ANALYZING DGAS WITH FRIDA

Leron Gray

CSC-840 Final

DGAS

- Domain Generation Algorithms (DGAs) are used by malware to generate a number of domains for C2 servers.

- The values used for the algorithm can be:
  - Deterministic – (e.g., time, dates)
  - Non-deterministic – (e.g., stock market prices)

- DGAs can be mathematically complex for obfuscation
  - Reverse engineering the algorithm can take a significant amount of time depending on complexity

RANBYUS DGA GENERATOR

- Takes in day, month, year, hardcoded seed, and hardcoded number of domains to generate.
- Has hardcoded TLDs.
- Domain generation is somewhat complex. Might take hours or days to confirm in a disassembler.
- This rewrite prints the domains but the malware likely returns a list of values to the function that called it.

```c
#include <stdio.h>
#include <stdlib.h>

char* dga(unsigned int day, unsigned int month, unsigned int year,
        unsigned int seed, unsigned int nr)
{
    char *tlds[] = {"in", "me", "cc", "su", "tw", "net", "com", "pw", "org"};
    char domain[15];
    int d;
    int tld_index = day;
    for(d = 0; d < nr; d++)
    {
        unsigned int i;
        for(i = 0; i < 14; i++)
        {
            day = (day >> 15) ^ 16 * (day & 0x1FFF ^ 4 * (seed ^ day));
            year = ((year & 0xFFFFFFF0) << 17) ^ ((year ^ (7 * year)) >> 11);
            month = 14 * (month & 0xFFFFFFFE) ^ ((month ^ (4 * month)) >> 8);
            seed = (seed >> 6) ^ ((day + 8 * seed) << 8) & 0x3FFFF00;
            int x = ((day ^ month ^ year) % 25) + 97;
            domain[i] = x;
        }
        printf("%s.%s\n", domain, tlds[tld_index++ % 8]);
    }
}

main (int argc, char *argv[])
{
    if(argc != 5) {
        printf("Usage: dga <day> <month> <year> <seed>\n");
        printf("Example: dga 14 5 2015 b6354bc3\n");
        exit(0);
    }

    dga(atoi(argv[1]), atoi(argv[2]), atoi(argv[3]),
            strtoul(argv[4], NULL, 16), 40);
}
```

VALUES GO IN

DOMAINS COME OUT

FRIDA

- Frida is a reverse engineering framework and instrumentation tool.
  - https://frida.re/
- Usually discussed in the context of mobile app security research.
- Also has features for use with Windows, Linux, macOS binaries.
- Has standalone tools but the library can be used with Python, C, or Swift to inject Javascript into processes.
- Allows for dynamic analysis of memory and function calls and can create of new functionality inside a running process.
- Basically reverse engineering magic.

FRIDA INTERCEPTOR

```
1 from __future__ import print_function
2 import frida
3 import sys
4
5 session = frida.attach("hello")
6
7 script = session.create_script("""
8 Interceptor.attach(ptr("%s"), {
9     onEnter: function(args) {
10         send(args[0].toInt32());
11     }
12 });
13 """ % int(sys.argv[1], 16))
14
15 def on_message(message, data):
16     print(message)
17
18 script.on('message', on_message)
19 script.load()
20 sys.stdin.read()
```

https://frida.re/docs/functions/

Line 5

Attach to process named hello

Lines 7 – 13

- Intercept call at given address and print argument 0 as an integer

Lines 15 – 18

- Allow Interceptor to send data back to Python

DGA AND FRIDA

- If VALUES GO IN and DOMAINS COME OUT, we can use Frida to log the arguments to the DGA function and return the values of the domains generated.
- DGA should still be analyzed but this may help save time.
- Since executable must be run to do this, it's best to neuter any malicious functionality beforehand.

DEMO

HTTPS://YOUTU.BE/TGNINZKXV0U

CONCLUSION

- Frida can help determine domains of DGAs with the values are deterministic.
  - Saves time when the DGA code is determined.

- May not always be as simple
  - Manual analysis is still required to determine how the results are returned