

Fidelis Threat Advisory #1016

Pushdo It To Me One More Time

Apr 16, 2015

Document Status: FINAL
Last Revised: 2015-04-15**Executive Summary**

Once thought to be defunct, the resilient Pushdo has surfaced with infections observed in more than 50 countries, with a substantial infection rate located in the Asia-Pacific region. Based on data aggregated from a controlled sinkhole, Fidelis Cybersecurity has observed some notable changes with the primary command and control (C&C) and conducted in-depth analysis of the secondary C&C Domain Generation Algorithm (DGA). In order to support network defenders, Fidelis Cybersecurity is offering a new, free data feed of verified indicators to support the detection and mitigation of Pushdo. **Our intention behind revealing these details is to enable widespread detection and remediation of this threat as well as to force a comprehensive retooling exercise on the operators of the Pushdo botnet.**

Key Findings:

- Pushdo continues to affect large numbers of users worldwide with higher concentrations observed in countries with high rates of pirated software usage such as India, Indonesia, Turkey and Vietnam.
- Pushdo has evolved its Command-and-Control techniques beyond what has previously been published in the research community. The DGA component of this infrastructure uses an elaborate algorithm and has moved entirely to domains registered in Kazakhstan (.kz).
- Pushdo continues to deliver a variety of secondary payloads such as Cutwail, Dyre, Fareit, and Zeus. These malware families are known to be used for DDoS and credential theft purposes.

Recommended Actions:

- Network defenders should operationalize the indicators that accompany this report, including the data feed containing domains based on our having reversed the DGA.
- Defenders can also utilize the Yara rule for Fareit detection that is provided in this report.

Threat Overview

History

First detected in 2007, Pushdo was once considered the second largest spam botnet in existence, sending 7.7 billion emails per day at its peak. Since then, it has proven resilient, enduring several take down attempts by law enforcement^{9,12,13} with new binaries still appearing regularly. Pushdo's modularity allows it be used to deploy a variety of malware, and it has been affiliated with secondary payloads including Cutwail, Dyre, Fareit, and Zeus.^{14,15}

Sinkholed Data – A Look Into Active Infection

Using the DGA process, we inserted the domains into a controlled sinkhole. Our findings concluded that the majority of the top five infected countries were located primarily in the Asia-Pacific region. While there is not enough information to formulate a judgment, initial speculation as to the reasons why these countries registered high infection rates is due to notoriously high rates of pirated software use recorded in each of the top five countries

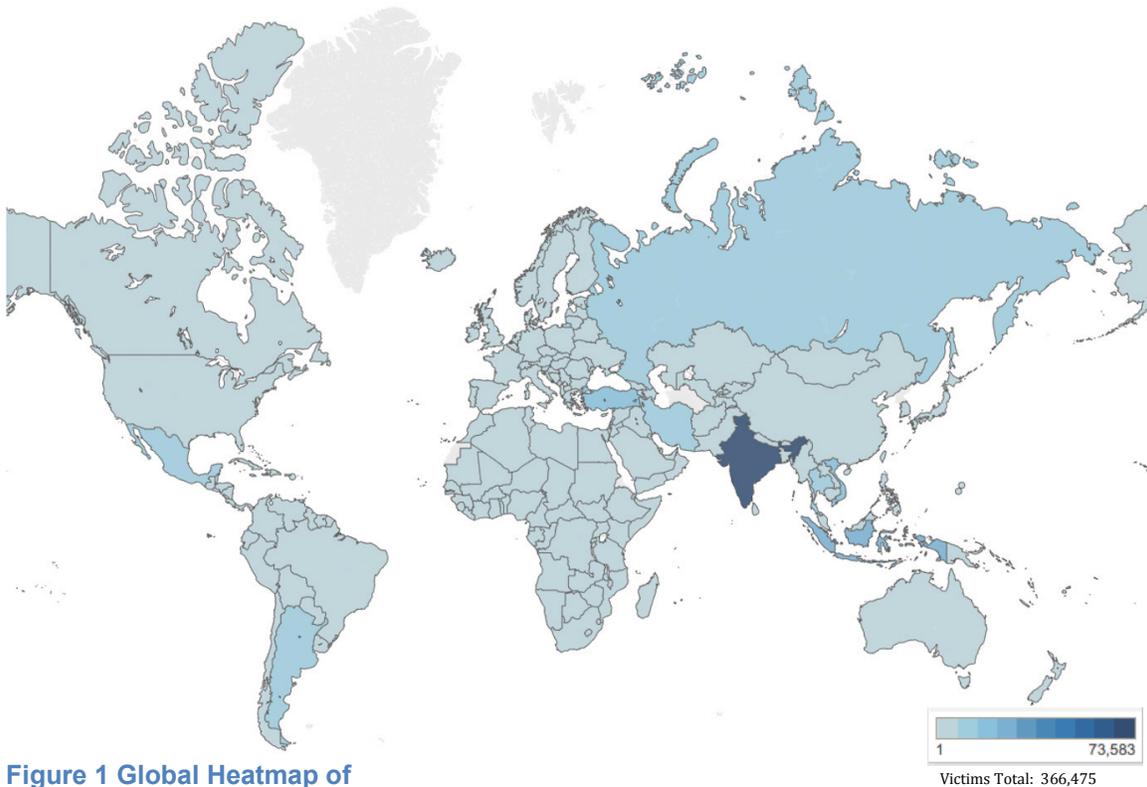


Figure 1 Global Heatmap of infections from Sinkhole

North America	South America	Europe	Asia	Africa	Australia
MX(9575)	AR(9721)	RU(9522)	IN(73583)	DZ(4202)	AU(1207)
US(6624)	PE(4716)	IT(5830)	ID(29149)	ZA(4156)	NZ(605)
SA(4119)	BR(4047)	PL(5688)	TR(21040)	EG(3658)	FJ(117)
CA(949)	CL(3169)	UA(4597)	VN(20899)	TN(2616)	VU(24)
GT(830)	CO(2467)	ES(4408)	TH(13925)	MA(2046)	NC(24)
DO(726)	EC(1859)	RO(3269)	IR(11529)	NG(1047)	TO(20)
CR(315)	VE(1484)	FR(2346)	MY(7046)	MU(911)	WS(18)
CU(224)	UY(1061)	DE(2338)	PK(6809)	SN(646)	PF(18)
PA(170)	BO(661)	GR(2216)	TW(6335)	SD(507)	PG(15)
SV(159)	PY(204)	GB(2172)	PH(4427)	CM(456)	GU(12)
Rest(751)	Rest(51)	Rest(14088)	Rest(35486)	Rest(3577)	Rest(9)

According to a 2013 survey by the BSA software alliance,¹⁹ the leading advocate for the global software industry before governments and in the international marketplace, as of 2013, the unlicensed software installation of the countries were as follows:

Indonesia – 84%

Vietnam – 81%

Thailand – 71%

India – 60%

Turkey – 60%

The high-rate of piracy coupled with the fact that Pushdo's preferred infection vector is via email may account for the high infection rates in the top five. Notably, China, a country with a 74% unlicensed software installation figure according to BSA did not register significant infection.

Threat Detail

Pushdo continues to rely on two methods to establish C&C: a primary technique utilizing seemingly legitimate sites and a DGA fallback technique. While these same basic techniques are the same as those previously discussed¹⁶, implementation details continue to evolve.

Primary C&C

Pushdo's primary C&C is difficult to identify due to the large number of POSTs generated upon execution. This functionality has not changed over the years. Though, currently active C&C responses no longer identify the embedded C2 as a jpeg. Instead, the response simply contains "<!—", the C2 marker sought by the malware followed by encoded and encrypted data.

```
</a>
</center>
<! -Ksf1hBOJoTzsFWAt2nsbEFd5GxC8E\4M+Ni XfgX7kPsLBzMMU4/rUEGE
</body>
<img src=blank.gif height=135>
```

Another previously discovered Pushdo characteristic was that the first four bytes of client POST data were a multiple of 0x1ecb¹⁶. This was offered as a way to identify communication to real C&C. In newer samples, the client POST value is base64 encoded preventing the use of that characteristic to identify true C&C traffic. The malware validates the server response by performing a series of calculations on the data, and then finally dividing four bytes of the result by 0x1ecb. The remainder must equal 0xA to be considered valid. Once the response is validated, the client decrypts and executes the payload(s).

Currently active primary C&C domain:

frimeset[.]com – 207.182.143[.]58

As of this writing, the payloads delivered from the above C&C IP are Fareit (de3b206a8066db48e9d7b0a42d50c5cd) and Cutwail (be284327e1c97be35d9439383878e29d).

Fallback C&C / DGA

DGAs in malware provide a powerful layer of infrastructure protection for crimeware. While the concept of creating a large number of domain names as rendezvous points is not complicated, the method used to create the seemingly arbitrary domain strings is often hidden behind varying layers of obfuscation (e.g., encryption, function renaming, custom packing, difficult-to-follow mathematical computations, or various other anti-analysis techniques). These protections must each be defeated to gain access to the DGA code, and then additional effort must be put forth to understand the functions that compose the exposed DGA. After achieving an understanding of the algorithm, it becomes a simple matter to replicate the functionality of the DGA within a script. This allows for the automation and systemic deployment of the generated domain indicators. It also means users of the script can predict upcoming domains and proactively profile them while monitoring for shifts that may indicate a change.

Writing a DGA requires a strong level of sophisticated programming skills, which is a reason why

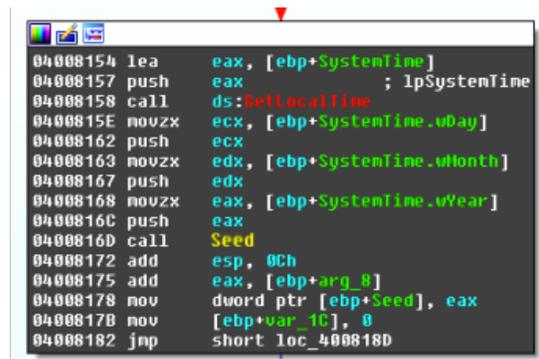
algorithms are often reused across versions. Reuse without adaption would result in the generated domains being the same across versions and make tracking and defeating the DGA trivial. To prevent this situation, simple-to-update static seed values/strings/indexes are incorporated to serve as input to the DGA; changing these input values, in turn, changes the output of the DGA.

Pushdo DGA Analysis

The Pushdo downloader bot has proven remarkably resilient to law enforcement efforts to take down the botnet on several occasions in the past. Since DGAs provide so much protection from detection, we will focus on analyzing the DGA itself. To begin, the functions responsible for manufacturing the domain strings must be identified. Isolating these functions allows the analyst to limit the scope of the investigation. Four notable components of the Pushdo DGA are system time, seed, MD5 and the domain name generator.

I. System Time

The first domain generated on each day is seeded from the system's current datetime value. Thus, a call to the GetLocalTime function serves to provide the necessary data. The day, month, and year are pushed onto the stack in hex format to be used by the seed function that follows.



```

04000154 lea    eax, [ebp+SystemTime]
04000157 push  eax           ; lpSystemTime
04000158 call  ds:GetLocalTime
0400015E movzx  ecx, [ebp+SystemTime.uDay]
04000162 push  ecx
04000163 movzx  edx, [ebp+SystemTime.uMonth]
04000167 push  edx
04000168 movzx  eax, [ebp+SystemTime.uYear]
0400016C push  eax
0400016D call  Seed
04000172 add   esp, 0Ch
04000175 add   eax, [ebp+arg_0]
04000178 mov   dword ptr [ebp+Seed], eax
0400017B mov   [ebp+var_1C], 0
04000182 jnp   short loc_400018D
  
```

Figure 2

II. Seed

The purpose of the seed function is to calculate an integer based on the current system datetime in the format of YYYYMMDD. This function receives three parameters from the previous GetLocalTime function and performs a series of calculations on those values to generate the seed.

$$YYYY = X$$

$$MM = R$$

$$DD = D$$

First, the year value, X, is shifted right by 2 bits resulting in Y.

$$X \gg 2 = Y$$

Then, the resulting value Y is subtracted from the original year value X to produce Z.

$$X - Y = Z$$

Next, Y is multiplied by 0x16e to produce A.

$$Y \times 0x16e = A$$

Then, Z is multiplied by 0x16d to produce B.

$$Z \times 0x16d = B$$

Next, the month value R is used to index an array of hard-coded static values T to calculate a month offset.

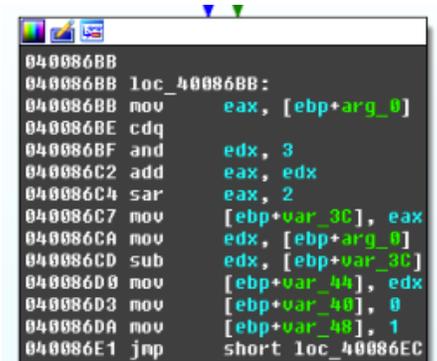
$$\sum_{n=1}^R T_n = E$$

The day value D is added to the month offset E to produce F.

$$E + D = F$$

Finally, A, B, and F are added to produce the seed.

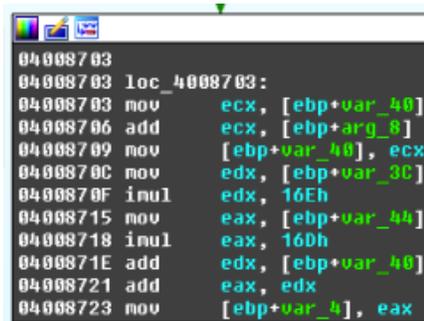
$$A + B + F = Seed$$



```

040086BB
040086BB loc_40086BB:
040086BB mov     eax, [ebp+arg_0]
040086BE cdq
040086BF and     edx, 3
040086C2 add     eax, edx
040086C4 sar     eax, 2
040086C7 mov     [ebp+var_3C], eax
040086CA mov     edx, [ebp+arg_0]
040086CD sub     edx, [ebp+var_3C]
040086D0 mov     [ebp+var_44], edx
040086D3 mov     [ebp+var_48], 0
040086D9 mov     [ebp+var_48], 1
040086E1 jnp    short loc_40086EC
  
```

Figure 3



```

04008703
04008703 loc_4008703:
04008703 mov     ecx, [ebp+var_40]
04008706 add     ecx, [ebp+arg_8]
04008709 mov     [ebp+var_40], ecx
0400870C mov     edx, [ebp+var_3C]
0400870F imul  edx, 16Eh
04008715 mov     eax, [ebp+var_44]
04008718 imul  eax, 16Dh
0400871E add     edx, [ebp+var_40]
04008721 add     eax, edx
04008723 mov     [ebp+var_4], eax
  
```

Figure 4

Example:

- (Main) Today's date: 20150326
- (Seed) Seed creation function input: 20150326
- (Seed) Year(2015) value = X, X = 0x7df
- (Seed) X >> 2 = Y, Y = 0x1f7
- (Seed) X - Y = Z, Z = 0x5e8

(Seed) $Y * 0x16e = A$, $A = 0x2cf22$
 (Seed) $Z * 0x16d = B$, $B = 0x86bc8$
 (Seed) Day(26) value = D, $D = 0x1a$
 (Seed) Month offset = E, $E = 0x3b$
 (Seed) $D + E = F$, $F = 0x55$
 (Seed) $A + B + F = \text{result}$, $\text{result} = 0xb3b3f$
 (Main) Seed function return value: $0xb3b3f$

III. MD5

An MD5 value is used as input to the domain name generation function. Two possible logical paths for MD5 creation exist. One path is taken the first time a domain for a particular date is generated. The second path is taken for the second and subsequent domains for that date.

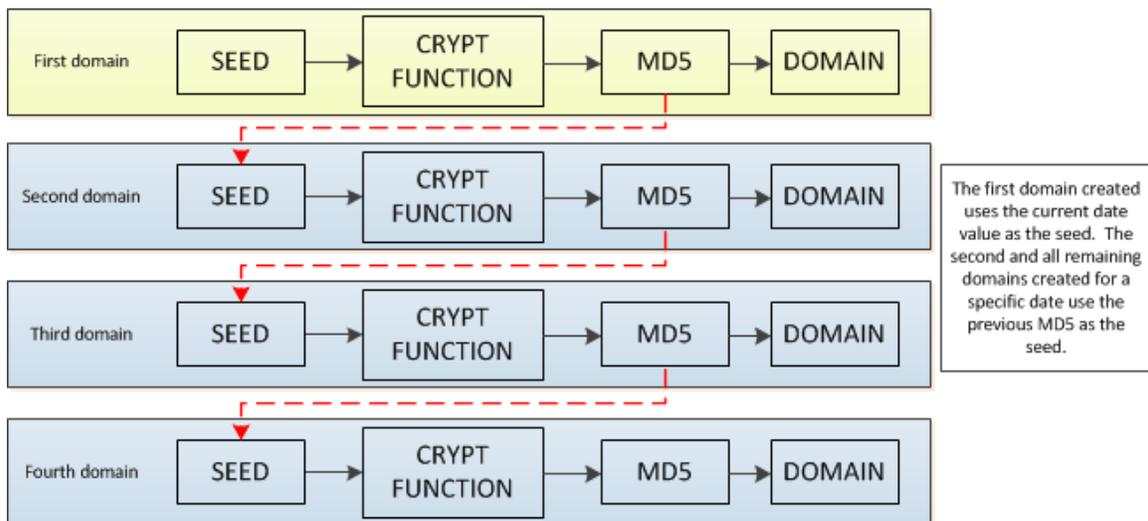


Figure 5

First Domain

The MD5 function calculates an MD5 based on the date-based seed. The MD5 is created using calls to CryptCreateHash and CryptGetHashParam. The output value will serve as the seed for the domain name generator function.

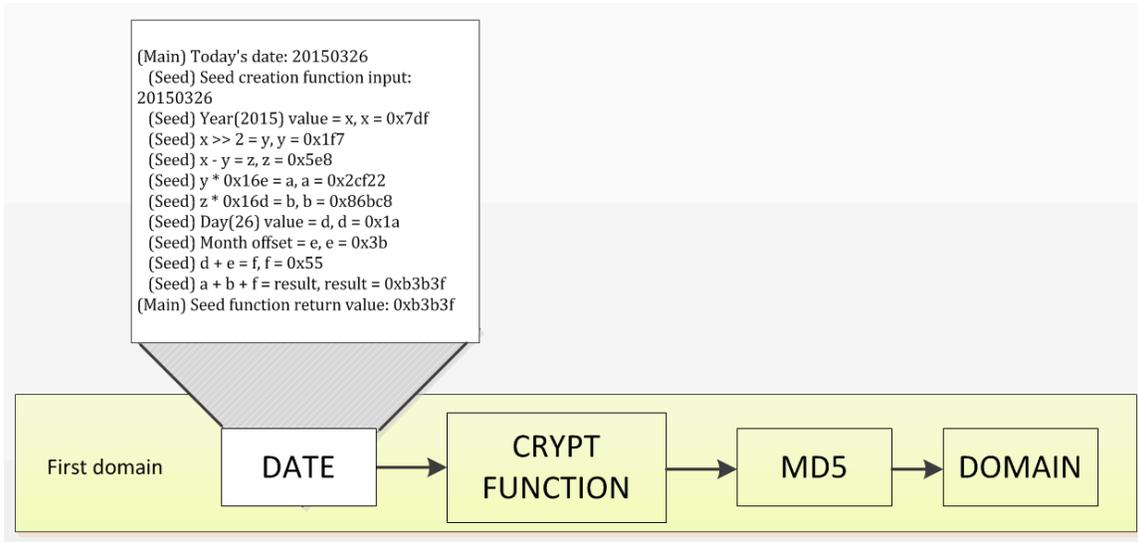


Figure 6

Remaining Domains

If an MD5 value already exists, i.e., the function has run previously, the MD5 function uses the first 4 bytes from the previous run's MD5 value to seed the current run. This happens 29 times (1 date-seeded + 29 domain-seeded = 30 total domains per day).

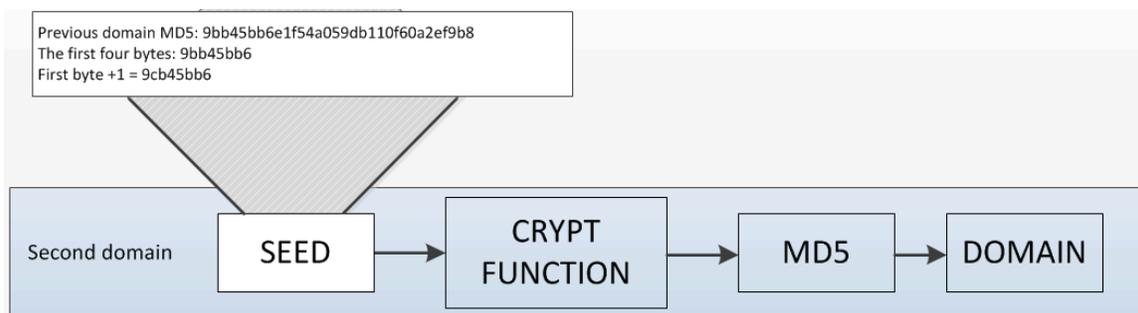


Figure 7

The concept of the subsequent domains relying on the first seed serves to protect the integrity of the DGA by causing a cascading failure if one generated string within the series is incorrect. The

vowel loop conditional logic within the domain name generator further strengthens the protection by making it very easy to create an incorrect domain string.

IV. Domain Name Generator

Contained within the binary are *four* unique, static strings of either consonants or vowels (keys). Each key is used uniquely within one of the four possible paths noted below. The keys are also different among the variants of the Pushdo samples analyzed. The domain name generation function uses the seed value passed from the MD5 function to calculate an index value. The index value is used to identify a specific character within one of the keys. Each identified character is added to the domain string being generated.

For each cycle, a byte from the MD5 is evaluated starting with the leftmost byte and moving sequentially right thereafter. At least two letters are added to the accumulating domain string per byte. Up to three total letters may be accumulated depending on the conditional path taken.

There are two distinct sets of characters within the four index strings: consonants sets and vowel sets. Each is used differently to accumulate a domain name. All possible paths are discussed below.

First Letter

The first key used contains only consonants. Thus, all output domains will begin with a consonant. The index value is calculated by dividing the current byte of the MD5 G by 19. The remainder is the index value.

$$\text{\$first letter index} = G \text{ \textbackslash mod } 19\text{\$}$$

Second Letter

The second key used contains only vowels. Therefore, the second letter will always be a vowel. The index value is calculated by adding 1 to the current MD5 byte and then dividing the result by 5. The remainder is the index value.

$$\text{\$second letter index} = (G + 1) \text{ \textbackslash mod } 5\text{\$}$$

Example:

MD5: 9bb45bb6e1f54a059db110f60a2ef9b8

Byte used: 0x9b

Consonant index value: 3

Consonant indexed letter: h

Vowel index value: 1

Vowel indexed letter: u

hufuqzyilaru[.]kz

Conditionals - Path A

The third key used contains only vowels. The first condition determines if a third addition to the string will be added using the current byte. Two evaluations occur to satisfy this condition and generate a new index and domain letter. The first evaluation checks to see if a specific, static letter was previously added. The sought out letter differs across multiple variants of Pushdo. In the case of a match, it next checks if the current byte added to 2 and logically ANDed with 8 equals 0. If the condition is false, a third index will be generated by adding 2 to the byte and dividing by 3. The remainder is the index value.

$$\text{\$third letter index}_A = (G + 2) \backslash \text{mod } 3\text{\$}$$

Example:

MD5: 8888584c6e11cad1d90f18310e8b77c7

Byte used: 0x88

Consonant index value: 3

Consonant indexed letter: h

Vowel index value: 2

Vowel indexed letter: y

Path A index: 0

Path A indexed letter: i

hyihyime[.]kz

We can see within the example above that the conditions for path A were met for byte 0x88 and a third letter was added. At this point the loop will break and a new byte will be evaluated.

Conditionals - Path B

The forth key used contains only consonants. Path B will be taken only if the conditions required for path A above were *not* met. In this case, the currently evaluated byte will be added to 2 and logically ANDed with 2. If the result does not equal 0, the condition will be met and a third index will be generated. The index is calculated by adding 3 to the byte and dividing by 15. The remainder is the index value.

$$\text{\$third letter index}_B = (G + 3) \backslash \text{mod } 15\text{\$}$$

Example:

MD5: b94bd85cb360c1565f037b7613cf0073

Byte used: 0xb9

Consonant index value: 14

Consonant indexed letter: x

Vowel index value: 1

Vowel indexed letter: u

Path B index value: 8

Path B indexed letter: n

xunsudyara[.]kz

Within the example above, the conditions for path B were met for byte 0xb9 and a third letter was added. The loop will now break and a new byte will be evaluated.

String Length

The length of each domain is calculated by dividing the first four bytes of the MD5 by 4. The remainder is then added to 9 (9-12 characters in length) to achieve an integer string length. Once the accumulated string meets the requirement, the loop breaks, all characters beyond the determined length are removed, and the typical .kz TLD is added creating a functional domain name. That domain name is then written to memory and the process repeats.

Example:

MD5: 8888584c6e11cad1d90f18310e8b77c7

Length: 10 characters $((0x4c588888 \%4)+9 = 9)$

hyihyimel[.]kz

Other Dates

After generating 30 domains for the current day, the malware generates values for the preceding day. 30 domains are generated for the preceding day and so on until 30 days worth of domains have been created. Next the algorithm begins to calculate the domains for future dates up to 15 total.

Network Resolution

On May 15, 2013, Damballa posted a blog¹⁷ update where the DGA moved from using the tld “com” to “kz”, otherwise known as the country code of Kazakhstan. Adhering to the behavioral analysis of the DGA, we have generated over 46,000 unique domains from the observation date to the current date. Using a combination of passive DNS and WHOIS data, we observed DNS entries for more than 200 of these generated domains.

AS	IP	AS Name	CC	Domain Count
24940	144.76.86.115	HETZNER-AS Hetzner Online AG	DE	2
33070	166.78.144.80	RMH-14 - Rackspace Hosting	US	9
16125	185.8.107.114	BALTICSERVERS1-AS UAB DUOMENU CENTRAS	LT	1
16276	188.165.228.199	OVH OVH SAS	FR	6
	192.168.1.1	Reserved		117
59564	195.211.153.33	UNIT-IS-AS Unit-IS Ltd.	UA	1
8426	195.22.26.231	CLARANET-AS ClaraNET LTD	GB	1
8426	195.22.26.252	CLARANET-AS ClaraNET LTD	GB	1
8426	195.22.26.253	CLARANET-AS ClaraNET LTD	GB	1
8426	195.22.26.254	CLARANET-AS ClaraNET LTD	GB	2
50482	212.154.192.98	KAZAKHTELECOM-AS JSC Kazakhtelecom	KZ	5
16276	37.59.37.160	OVH OVH SAS	FR	4
16276	46.105.173.196	OVH OVH SAS	FR	2
24940	5.9.61.148	HETZNER-AS Hetzner Online AG	DE	9
30058	50.7.210.226	FDCSERVERS - FDCservers.net	US	1
16509	54.186.239.165	AMAZON-02 - Amazon.com, Inc.	US	3

10439	66.240.194.139	CARINET - CariNet, Inc.	US	1
19969	69.195.129.70	JOESDATACENTER - Joe_s Datacenter, LLC	US	1
8708	86.124.164.25	RCS-RDS RCS & RDS SA	RO	6
16276	91.121.220.199	OVH OVH SAS	FR	3
16276	91.121.5.75	OVH OVH SAS	FR	1
16276	94.23.247.220	OVH OVH SAS	FR	1
8708	5.2.189.251	RCS & RDS Business	RO	21
16509	54.201.30.58	AMAZON-02 - Amazon.com, Inc.	US	2
394	188.165.158.116	OVH OVH	FR	2

Table 1: IP information and associated domain counts

Included in this list are a combination of sinkholes, domain parking, domain registrars, and other interesting observations.

One of the generated domains from the DGA script, safpobdazy[.]kz first seen in passive DNS 12/16/2014, pointed to an OVH managed IP address 188.165.158.116. Notably, the same IP also resolved to the current C2 domain, frimeset[.]com, during the same time period.

The follow are additional resolutions of frimset[.]com:

IP Address	First Seen	Last Seen	AS	AS Name	CC
192.99.161.86	8/27/14 6:54	1/13/15 8:35	16276	OVH OVH SAS	FR
5.135.76.168	9/17/14 5:04	10/31/14 6:09	16276	OVH OVH SAS	FR
178.32.164.53	10/31/14 7:51	11/4/14 9:16	16276	OVH OVH SAS	FR
178.32.0.248	11/4/14 10:23	12/16/14 2:55	16276	OVH OVH SAS	FR
188.165.158.116	12/16/14 3:04	1/13/15 7:04	16276	OVH OVH SAS	FR
5.196.97.152	1/13/15 5:43	1/17/15 4:03	16276	OVH OVH SAS	FR
37.187.74.148	1/17/15 5:46	1/29/15 2:59	16276	OVH OVH SAS	FR

207.182.143.58	1/21/15 6:38	4/13/15 3:56	10297	ENET-2 - eNET Inc.	US
188.165.2.54	1/23/15 7:14	1/26/15 1:55	16276	OVH OVH SAS	FR
64.79.90.83	1/29/15 4:03	3/26/15 3:23	10297	ENET-2 - eNET Inc.	US

Table 2: Frimeset[.]com DNS history

Indicators and Mitigation Strategies

Fidelis Cybersecurity is making available 2 files that represent all DGA generated domains for 2015^{20,21}.

The following Yara rule detects Pushdo's current Fareit sample and other similar payloads.

```

rule crime_win_PWS_Fareit
{
  meta:
    description = "Fareit password stealer"
    author = "General Dynamics Fidelis Cybersecurity Solutions - Threat Research Team"
    reference = "TBD"
    date = "20150414"
    filetype = "exe"
    hash_1 = "e93799591429756b7a5ad6e44197c020"
    hash_2 = "891823de9b05e17def459e04fb574f94"
    hash_3 = "6e54267c787fc017a2b2cc5dc5273a0a"
    hash_4 = "40165ee6b1d69c58d3c0d2f4701230fa"
    hash_5 = "de3b206a8066db48e9d7b0a42d50c5cd"
    hash_6 = "b988944f831c478f5a6d71f9e06fbc22"
    hash_7 = "7b7584d86efa2df42fe504213a3d1d2c"
    hash_8 = "f088b291af1a3710f99c33fa37f68602"
  strings:
    $mz = {4d5a}
    $s1 = "SELECT hostname, encryptedUsername, encryptedPassword FROM moz_logins"
    $s2 = "gate.php"
    $s3 = "STATUS-IMPORT-OK"
    $s4 = "Client Hash"
    $s5 = "YUIPWDFILE0YUIPKDFILE0YUICRYPTED0YUI1.0"
    $c1 = "wisefpsrvs.bin"
    $c2 = "out.bin"
  condition:
    $mz at 0 and filesize < 105KB and all of ($s*) and ($c1 or $c2)
}

```

The Fidelis Take

Pushdo continues to demonstrate its resiliency with a new variant that to date has infected systems in more than 50 countries, with a substantial concentration in the Asia-Pacific region. While there is limited information, we speculate that one reason for this is the significant amount of pirated software in the region especially by those countries on the top five list. Furthermore, the diversity of the target countries suggests that the infections were the product of opportunity rather than the intended specific targeting of any particular entity or country. Based on its history, we expect Pushdo to continue to be leveraged in support of these types of activities. Its ability to function alongside a variety of other malware that utilize dynamic DGAs to avoid detection will likely make it an attractive option for malicious actors seeking to leverage its capabilities in tandem with secondary payloads. Understanding the new features of the variant and how they work allows defenders to quickly adopt and deploy the indicators provided by the Fidelis Cybersecurity feed.

Fidelis Cybersecurity's advanced threat defense product, Fidelis XPS™, detects all of the activity documented in this paper.

References

1. "PushDo Evolves Again: Enhances Evasion with Domain Generation," Damballa, May 2013, <https://www.damballa.com/pushdo-evolves-again-enhances-evasion-with-domain-generation-algorithm-dga/>
2. "Analysis of Dyreza - Changes & Network Traffic," Stopmalvertizing.com, July 2014, <http://stopmalvertizing.com/malware-reports/analysis-of-dyreza-changes-network-traffic.html>
3. "Pushdo – SecureWorks," Secure Works, December 2007, <http://www.secureworks.com/cyber-threat-intelligence/threats/pushdo/>
4. "A New Version of the 3rd Generation of Pushdo," Fortinet, September 2014, <https://blog.fortinet.com/post/a-new-version-of-the-3rd-generation-of-pushdo>
5. "A Closer Look Into the Pushdo Bot," Bit Defender, December 2013, <http://labs.bitdefender.com/2013/12/in-depth-analysis-of-pushdo-botnet/>
6. "A Study of the Pushdo / Cutwail Botnet," Trend Micro, May 2009, http://www.trendmicro.com/cloud-content/us/pdfs/business/white-papers/wp_study-of-pushdo-cutwail-botnet.pdf
7. "Pushdo and Cutwail: Enhancing Command and Control," Fortiguard, <http://www.fortiguard.com/legacy/analysis/pushdoanalysis.html>
8. "Operation Bot Roast – Botherders Charged as Part of Initiative," FBI, June 2007, http://www.fbi.gov/news/stories/2007/june/botnet_061307
9. "Researchers Kneecap 'Pushdo' Spam Botnet," Krebs on Security, August 2010, <http://krebsonsecurity.com/2010/08/researchers-kneecap-pushdo-spam-botnet/>
10. "Info on the Pushdo DGA," Kleissner, May, 2013, blog.kleissner.org/?p=219
11. "Cracking Pushdo and How to Bust Through Most Crypters," Open DNS Security Labs, July 2014, <https://labs.opendns.com/2014/07/31/pushdo-crypter/>
12. "Pushdo Takedown Damages Botnet," Trend Micro, 2010, <http://blog.trendmicro.com/trendlabs-security-intelligence/pushdo-takedown-damages-botnet>
13. "McColo," Wikipedia, 2008, en.wikipedia.org/wiki/McColo
14. "Pushdo Botnet Spams Malware Analysis Site, Researchers Find," PC World, September 112, 2013, <http://www.scmagazine.com/pushdo-botnet-spams-malware-analysis-site-researchers-find/article/311473/>

15. "Spammers Accelerate Dyre Distribution," Threat Track Labs Security, December 12, 2014, <http://www.threattracksecurity.com/it-blog/dyre-spam/>
16. "Unveiling The Latest Variant of Pushdo Mv20: A case study on the new Pushdo-DGA ," May 2013, https://www.damballa.com/downloads/r_pubs/Damballa_mv20_case_study.pdf
17. "PushDo Evolves Again: Enhances Evasion with Domain Generation Algorithm (DGA)," May 2013, <https://www.damballa.com/pushdo-evolves-again-enhances-evasion-with-domain-generation-algorithm-dga/>
18. "Frimeset[.]com," Virus Total, <https://www.virustotal.com/en/domain/frimeset.com/information/>
19. "2013 BSA Global Software Survey," BSA, June 2014, http://globalstudy.bsa.org/2013/downloads/studies/2013GlobalSurvey_inbrief_en.pdf
20. Pushdo DGA1 domains for 2015, <http://www.fidelissecurity.com/sites/default/files/pushdo-2015-dga1.txt>
21. Pushdo DGA2 domains for 2015, <http://www.fidelissecurity.com/sites/default/files/pushdo-2015-dga2.txt>