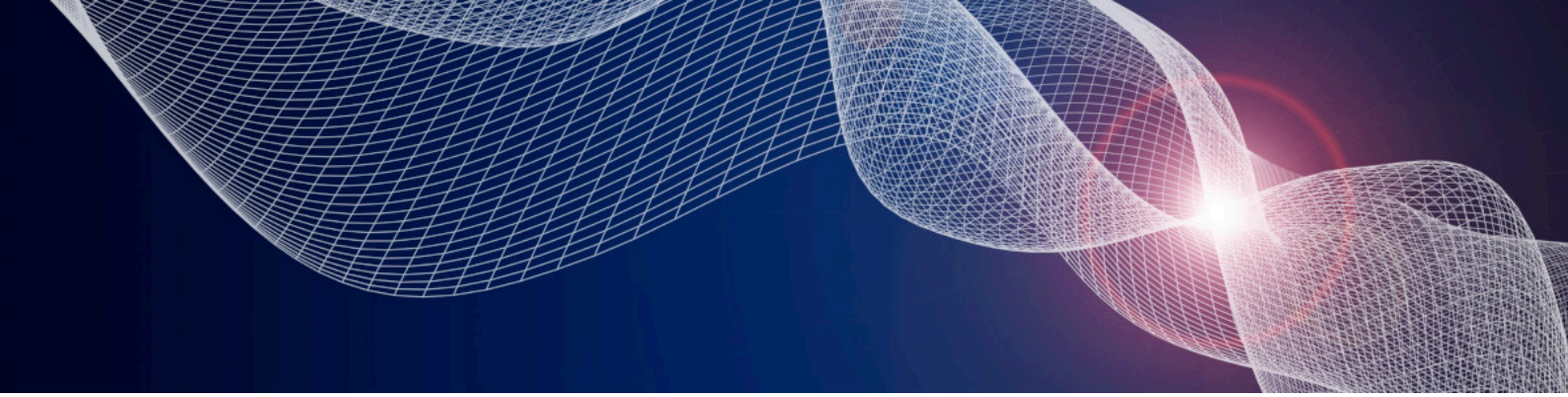

2. Deobfuscation

2.1 Analysis

2.1.1 VBA (80')

- Download this *maldoc* [1] .

> DO NOT open it in office! Or at least don't enable macros...

1. Run *olevba* to extract the obfuscated code inside the maldoc.
2. Have a look at the code.
    1. How is the execution of the code triggered?
    2. What obfuscation techniques have been used?
3. See how far you get with manual deobfuscation of the code.
    1. Describe the steps you took.
    2. Any idea what the VBA part of this malware does?

2.1.1.1 Automated Deobfuscation (40')

Use *ViperMonkey* to analyze the above mentioned maldoc. Compare with the result of your manual analysis. No need to submit any results but make sure you can run *ViperMonkey* and get a result.

[1] https://github.com/ti-ng/re-deobfuscation/raw/master/Sample.doc

2.1.2 PowerShell

2.1.2.1 Language (60')

When you were done with the VBA deobfuscation, you ended up with something like the following excerpt (or to be safe, just get it from here [2]):

DOS

```
1 powershell -w hidden -enc IABTAGUAVAAtAHYAQQBSAGkAYQBCAEwAZQAgACgAIgBUADQ ... CcAKQA=
```

Any idea what that could be? To get started on analyzing the program designed to be launched in `PowerShell`, have a look at what the `-enc` option does!

1. Create a *CyberChef* recipe which gets you as far as possible.
   - You can export the recipe by using "Save Recipe" and copying from "Chef Format".
2. What obfuscation techniques have been used?
3. What does the PowerShell part of this malware do?

2.1.2.2 Automated Deobfuscation (60')

1. Use *PSDecode* to analyze the above mentioned code. Compare with the result of your manual analysis.
2. Where did the automated analysis fail? Any idea why?
3. Can you overcome this and manually figure out what the malware does?

[2] https://github.com/ti-ng/re-deobfuscation/raw/master/PowerShellInput.txt

2.1.3 JavaScript (80')

Download this malware sample [3] .

> DO NOT execute the contained JS (at least not if you're on a Windows host).

1. Have a look at the code. What obfuscation techniques have been used?
2. See how far you get with manual deobfuscation of the code.
    1. Describe the steps you took.
    2. Can you devise what this malware does?

2.1.3.1 Automated Deobfuscation (40')

1. Use *box-js* to analyze the above mentioned maldoc. Compare with the result of your manual analysis.
2. How far did the analysis get you? What's missing?

[3] https://github.com/ti-ng/re-deobfuscation/raw/master/Sample.js