FokerScript – Documentación

FokerScript v0.1.0.alpha.0

Introducción

¡Hola! Soy Stunx y esta es la documentación oficial para FokerScript. En este documento se explica de que va el lenguaje, sus limitaciones, sintaxis, tips, uso y, por supuesto, la documentación total de la librería standard al final del libro.

¿Por qué FokerScript?

Buena pregunta, FokerScript nació de la necesidad de mejorar el desarrollo de scripts de la 3ra generación de los juegos de Pokémon Advance en el ROMHacking binario. Aunque bien es cierto que ya no se debería de usar el hackeo en binario, sino usar la decompilación, hay algunas personas que aún siguen insistiendo en usar el binario, ya sea porque la decompilación no la entienden bien, no tienen conocimiento del lenguaje C o cualquier otra excusa.

Quisiera decir que, en mi opinión, se debería recurrir a la decompilación para desarrollar un juego sobre la saga Pokémon y abandonar completamente el binario. Pero el motivo por el cual hago este proyecto, además de lo ya explicado arriba, es el hecho de que aún existen hacks hechos en binario que no se han concluido, ejemplo de esto es Pokémon Ancient, cuyo autor, Nacho, aún sigue en su desarrollo en binario.

Beneficios de usar FokerScript

Al igual que Poryscript facilita el desarrollo de scripts en decomp, FokerScript planea el mismo objetivo en el binario. Los beneficios que trae usar FokerScript son

- **Scripts más legibles:** Es cierto que el scripting en binario es algo confuso si no se logra entender bien del todo, pero con FokerScript esto se hace legible y fácil de entender.
- **Modularización:** Mientras que el preprocesador de XSE (eXtreme Script Editor) permite hacer inclusiones de varios archivos de código RBH en un script, FokerScript permite cargar los módulos (o archivos de código Foker) directamente en un script sin problema alguno, para que pueda ser usado en este.

Esto son los beneficios que te trae el usar FokerScript en tus proyectos de ROMHacking binario, si esto es lo que buscabas, siéntete libre de continuar con el libro, de lo contrario vete pal’ carajo :).

Diferencias entre FokerScript y el scripting de XSE

He aquí una tabla de diferencias entre FokerScript y el scripting de XSE, siéntase libre de saltar este apartado si así lo desea.

Nombre del ejemplo	Scripting de XSE	FokerScript

¡Hola Mundo!	#dynamic 0x800000
	
	#org @main
	msgbox @msg 0x2
		script main {
		  msg(“¡Hola mundo!”);
		}

	end #org @msg = ¡Hola Mundo!	

Cambio del offset del dynamic	#dynamic 0x900000
	
	#org @main
	end
		dynamic 900000;
		
		script main {
		}

Buceando en FokerScript

Había una vez un jaizuneta, del cual no me acuerdo su nombre (¿qué raro no?), que expresó una frase que me encanto, la cual quiero citar aquí:

> “Un buceador principiante no puede nadar aún en el fondo, aunque este sea su principal objetivo.” Un Jaizuneta.

Con esa frase en mente, pensemos que nosotros somos buceadores principiantes, que ya estamos llenos de teoría pura sobre el [1]scripting de XSE y listos con el traje de buceo para iniciar este largo proceso de aprendizaje y practica.

Los bloques de script

Comencemos con lo más básico, es decir, los bloques de scripts. En FokerScript estos bloques son el corazón de un script, el cual se compone de varios statements (o declaraciones) y expressions (o expresiones).

Por ejemplo, veamos este simple archivo de código FokerScript:

```
script main {
    msg(“Hola mundo!”);
}
```

Tranquilo si no entiendes nada de lo qué está ahí, así que vamos por partes para lograr entender esto: Para declarar un bloque script necesitamos seguir el siguiente patrón:

[2]
```
block_script := ‘script’ <name> (‘(‘ param (‘,’ param)* ‘)’)? ‘{‘ (statements | expressions)*  ‘}’;
param := ident ‘:’ ident
```

En <*name*> ira el nombre de nuestro bloque de script, así, por ejemplo:

```
script miname {}
```

Crea un bloque de script llamado ‘miname’ sin contenido alguno. Los bloques de script pueden soportar variables en su cabecera, estos en programación se conocen como “parámetros”, son usados dentro de un bloque de script para ciertas utilidades, como por ejemplo, saber si el usuario ha hecho algo o no, etc. Para declarar los parámetros:

```
script obtener_edad_en_base_a(base: int) {
    match (base) {
        0 .. 5 {

        }
    }
}
d
```

1 Si no se tiene conocimiento sobre el scripting de XSE, por favor mirar los tutoriales del foro de Whack a Hack.

2 El formato aquí usado para los patrones está inspirado en la sintaxis de ANTLR4 (https://wikipedia.com/es/ANTLR)