The DGA of DirCrypt

The DGA

DirCrypt is an inactive Ransomware that uses a Domain Generation Algorithm (DGA) for its callback call. Because I couldn`t find the DGA algorithm online, I decided to reverse engineer this sample from malwr.com. I list more samples that use the DGA in section *Sample on malwr.com*.

The DGA of DirCrypt uses a hardcoded seed located in the resource section of the executable. For the examined sample, the seed is labeled with the integer identifier `0x7D`:

For my sample, the value of resource identifier `0x7D` was `0xF2113C2A`:

The malware passes the seed and the number of distinct domains it wants to generate to a subroutine I called `spawn_6_callback_threads`:

The subroutine creates six callback threads - all getting a pointer to the same structure with seed and number of domains. The routine will wait for all six threads to finish before it returns:

The callback routine `callback_loop` creates new domains with the following routine “ `the_dga` ”. The counter `dga_nr_of_domains` (initialized to 30) is decreased after a new domain is generated. The thread returns when a

command-and-control callback is successful or the counter reaches zero. Here is the disassembly of the DGA:

```
 UPX0:0040183B the_dga         proc near
UPX0:0040183B
UPX0:0040183B seed            = dword ptr  4
UPX0:0040183B domain          = dword ptr  8
UPX0:0040183B
UPX0:0040183B                 push    ebx
UPX0:0040183C                 push    esi
UPX0:0040183D                 push    edi
UPX0:0040183E                 push    20
UPX0:00401840                 push    8
UPX0:00401842                 lea     eax, [esp+14h+seed]
UPX0:00401846                 push    eax
UPX0:00401847                 call    rand_int
UPX0:0040184C                 mov     ebx, [esp+0Ch+domain]
UPX0:00401850                 mov     edi, eax
UPX0:00401852                 xor     esi, esi
UPX0:00401854                 test    edi, edi
UPX0:00401856                 jbe     short loc_40186E
UPX0:00401858
UPX0:00401858 loc_401858:
UPX0:00401858                 push    'z'
UPX0:0040185A                 push    'a'
UPX0:0040185C                 lea     eax, [esp+14h+seed]
UPX0:00401860                 push    eax
UPX0:00401861                 call    rand_int
UPX0:00401866                 mov     [esi+ebx], al
UPX0:00401869                 inc     esi
UPX0:0040186A                 cmp     esi, edi
UPX0:0040186C                 jb      short loc_401858
UPX0:0040186E
UPX0:0040186E loc_40186E:
UPX0:0040186E                 push    offset a_com    ; ".com"
UPX0:00401873                 add     edi, ebx
UPX0:00401875                 push    edi
UPX0:00401876                 call    strcpy
UPX0:0040187B                 mov     eax, [esp+0Ch+seed]
UPX0:0040187F                 pop     edi
UPX0:00401880                 pop     esi
UPX0:00401881                 pop     ebx
UPX0:00401882                 retn    8
UPX0:00401882 the_dga         endp
UPX0:00401882
```

with `rand_int` being:

```
 UPX0:00404E9E rand_int        proc near
UPX0:00404E9E
UPX0:00404E9E
UPX0:00404E9E seed            = dword ptr  4
UPX0:00404E9E lower           = dword ptr  8
UPX0:00404E9E upper           = dword ptr  0Ch
UPX0:00404E9E
UPX0:00404E9E                 mov     eax, [esp+upper]
UPX0:00404EA2                 sub     eax, [esp+lower]
UPX0:00404EA6                 push    eax             ; span
UPX0:00404EA7                 push    [esp+4+seed]
UPX0:00404EAB                 call    rand_mod
UPX0:00404EB0                 add     eax, [esp+lower]
UPX0:00404EB4                 retn    0Ch
UPX0:00404EB4 rand_int        endp
```

and `rand_mod` being a standard *linear congruential generator*:

```
 UPX0:00404E6B rand_mod        proc near
UPX0:00404E6B
UPX0:00404E6B
UPX0:00404E6B seed            = dword ptr  4
UPX0:00404E6B span            = dword ptr  8
UPX0:00404E6B
UPX0:00404E6B                 mov     ecx, [esp+seed]
UPX0:00404E6F                 mov     eax, [ecx]
UPX0:00404E71                 xor     edx, edx
UPX0:00404E73                 push    esi
UPX0:00404E74                 mov     esi, 127773
UPX0:00404E79                 div     esi
UPX0:00404E7B                 pop     esi
UPX0:00404E7C                 imul    eax, 2836
UPX0:00404E82                 imul    edx, 16807
UPX0:00404E88                 sub     edx, eax
UPX0:00404E8A                 mov     eax, [esp+span]
UPX0:00404E8E                 mov     [ecx], edx
UPX0:00404E90                 lea     ecx, [eax+1]
UPX0:00404E93                 mov     eax, edx
UPX0:00404E95                 xor     edx, edx
UPX0:00404E97                 div     ecx
UPX0:00404E99                 mov     eax, edx
UPX0:00404E9B                 retn    8
UPX0:00404E9B rand_mod        endp
```

As mentioned above, all six threads access — inside a *critical section* — the same `seed` and `dga_nr_of_domains`. Therefore, at most 30 different domains are created. The following Python code generates the 30 domains of the DGA for a given seed:

```
 import argparse

class RandInt:

    def __init__(self, seed):
        self.seed = seed

    def rand_int_modulus(self, modulus):
        ix = self.seed
        ix = 16807*(ix % 127773) - 2836*(ix / 127773) & 0xFFFFFFFF
        self.seed = ix
        return ix % modulus

def get_domains(seed, nr):
    r = RandInt(seed)
    for i in range(nr):
        domain_len = r.rand_int_modulus(12+1) + 8
        domain = ""
        for i in range(domain_len):
            char = chr(ord('a') + r.rand_int_modulus(25+1))
            domain += char
        domain += ".com"
        yield domain

if __name__=="__main__":
    parser = argparse.ArgumentParser(description="generate Dircrypt domains")
    parser.add_argument("seed", help="seed as hex")
    args = parser.parse_args()
    for domain in get_domains(int(args.seed, 16), 30):
        print(domain)
```

For example:

```
 $ python dga.py f2113c2a
rauggyguyp.com
llullzza.com
mluztamhnngwgh.com
mycojenxktsmozzthdv.com
inbxvqkegoyapgv.com
furiararji.com
zrkdvzjhse.com
wyuhdsdttczd.com
```

```
hpaxgpkteomjaxywwelr.com
mydojltbqjnwailyyoa.com
wbgzpjfxlxlcvbth.com
pibqzedhzwt.com
vlbqryjd.com
nsxdczggybtkdukmyf.com
jarjvddjzqrmnepeqwd.com
plxeyaja.com
lfehajeex.com
swtjyuhuefvl.com
ftdkuoulfhfudds.com
eblgaosyeszzjkbhhdyh.com
afececrkycbeyqm.com
xnloppwhfamkcltuxkif.com
xjjcditjfkgkihfe.com
mblmvrla.com
vxlkofoazme.com
ktqyrmiyvnidd.com
jsntwyjcv.com
wvquldqwwsttp.com
pivzovznpssx.com
ggspyfmreouxnhqi.com
```

The following table summarizes the properties of the DGA:

property	**value**
seed	hardcoded in resource section of executable
domains per seed	30
tested domains	all
sequence	one after another, but DNS queries can occur out of order because six concurrent threads make callback calls
wait time between domains	none
top level domain	`.com` for all observed seeds
second level characters	lower case letters, picked uniformly at random
second level domain length	8 to 20 characters

Samples on malwr.com

I sifted through all samples on malwr.com where at least one of the virus scanners identified the sample as *"DirCrypt"*. I then brute forced the seed that leads to the observed domains. Because the callbacks run in six concurrent threads, the domains sometimes appear out of order. Also, some of the DirCrypt samples use an additional hardcoded domain: *pdstriker.com*, *oktendentaries.com* or *jwuiygpnslht.com* (this domain is generated by the DGA, just not with the hardcoded seed).

The following table lists the md5 hash of the sample (linked to the analysis on malwr.com), the submission date to malwr.com, the used seed, and any additional domains that are not covered by the DGA's seed. The periodicity of the pseudo random number generator is $2^{32}/2$ or half the number range; therefore, there are two seeds for each sequence of domains.

seed	md5	date	not covered
18a62b7a, 98a62b79	4bb6c6c3f1ad7c2fb6096f6156c1df9b	10. Jul. 2013	pdstriker.com
18a62b7a, 98a62b79	3c03f0478ed6b0e81397b8e93cd4be90	29. Jul. 2013	
1fcbef63, 9fcbef62	339901b416c580d4d6c7fae4a088d2e4	28. Aug. 2013	oktedentaries.com
18a62b7a, 98a62b79	d224637a6b6e3001753d9922e749d00d	06. Sep. 2013	
1a11b7cd, 9a11b7cc	c1c117a8fbcd87b1c52a7c1c8e4bd2c9	30. Sep. 2013	
72113c2b, f2113c2a	dd69a49ab475dafc7246dee9f0f4c877	06. Oct. 2013	
72113c2b, f2113c2a	42b77df04c7c34294c0e9459550cde9b	06. Oct. 2013	
72113c2b, f2113c2a	fa126a680351484beb450053e7ccccd0	06. Oct. 2013	
72113c2b, f2113c2a	e53d4e64930a40a12cd994f2779a11e9	07. Oct. 2013	
1a11b7cd, 9a11b7cc	7d978608d8fbaf3b756d692fff243450	15. Oct. 2013	
741fd6e2, f41fd6e1	70b86fdf69b8059ed4bf12e2a7707ae6	23. Oct. 2013	

seed	md5	date	not covered
72113c2b, f2113c2a	70d0a1b577dde513a0dfae09722d3ddd	25. Oct. 2013	
6c75a989, ec75a988	0a807e0a2d29f19c95b313d018e1c2bd	16. Nov. 2013	
72113c2b, f2113c2a	a88cfaa2e408df1245d74d0b50531976	02. Dec. 2013	
72113c2b, f2113c2a	1186590b731d17206c63aadbe5a0484a	02. Dec. 2013	
78731d07, f8731d06	0e5e8f6edd2c1496614bb6a71ba3f256	10. Dec. 2013	jwuiygpnslht.com *6522e630, e522e62f*
6c75a989, ec75a988	b2752b6151b6fd8342e68b9bd5aa632b	11. Dec. 2013	
6e46566, 86e46565	f99f10c3a02eff983e99216cd5f54ce9	31. Dec. 2013	
6c75a989, ec75a988	f7b0ae2f4d669e3705b60fe20a5bbf7a	08. Jan. 2014	
1fcbef63, 9fcbef62	ee3c8b0bbea638e10eda11fa042069e0	11. Jan. 2014	oktedentaries.com
52ce8a67, d2ce8a66	80b356b9203d7e494ccc795d15999133	19. Apr. 2014	
22a47ee8, a2a47ee7	83f94b0697e3d69c3b219191984620d6	22. Apr. 2014	
52ce8a67, d2ce8a66	bbc1d7261ee18363aa2677708abeb5a0	25. Apr. 2014	
52ce8a67, d2ce8a66	08956c46e09c2375a6ee64313adc9d4a	26. Apr. 2014	
52ce8a67, d2ce8a66	ec92487de0c66ceac950daff102c5576	03. May. 2014	
52ce8a67, d2ce8a66	b9e7b880bd095d11c16d6adc40eaff3d	05. May. 2014	

seed	md5	date	not covered
4caa1fc5, ccaa1fc4	1451cf7b82c70be7ea6744b69acc9960	29. May. 2014	
4caa1fc5, ccaa1fc4	bc918d15033b2f97bc0ba745949577d2	29. May. 2014	
52ce8a67, d2ce8a66	0d24562e7e2ae008b757c471976bd2f6	29. May. 2014	
52ce8a67, d2ce8a66	245d39fad0e9c31dfac810ae413e4a96	30. May. 2014	
52ce8a67, d2ce8a66	44bc29f11d907a33eca52cb1c872f9d6	30. May. 2014	
52ce8a67, d2ce8a66	5af46d0edfffb0089dd1c1c9945e1170	30. May. 2014	
52ce8a67, d2ce8a66	ba682f257c4acf0d706e4ed29cabf476	20. Jul. 2014	

Most samples use the seed `52ce8a67 / d2ce8a66` (10 samples) and `72113c2b / f2113c2a` (7 samples). The following table summarizes the seeds that I was able to identify, the first five generated domains, and the number of samples on malwr.com:

seed	first 5 domains	found hashes
6e46566, 86e46565	wejcqzbosbczzlnikyvt.com, muiccxbvkvjb.com, tqwpmpwckhidiss.com, gzredieexn.com, ghhcwldtj.com	1
72113c2b, f2113c2a	rauggyguyp.com, llullzza.com, mluztamhnngwgh.com, mycojenxktsmozzthdv.com, inbxvqkegoyapgv.com	7
741fd6e2, f41fd6e1	cbhytcvyxzzj.com, ervqveknzq.com, jxuynwdac.com, bucelslmpwyajzlguis.com, zhszoxeavbhmtkbju.com	1
1a11b7cd, 9a11b7cc	lldpoyrzfi.com, chbqrhunxg.com, iqhbyacfnea.com, lgsfbhyyrrnalpcbqkob.com, fktihyjhkomdxqkucg.com	2
18a62b7a, 98a62b79	viweabkkfe.com, lscyqrjofqmtn.com, ltcfpuctidqqqxxzpikz.com, wowsfhnnvlwhlotryvh.com, linbzxpkmdtngnbdg.com	3
4caa1fc5, ccaa1fc4	qjdygsnoiqaudcq.com, iwqvktutvmptevjbnzy.com, vcgietkhdgvjhhsbdu.com, mkhjbvxvuqznmcjmy.com, jgtkrjdnqeyrjpbnqxym.com	2

seed	first 5 domains	found hashes
1fcbef63, 9fcbef62	fzfqphobttefkhbvkzs.com, pmyddiicql.com, pihxsxitdfzpvpgeusf.com, glurejnjtdbj.com, oomxzlhazpiz.com	2
78731d07, f8731d06	ttaebamktjdbizrnqxp.com, znpszzwstgzzyk.com, jsngvficglxttjwg.com, frwwkrpnkvig.com, egdbvrhtcptgoqorompj.com	1
52ce8a67, d2ce8a66	aexluxmagbyg.com, izllzixotympqqr.com, pwxqjnhsocyln.com, pmzlyoesekeqytc.com, ypveltysbgcpm.com	10
22a47ee8, a2a47ee7	mhrmhuxlcvkxay.com, lvphxfvpsigghujpdm.com, ctskthnhq.com, safkylboxhb.com, gcifbxymnmmdfay.com	1
6c75a989, ec75a988	hiuctidthkvwowhvo.com, fcnjgeiicc.com, jpryjfvwlf.com, mlavvgdzq.com, rvcysvtrdqvfeoxpkgay.com	3

Source: https://www.johannesbader.ch/2015/03/the-dga-of-dircrypt/