Blue Galaxy Energy: a new White-box Cryptanalysis Open Source Tool (./blue-galaxy-energy-a-new-white-box-cryptanalysis-open-source-tool.html)

Date Thu 21 December 2023 By Nicolas Surbayrole (./author/nicolas-surbayrole.html) Philippe Teuwen (./author/philippe-teuwen.html) Category Cryptography (./category/cryptography.html) Tags cryptography (./tag/cryptography.html) white-box (./tag/white-box.html) tool (./tag/tool.html) release (./tag/release.html) BGE (./tag/bge.html) 2023 (./tag/2023.html)

We introduce a new white-box cryptanalysis tool based on the pioneering BGE paper but without known open source public implementation so far.

Introduction

A few months ago, we presented Dark Phoenix in this blog post (https://blog.quarkslab.com/dark-phoenix-a-new-white-box-cryptanalysis-open-source-tool.html), a cryptanalysis tool performing *Differential Fault Analysis* (DFA) against AES white-boxes with so-called *external encodings*, completing the existing Side-Channel Marvels (https://github.com/SideChannelMarvels) set of tools.

Dark Phoenix differed from the *Differential Computation Analysis* (DCA) attack and the DFA tool implemented in Jean Grey (https://github.com/SideChannelMarvels/JeanGrey) by the fact that it can attack implementations using external encodings, i.e., extra layers of obfuscation applied to the data before being sent to the AES and removed afterward. However, this came at the cost of reverse-engineering efforts to isolate and run individual rounds of the implementation, while the two other attacks can be largely automated.

The same holds for the *BGE attack*: it is able to defeat AES white-box implementations with or without external encodings, but at the cost of some prior reverse-engineering.

In this blog post, we highlight our open-source implementation of this attack introduced in 2004. That's our way of celebrating this 20th anniversary!

Blue Galaxy Energy

Hologram: Shut up! You do not know the power of the Blue Galaxy Energy (https://marvel.fandom.com/f/p/3343172654596164836/)! Also known as the "B.G.E". Mr. Whereabout: The Loss, Part III, Volume I

Blue Galaxy Energy is a tool designed for executing the so-called BGE attack described in *Cryptanalysis of a White Box AES Implementation* (https://doi.org/10.1007/978-3-540-30564-4_16) by Olivier Billet, Henri Gilbert and Charaf Ech-Chatbi, with the optimizations proposed in *Improved cryptanalysis of an AES implementation* (https://api.semanticscholar.org/CorpusID:117052545) by Ludo Tolhuizen and in *Revisiting the BGE Attack on a White-Box AES Implementation* (https://ia.cr/2013/450) by Yoni De Mulder, Peter Roelse and Bart Preneel.

Installation

To install the tool, install gmp and ntl libraries and development headers with your OS package manager.

```
$ sudo apt install libgmp-dev libntl-dev
```

or

```
$ sudo pacman -S gmp ntl
```

Then compile and install the Python module in a virtual environment.

```
$ python3 -m venv venv
$ source venv/bin/activate
$ pip install bluegalaxyenergy
```

Usage

Similarly to Dark Phoenix, to use this tool against a given white-box AES implementation, you need to provide an implementation of your own class inheriting from the provided `WhiteBoxedAES` class.

This class serves as the interface between the white-box and the attack script. It must be capable of applying a single round of the white-box implementation to attack and return the intermediate state.

Example

We will take the NoSuchCon 2013 white-box as target example for this BGE attack (https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_nsc2013/BGE).

This white-box has the particularity of having external encodings and cannot be attacked with classical DCA or DFA.

Since the NoSuchCon 2013 white-box structure is well understood, it is possible to provide a method that performs a single round at once.

The class to be written is identical to the one we wrote in our previous blog post for Dark Phoenix, except that the base class comes from the Blue Galaxy Energy module.

Create a file `nosuchcon_2013_whitebox.py` :

```python
from bluegalaxyenergy import WhiteBoxedAES
class NSCWhiteBoxedAES(WhiteBoxedAES):
    def __init__(self):
        with open("../RE/result/wbt_nsc", "rb") as f:
            # initialize tables based on the white-box file
            self.initSub_sub = list(f.read(0x100))
            self.initSub_inv_sub = list(f.read(0x100))
            self.finalSub_sub = list(f.read(0x100))
            self.finalSub_inv_sub = list(f.read(0x100))
            self.xorTables0 = list(f.read(0x10000))
            self.xorTables1 = list(f.read(0x10000))
            self.xorTables2 = list(f.read(0x10000))
            self.roundTables=[[[None]*4 for _ in range(16)] for _ in range(9)]
            for i in range(9):
                for j in range(16):
                    for k in range(4):
                        self.roundTables[i][j][k] = list(f.read(0x100))
            self.finalTable=[None]*16
            for i in range(16):
                self.finalTable[i] = list(f.read(0x100))

    def getRoundNumber(self):
        return 10

    def isEncrypt(self):
        return True

    def hasReverse(self):
        return False

    def apply(self, data):
        for round in range(10):
            data = self.applyRound(data, round)
        return data

    def applyRound(self, data, roundN):
        output=[None]*16
        if roundN < 9:
            for i in range(16):
                b = [0, 0, 0, 0]
                for j in range(4):
                    b[j] = self.roundTables[roundN][i][j][data[j*4+((i+j)%4)]];
                    output[i] = self.xorTables2[(self.xorTables0[(b[0]<<8)|b[1]] << 8) |
                                                self.xorTables1[(b[2]<<8)|b[3]]]
        else:
            for i in range(16):
                output[i//4 + (i%4)*4] = self.finalTable[i][data[(i&(~3)) +((i+i//4)%4)]]
        return output
```

To execute the attack, we need to write the following script and optionally specify the rounds on which the attack should be applied. Typically, the first inner rounds have fewer countermeasures compared to the last rounds, as those are designed to defend against DFA attacks with potentially unconventional structures. However, it is important to note that the attack requires three consecutive rounds to extract a single round key. Therefore, for AES128, a minimum of three consecutive rounds is needed to extract the key and for AES192 and AES256, the minimum is four consecutive rounds.

Create a file `runme.py` :

```python
from bluegalaxyenergy import BGE
from nosuchcon_2013_whitebox import NSCWhiteBoxedAES

bge = BGE(NSCWhiteBoxedAES())
bge.run(roundList=[2,3,4,5])
key = bge.computeKey()
if key is not None:
    print("key:", key.hex())
```

```
$ python3 runme.py
```

If at least two round keys were found and the previous `computeKey` operation failed, it may mean that the round keys were transposed. Actually, it is the case for this particular white-box implementation and it is necessary to indicate that the round keys were transposed.

```
key = bge.computeKey(transposed_rk=True)
if key is not None:
    print("key:", key.hex())
```

The key is now recovered in less than 5 seconds.

```
$ time python3 runme.py
key: 4e5343234f707069646123b8dce442d0

real    0m1,464s
user    0m4,466s
sys 0m0,107s
```

A second more complex example is also provided (https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_grehack2019/BGE) against the white-box implementation of the GreHack2019 CTF. It utilizes QBDI (https://qbdi.quarkslab.com/) to instrument the binary. Feel free to take a look at it.

Limitations

The current version of Blue Galaxy Energy has some limitations:

- It only supports white-box implementations of AES encryption, not AES decryption ;
- It does not support the randomization in the order of the bytes of the intermediate results in AES, as mentioned in the De Mulder *et al.* paper ;
- It only supports 8-bit wide encodings.

It's important to note that deploying the BGE attack on a real white-box implementation can be significantly more complex compared to applying DFA or DCA attacks.

We have based our example on a naked version of the NoSuchCon 2013 white-box, which was the result of reverse-engineering efforts (http://0vercl0k.tuxfamily.org/bl0g/?p=253) by Axel Souchet, who initially worked on the Windows executable, to obtain an equivalent but still obfuscated source code. We then performed some post-processing to obtain clean tables and the round structure used in our `NSCWhiteBoxedAES` class. More details about this process can be found in the Deadpool repository (https://github.com/SideChannelMarvels/Deadpool/tree/master/wbs_aes_nsc2013/RE) and in the write-up provided on the Yobi wiki (https://wiki.yobi.be/index.php/NSC_Writeups#Epilogue).

Conclusion

Indeed, the difficulty of applying the BGE attack to a white-box implementation is directly related to the complexity of reverse engineering its obfuscation layers. However, the BGE attack becomes straightforward and highly effective if these obfuscation layers can be successfully removed.

Blue Galaxy Energy is released under the Apache 2.0 license (https://www.apache.org/licenses/LICENSE-2.0). The source code can be found in the Blue Galaxy Energy repository (https://github.com/SideChannelMarvels/BlueGalaxyEnergy).

For more information about the project, please refer to its README (https://github.com/SideChannelMarvels/BlueGalaxyEnergy/blob/main/README.md). If you're interested in diving into the technical details of the implementation choices, you'll find them there (https://github.com/SideChannelMarvels/BlueGalaxyEnergy/blob/1bdc668be01b1d6a6dd61ec4c69fa23fbd56e56c/src/bluegalaxyenergy/README.md).

Enjoy using Blue Galaxy Energy to analyze other white-box implementations with external encodings, and feel free to share your results whenever possible. Feedback, suggestions for improvement, and contributions to support decryption AES or bits shifting are always welcome.

Acknowledgments

We extend our gratitude to Laurent Grémy, who authored the core implementation of Blue Galaxy Energy.

If you would like to learn more about our security audits and explore how we can help you, get in touch with us (https://content.quarkslab.com/talk-to-our-experts-blog)!

Back to top