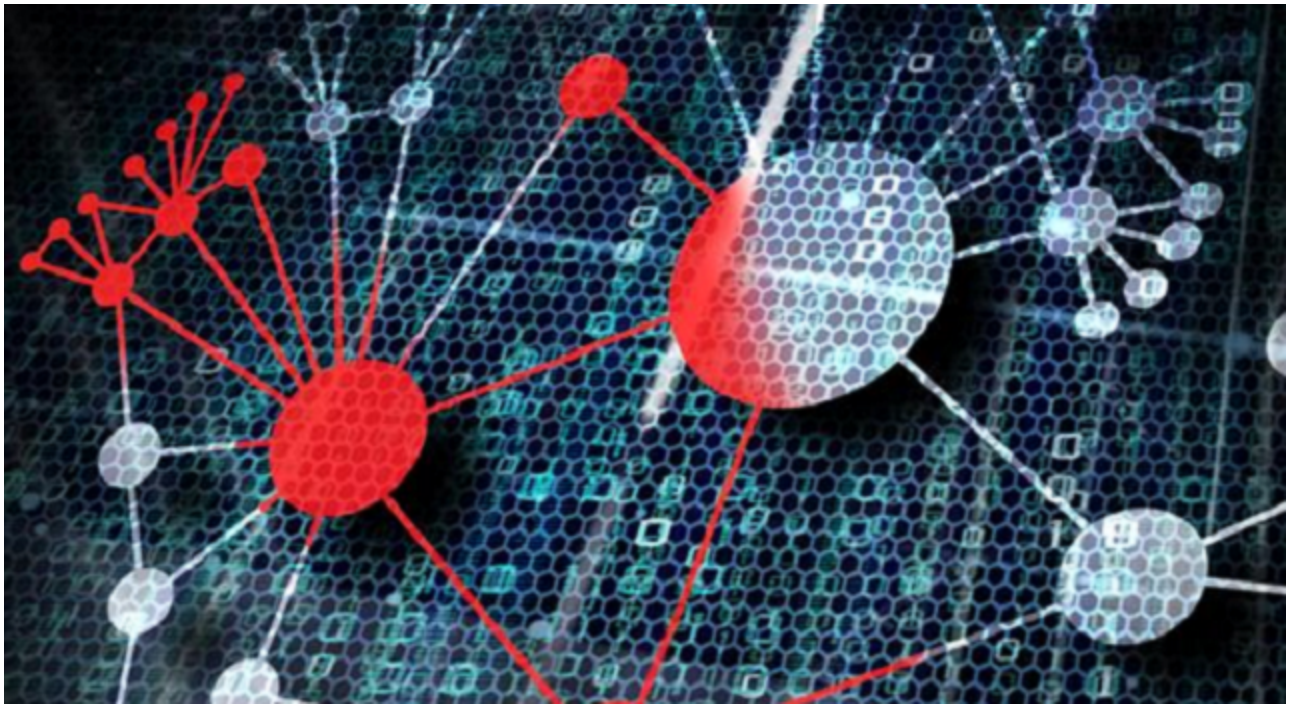
Potential MuddyWater Campaign Seen in the Middle East

blog.trendmicro.com/trendlabs-security-intelligence/campaign-possibly-connected-muddywater-surfaces-middle-east-central-asia/

March 12, 2018



We discovered a new campaign targeting organizations in Turkey, Pakistan and Tajikistan that has some similarities with an earlier campaign named MuddyWater, which hit various industries in several countries, primarily in the Middle East and Central Asia. Third party security researchers named the MuddyWater campaign as such because of the difficulties in attributing the attacks. However, given the nature of the targets, as well as the gathering and uploading of information to C&C servers, it appears that the attackers are mainly concerned with espionage activities — with the Saudi Arabia's National Cyber Security Center (NCSC) publishing an alert on their website regarding the attacks.

Given the number of similarities, we can assume that there is a connection between these new attacks and the MuddyWater campaign. It also signifies that the attackers are not merely interested in a one-off campaign, but will likely continue to perform cyberespionage activities against the targeted countries and industries.

Comparing the earlier MuddyWater campaign with this new one reveals some distinct similiarities:

	2017 MuddyWater Campaign	2018 "MuddyWater" Campaign

Countries of Targeted Organizations	Georgia, India, Iraq, Israel, Pakistan, Saudi Arabia Turkey United Arab Emirates, and the USA	Turkey, Pakistan, Tajikistan
Decoy Documents	The documents try to mimic government organizations, including the Iraqi National Intelligence Service, the National Security Agency, and the Ministry of Interior of Saudi Arabia	The documents try to mimic government organizations such as the Ministry of Internal Affairs of the Republic of Tajikistan. Some documents also come with government emblems.
Dropped Files	Visual Basic file and Powershell file; the VBS file executes the PS file	
Proxies	Hundreds of hacked websites are used as proxies.	

In addition to the common characteristics seen above, the campaigns also use similar obfuscation processes, as are the internal variables after deobfuscation. A list of isDebugEnv is also present in both campaigns.

Infection Chain

Figure 1. Infection chain for the attack

Figure 1. Infection chain for the attack

Our research found malicious delivery documents (Detected by Trend Micro as **JS_VALYRIA.DOCT and W2KM_VALYRIA.DOCT)** containing text and file names in the Tajik language attempting to target individuals working for government organizations and telecommunication companies in Tajikistan. Each document uses social engineering to trick potential victims into clicking it to enable the macros and activate the payload. While some of the payloads we observed were embedded inside the document itself, some of the payloads were also downloaded from the internet after the lure was clicked.There is a separate lure with a program key generator written in Java that was bundled with a Java downloader. However, the actual payload is the same.

Some examples of the lure documents used in the campaign can be seen below:

Figure 2. A sample document used in the campaign. Note that it uses the Tajikistan emblem, signifying that this is likely used to target government organizations or make it seem that it came from one

Figure 2. A sample document used in the campaign. Note that it uses the Tajikistan emblem, signifying that this is likely used to target government organizations or make it seem that it came from one

Figure 3. A second lure document that we found being used in the campaign designed to look like a document sent to telecommunication companies regarding dissatisfaction with their service; it also asks them to fill out a form, which can be seen in the table at the bottom

Figure 4. Another example of a header allegedly from the Ministry of Internal Affairs of Tajikistan

Figure 4. Another example of a header allegedly from the Ministry of Internal Affairs of Tajikistan

After enabling the macros and the payload executes, two files – an obfuscated Visual Basic script (Detected by Trend Micro as **VBS_VALYRIA.DOCT)**, and an obfuscated PowerShell script (Detected by Trend Mico as **TROJ_VALYRIA.PS)** — are created in the *ProgramData* directory placed in randomly-named directories. The purpose of the .VBS script is to execute the PowerShell script. The path to the VBS script is added to the task scheduler as a form of persistence.

Figure 5. The installed backdoor and persistence script

Figure 5. The installed backdoor and persistence script

In other campaigns, two files are also dropped. One of them is the VBS script, however, the second file is a base64 encoded text file, which, after decoding, results in the Powershell file, as in the previous campaign. This is one simple layer of obfuscation, likely to avoid some antivirus detections.

The latest change, drops three files – an.sct scriptlet file, an.inf file and a base64 encoded data file. The scriptlet file and inf file use publicly available code for bypassing applockerCode examples are also available on github.

The PowerShell script, which employs several layers of obfuscation, is divided into three parts. Part one contains global variables like paths, encryption keys, a list of a few hundred gates or hacked websites which serve as proxies:

Figure 6. The configuration portion of the PowerShell script

Figure 6. The configuration portion of the PowerShell script

The second part contains functions related to the encryption, which is a standard RSA encryption with very small keys.

The third part contains the backdoor function. This function will first collect machine information and take screenshots before it sends this data to a command-and-control (C&C) server while waiting for commands. These include the following actions: clean, reboot, shutdown, screenshot, and upload.

The *clean* command attempts to recursively delete all the items from drives C, D, E, and F.


Figure 7. The clean command wipes drives C, D, E and F

Figure 7. The clean command wipes drives C, D, E and F

C&C Communication

The communication is done via XML messages with the following supported ACTION commands:

- REGISTER
- IMAGE
- COMMAND RESULT
- UPLOAD

The backdoor first finds out the machine IP address by querying the internet service api[.]ipify[.]org, which returns the IP address of the currently infected machine. This IP address is then fed to another internet service called apinotes[.]com, which returns the location information of the given IP address.

The backdoor then collects the system information about the infected machine such as the Operating System name, architecture, domain, network adapter configuration, and username. It then separates each piece of information with **, and sends this system info as part of the REGISTER message:


Figure 8. The register message before encryption

Figure 8. The register message before encryption

A simple RSA algorithm with very small keys encrypts the message seen above. Let's take the first character as an example. Character "{" = 0x7B =123. Variable ${prIVATE} = 959, 713 from section 1 of the PowerShell script has two values; the first number is the key and the second number is the modulus. By computing (123 ^ 959) mod 713 = 340 we get the encrypted value of the first character (see number 340 in the figure below). The message above gets encrypted as shown in figure 9 below, then its contents are sent via post request to one of many hacked gates.

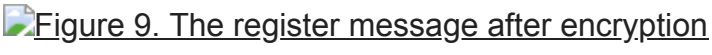


Figure 9. The register message after encryption

Figure 9. The register message after encryption

The response to this message is another set of decimal numbers which can be decrypted by the public key, which is stored in ${pUbLIC} = 37, 437 variable in part 1 of the PowerShell script.


Figure 10. The encrypted response to the register message

Figure 10. The encrypted response to the register message

The message above can be decrypted to:

{"STATUS": "OK", "TOKEN": "d02153ffaf8137b1fa3bb852a27a12f8"}

The XML message containing screenshot can be seen below. Note that the previously obtained SYSID that serves as a machine identifier, ACTION:”IMAGE” tells us that a base64 encoded image will be followed in IMAGE field.


Figure 11. The XML message with the screenshot

Figure 11. The XML message with the screenshot

It seems that the attackers are actively monitoring the incoming connections to the C&C. In one of our attempts, we sent an improper request to the C&C server, which replied with the following message: “Stop!!! I Kill You Researcher.” This level of personalized messaging implies that the attackers are monitoring what data is going to and from their C&C server.


Figure 12. When the threat actor discovers the researcher via an improper request

Figure 12. When the threat actor discovers the researcher via an improper request

Another hidden message or a false flag?

For the PowerShell script, the first part contains a variable named dragon_middle, which is an array containing a few hundred URLs ending with *connection.php* that serve as proxies between victim and C&C. If communication with C&C fails, and if the PowerShell script is run from a command line, a few error messages written in simplified Mandarin Chinese are displayed, with a curious phrase that translates to "waiting for dragon":

- 无法访问本地计算机寄存器 (Unable to access local computer register)
- 任务计划程序访问被拒绝 (Mission Scheduler access is denied)
- 无法连接到网址，请等待龙 (Cannot connect to URL, please wait for dragon)
- 无法连接到网址，请等待龙 (Cannot connect to website, please wait for dragon)

These messages may not reveal anything about the real attackers as the malware writers sometimes like to embed false flags into their programs to confuse researchers. The syntax and grammar suggest that the language could have been machine-translated rather than written by a native speaker.

Countermeasures and Trend Micro Solutions

Users unfamiliar with the various kinds of social engineering techniques might find it difficult to distinguish a legitimate message from a malicious one – thus the need for education on identifying and mitigating phishing attacks – especially if it involves organizations in sensitive industries such as government and manufacturing. Context, in this case, is important. Users need to consider *why* they received an email and avoid clicking on any links or attachments in general until they are certain that they are legitimate.

Trend Micro™ Deep Discovery™ provides detection, in-depth analysis, and proactive response to today's stealthy malware, and targeted attacks in real time. It provides a comprehensive defense tailored to protect organizations against targeted attacks and advanced threats through specialized engines, custom sandboxing, and seamless correlation across the entire attack lifecycle, allowing it to detect threats even without any engine or pattern update.

Malware such as the one analyzed in this entry also use email as an entry point, which is why it's important to secure the email gateway. Trend Micro™ Email Security is a no-maintenance cloud solution that delivers continuously updated protection to stop spam, malware, spear phishing, ransomware, and advanced targeted attacks before they reach the network. Trend Micro™ Deep Discovery™ Inspector and InterScan™ Web Security prevent malware from ever reaching end users. At the endpoint level, Trend Micro™ Smart Protection Suites deliver several capabilities that minimize the impact of these attacks.

These solutions are powered by the Trend Micro XGen™ security, which provides a cross-generational blend of threat defense techniques against a full range of threats for data centers, cloud environments, networks, and endpoints. It features high-fidelity machine learning to secure the gateway and endpoint data and applications, and protects physical, virtual, and cloud workloads.

Indicators of Compromise (IOCs)

Hashes detected as W2KM_VALYRIA.DOCT:

- 009cc0f34f60467552ef79c3892c501043c972be55fe936efb30584975d45ec0
- 153117aa54492ca955b540ac0a8c21c1be98e9f7dd8636a36d73581ec1ddcf58
- 18479a93fc2d5acd7d71d596f27a5834b2b236b44219bb08f6ca06cf760b74f6
- 18cf5795c2208d330bd297c18445a9e25238dd7f28a1a6ef55e2a9239f5748cd
- 1ee9649a2f9b2c8e0df318519e2f8b4641fd790a118445d7a0c0b3c02b1ba942
- 2727bf97d7e2a5e7e5e41ccbfd7237c59023d70914834400da1d762d96424fde
- 2cea0b740f338c513a6390e7951ff3371f44c7c928abf14675b49358a03a5d13
- 3b1d8dcbc8072b1ec10f5300c3ea9bb20db71bd8fa443d97332790b74584a115
- 3d96811de7419a8c090a671d001a85f2b1875243e5b38e6f927d9877d0ff9b0c
- 3da24cd3af9a383b731ce178b03c68a813ab30f4c7c8dfbc823a32816b9406fb

- 6edc067fc2301d7a972a654b3a07398d9c8cbe7bb38d1165b80ba4a13805e5ac
- 76e9988dad0278998861717c774227bf94112db548946ef617bfaa262cb5e338
- 9038ba1b7991ff38b802f28c0e006d12d466a8e374d2f2a83a039aabcbe76f5c
- 93745a6605a77f149471b41bd9027390c91373558f62058a7333eb72a26faf84
- a70aca719b06fc8ef0cd0b0e010c7bc8dc6d632e4f2f874e4c0e553bd8db2df2
- aa60c1fae6a0ef3b9863f710e46f0a7407cf0feffa240b9a4661a4e8884ac627
- af5f102f0597db9f5e98068724e31d68b8f7c23baeea536790c50db587421102
- cee801b7a901eb69cd166325ed3770daffcd9edd8113a961a94c8b9ddf318c88
- d07d4e71927cab4f251bcc216f560674c5fb783add9c9f956d3fc457153be025
- dfbd67177af9d35188fc9ff9363c2b9017e9ccfe6719e3d641a56fb5dc0d47f7
- eff78c23790ee834f773569b52cddb01dc3c4dd9660f5a476af044ef6fe73894
- fbbda9d8d9bcaaf9a7af84d08af3f5140f5f75778461e48253dc761cc9dc027c

Hash detected as VBS_VALYRIA.DOCT:

- 0A9FC303CA03F4D9988A366CBBD96C24857E87374568EC5A4AAA4E55FE2C3C7E
- 0BC10D5396B3D8ECC54D806C59177B74E167D9F39D8F1B836806127AF36A7C4E
- 0BC10D5396B3D8ECC54D806C59177B74E167D9F39D8F1B836806127AF36A7C4E
- 25186621282D1E1BAD649B053BDB7B56E48B38189F80DB5A69B92301EF9ED613
- 25186621282D1E1BAD649B053BDB7B56E48B38189F80DB5A69B92301EF9ED613
- 3607432758176a2c41a1971b3c4d14a992a68b231851f8b81c6e816ea9ea29b2
- 59F9E0FAA73E93537AE4BD3A8695874BA25B66CEFA017537132914C770D0CF70
- 59F9E0FAA73E93537AE4BD3A8695874BA25B66CEFA017537132914C770D0CF70
- 59F9E0FAA73E93537AE4BD3A8695874BA25B66CEFA017537132914C770D0CF70
- 6228d79f56c574ceada16453404c54dd95641aa78d3faed6874daf485116793b
- 66af894eee6daae66bf0bcb87cb7abe2a0ebb6a59779f652db571e7ee298d751
- 92C7FEAD5EE0F0ECD35FE247DBE85648AADA4B96F1E960B527B4929E42D47B01
- c006911be5480f09e0d8560c167561f68681607ca8f7e3c4f5d476dc6673594f
- F05C18C1D4428349137A9DF60CDEBE8A0F9E6DA47B359DC0616FF8D47E46704E

Hash detected as TROJ_VALYRIA.PS:

- 0065d592d739ac1dd04d0335151c8855c7fafbf03e86134510ac2fc6766e8d60
- 0073ce0f4c82fc4d0470868e124aab9ad08852e1712564136186e5019fca0da0
- 02F58256FF52ED1CDB21064A28D6E5320005F02EF16E8B2FE851438BBC62A102
- 02F58256FF52ED1CDB21064A28D6E5320005F02EF16E8B2FE851438BBC62A102
- 04d61b1d2c3187280b3c4e93d064a051e9ee0f515f74c6c1c44ba577a7a1c804
- 04d61b1d2c3187280b3c4e93d064a051e9ee0f515f74c6c1c44ba577a7a1c804
- 0A9FC303CA03F4D9988A366CBBD96C24857E87374568EC5A4AAA4E55FE2C3C7E
- 0A9FC303CA03F4D9988A366CBBD96C24857E87374568EC5A4AAA4E55FE2C3C7E
- 4DD5C3CE5ED2145D5AFA8DD476A83DFC693E5FC7216C1EABB3FA0EB6B5F8590D
- 4DD5C3CE5ED2145D5AFA8DD476A83DFC693E5FC7216C1EABB3FA0EB6B5F8590D
- 55ae821cf112ff8d6185ce021f777f73d85150c62a835bb1c02fe9e7b3f863bf
- 61d846708f50024e1c65237eb7158beac9b9c5840853b03ef7c73fe5293a9a8d

- 624762a90b7272e247e5022576b7912d1aa0b32bc13aabc7ee47197e5b87a41b
- 6421C22D854C199B761436C87CAE1EAFFBA8783A3A40C00D4A0982D7C242EA79
- 92C7FEAD5EE0F0ECD35FE247DBE85648AADA4B96F1E960B527B4929E42D47B01
- a53f832edc18de51e0ffaf67047072a6bbd5237defa74f5bf35dfc0df2aeca1b
- C1780F3AD76AF703CEDDD932B187CF919866A00BB3E2D6F0827B9DAE9D8875B6
- C1780F3AD76AF703CEDDD932B187CF919866A00BB3E2D6F0827B9DAE9D8875B6
- C9D782FFAA98791613FEF828E558B296932FA245192BD0EBA8F76536860DB84E
- C9D782FFAA98791613FEF828E558B296932FA245192BD0EBA8F76536860DB84E
- CCA8E84901C4184BE2849D29C39294FD4B6940F9A6668FDCFF9728CD319FFF96
- CCA8E84901C4184BE2849D29C39294FD4B6940F9A6668FDCFF9728CD319FFF96
- cca8e84901c4184be2849d29c39294fd4b6940f9a6668fdcff9728cd319fff96
- e57dbce8130e281a73727122d33cbff170a54237cd0016d79b30ace18c94e7d4

Hash detected as JS_VALYRIA.DOCT:

070EBCAC92FB7619F957BF3F362099574158E5D2D0BC0CF9206A31BA55EDD48F

Scriptlets and inf files related to applocker bypass:

- 2791fdc54ee037589f951c718935397e43d5f3d5f8e078e8b1e81165a3aebbaf
- 288afbe21d69e79a1cff44e2db7f491af10381bcc54436a8f900bcbd2a752a6f
- 5e173fbdcd672dade12a87eff0baf79ec4e80533e2b5f6cf1fac19ad847acba0