins
- The control pages of the plugins are fixed
The builder

- Campaign code
- Notes
- Anti-debugger settings
- Installation settings (service and registry)
- Process injection settings
- C&C servers
- Connection modes
The business model of ShadowPad
How flexible is the controller?

- The plugins are placed in a directory.
- In theory, you can upload any plugin within the correct format.
How flexible is the controller?

- … but the user cannot interact with the plugins without an interface
  - The interfaces are hardcoded in the controller
  - No options to add a new interface
A piece of sold malware with extendibility

- ShadowPad is not extendible by the users except the original developer.
  - Not originally designed as a framework.

- The developer can remove a plugin from the package easily.
  - Just remove the plugins from the directory.
Selling the plugins separately

Basic set: Root, Plugins, Config, Install, Online, TCP, HTTP, UDP, DNS

The number of plugins embedded in the samples
Selling the plugins separately

A special version: packed by VMProtect, different plugin IDs

The number of plugins embedded in the samples
Selling the plugins separately

With utility plugins embedded

The number of plugins embedded in the samples
Selling the plugins separately (cont.)

- Most of the samples do not have utility plugins embedded in them.
  - Need to upload other plugins remotely through the controllers.
  - The plugins are placed in the directory: easy to be removed from a package

- Tick – an active user of ShadowPad – developed a tool with the plugin “Software” in an overlapped timeframe.
  - Tick did not have access to that plugin while it was already available

- The plugins should be provided separately.
  - Not given in a full bundle.
The business model of ShadowPad

- ShadowPad is a sold malware.
- The plugins need to be acquired separately.
- It is only sold to a limited set of users.

Why is ShadowPad a good choice for the attackers:
- The cost to develop a stable backdoor/RAT is high.
- The plugins (functionalities) are complete, and the attackers can choose which they want.
  - ShadowPad was the primary backdoor for long-term espionage in several campaigns.
- The use of a shared backdoor reduces the chance to be attributed.
Which threat actors are using ShadowPad?
Which threat actors are using ShadowPad?

- WINNTI/APT41
- Operation Redbonus
- Fishmonger
- Tick and Tonto Team
- Operation Redkanku
- Tropic Trooper
- LuckyMouse
WINNTI/APT41

Two sub-groups of WINNTI/APT41

- BARIUM (Tan Dailin aka Rose and Zhang Haoran)
  - Against the gaming industry and several supply chain attacks, e.g., NetSarang, ASUS, and allegedly, CCleaner

- LEAD (Chengdu 404 Network Technology Co., Ltd)
  - Attack for financial and espionage purposes
What is NCPH

A hacking group that developed lots of tools and freely shared on NCPH websites

- They also declared that the source code was on sale
- Rose and whg have collaboration on malware development since 2006

Rose likely had high privilege access – or was a co-developer – to ShadowPad
Tick and Tonto Team

- Two groups amalgamated into a new institution during the reorganization of PLA
- Started to use ShadowPad as their primary backdoor for conducting intrusion
- TICK
  - Sent spear phishing emails to deliver ShadowPad
- Tonto Team
  - Exploited CVE-2019-9489 and CVE-2020-8468 in Trend Micro’s security solutions to deliver ShadowPad
Customized tools for intrusion

Customized tools

- Modified mimikatz
- Screen capture tool
- Packet transmission tool
- Tool to list the software installed on a computer
  - ShadowPad has a plugin with the same functionality
- VBScript command executor tool
  - Generate a payload of VBScript
  - Bypass TrendMicro products
Operation Redbonus

- Against Indian country
- Other backdoors in use, such as Whitebird, IceFog and a customized instance of PCShare
Operation Redkanku

- All of the C&C servers had a self-signed certificate
- Some related samples were documented to be a part of the ProxyLogon attacks
Fishmonger

- New version of ShadowPad which had updates and more advanced obfuscation techniques
- They are interested in COVID-19 research in Hong Kong, Taiwan, India and the US.
- ShadowPad and Spyder as their primary backdoors for long-term monitoring
- A self-signed certificate is installed on several C&C servers of ShadowPad and Spyder
Landscape shift
From developing backdoors to acquiring backdoors

- **Past**
  - Chinese threat actors develop their own tool sets based on their needs during operations

- **Now**
  - The popularity of malware such as ShadowPad and CobaltStrike among Chinese espionage groups
  - Here is an example about Tick's timeline of backdoor
The benefit of acquiring backdoors

- Lower the cost
  - Reduces the cost of a well-designed piece of malware
  - Reduces the human resource to develop the malware in-house

- Keeps enhancing the stability and usability
  - The service provider will provide newest version of the backdoor with new features added
  - Unlike much of the commodity malware found in cybercriminal circles or underground forums
Mitigation advice
Mitigation advice for ShadowPad

- Audit the services and the registries to find any suspicious items
- Monitor dynamic behaviors of “spawning a new process” and “process injection”
- Apply memory forensics periodically to identify malware which resides in-memory
- Adopt an Endpoint Detection and Response (EDR) solution across your organization
- A well consolidated monitoring capability provides visibility into cyber threats
Conclusion
Conclusion

• Why do the actors choose to use ShadowPad?
  – Experienced developers to develop something much better with active updates
  – Reducing the cost of operation and development
  – Harder for security company to do further research

• Why ShadowPad is not disappearing?
  – Still under updates with more advanced obfuscation and persistence techniques
  – A powerful backdoor with more functionalities so good for long-term espionage operations and keep stealthy under the radar

• How ShadowPad affect threat intelligence?
  – Need to develop more systematic ways for attribution
Thank You