DreamProtocol SC2 MapScript Obfuscator

使用说明文档

注：DreamProtocol SC2 MapScript Obfuscator 也被称为混淆器（提供 LC4，RTD/+ITP）。

混淆引擎功能

1. **重命名**
   规则引擎说明
   a. 包含所有 gv_，lv_这种类似开头的脚本字符
   b. 对 string（x22 引号中的字符进行额外分析），如果发现和其他变量存在全字匹配，则参与重命名例程
   c. 额外包含和例外规则，参见下方**混淆器的配置**
   重命名的目标默认为条码形态（小写 L，大写 i，数字 1 组成的三进制流）
   在深渊模式下，为（小写 O，大写 o，数字 0 组成的三进制流）

2. **常量混淆**
   a. **Int/Fixed 常量**
      例如 for 循环中的 start 和 end 表达式（1~12 循环）
      例如加密/校验中的固定常量，或者带有类似于编译器行为的关键值
      例如 0x5A827999/0xFFFFFF/0xFF00
      例如一些地图中固定的 XP 奖励数值，例如+1500XP，搜索 1500
      此过程不会影响原先的非整形运算的结果
      此保护机制不会对 GVM Runtime 造成任何性能影响（编译后恢复）
      DreamProtocol 不仅对这些常量进行表达式复杂化，并且会随机一些变量参与这些转化进程，这使得反混淆变的更为困难。
   b. **字符串常量**
      DreamProtocol 不再使用字符串转义的方式进行字符串保护
      取而代之的是字符串内卷技术，混淆器会扫描整个代码区域，提取所有代码中的字符串，重新分析成字节组，再封装到一个浮动的区域。
      这不仅保留了星际 2 原来的脚本特性，并且去除了所有的字符串。
      此保护机制会对 GVM Runtime 造成极其轻微性能影响（读图时解密）
      +1 引用 [间接访问] 直接资源区偏移计算变成获取引用，然后得到偏移
      请注意：如果是 PlayerHandle 类型的字符串，DreamProtocol 会令其在访问时候重新解密
   c. **基础库中的常量转化**
      例如 c_messageAreaObjective，c_bankOptionSignature 一些关键词会对脚本分析非常有用，使得很容易定位到一些行为的位置。
      混淆器在初始化混淆引擎的时候，会先获取脚本的库引用，并且加载库脚本，获取常量关键词，标记使用情况。然后转化所有的库常量到一个状态，这使得所有脚本中的这些关键词都引用了一个乱码变量，无法通过搜索追踪到。

3. 行为隐藏
   Natives 中提供了所有的对游戏进行访问和修改的 API
   无论执行什么操作，只要是要对游戏中产生实际行为，都必须调用这些。
   DreamProtocol 会隐藏所有的库函数调用
   这使得类似于 BankSave，GameOver，UIDisplayMessage 不再能够对脚本分析产生任何启发性的帮助。
   混淆器引擎在初始化的过程中会合并引用的库，并且对其进行引用分析。通过产生复杂的调用网络使得这些功能最后被合并入一个匿名调用组。这使得去混淆过程几乎变得不可能。混淆会“理解”并且重新生成新的调用代码，之后对其进行再链接。
   对 GVM Runtime 性能产生中等影响（基本感觉不到）

4. 变量保护
   存档的保护包含三个部分，然而绝大多数人只停留在第一个部分上，也就是加密及校验。混淆器除了提供第二部分的脚本分析障碍以外，还可以提供变量的内存分散。通常情况下，变量按照字节紧凑的堆放在 GVM 的虚拟堆上。Bool 会填充单个 Byte，Int、Fixed 4 字节，String 之类的引用是以 4 字节的引用表示，struct 类型的数据则预先分配所有的单元空间，一个接一个的填充在这片内存区域。
   这导致就算是今天，使用 CE 去掉 FastScan（快速扫描）的选项，就可以非常容易的搜索到这片区域中的 Int/Fixed 变量所在位置。
   混淆器可以让用户填写一些变量数组的名称，对其进行隐藏（分散和校验），这些数组类型包含，Int 组，Bool 组，String 组，其中：
   Int 组包含分散存储和值拆解及校验
   Bool 组会完全分散存储
   String 组会分散的存储成字节状态，仅在访问的时候构建临时结果
   在访问的时候造成中等性能影响，这意味着你不应该把大量的变量添加到这个保护列表中，仅应该存放经验值，或者是一些存档中的解锁状态。
   对于需要保护的 string，并且需要进行遍历 string 组的应该提前建立 Dictionary for Index 数据组，推荐使用 DataTable 构建字典。

使用混淆器的时候，先将脚本复制到单独的文件，在前面放置配置好的 HEADER 文本，然后复制到混淆器中，再按 LAUNCH，HEADER 的配置请参见混淆器配置

高级功能

DreamProtocol 给这个程序的定位除了脚本保护以外，本程序包含大量的高级功能

1. RTD 引擎

   RTD 是一个及时调试器，相比深渊模组使用 Pointer Swap 进行引用访问和修改之外，RTD 要友好的多。RTD 具备如下功能

   a. 变量（包含数组变量）设置新的参数

   b. 动态代码执行

   RTD 使得作者无需在地图中配置任何代码，即可在使用的时候达到一些目的，例如执行 natives，或者给脚本中的变量设置新的值，请注意这仍然受到重命名的影响，你必须保留重命名模块生成的变量名称对应表。

   RTD 使用 RTD 原生指令的方式来执行请求，类似于 NET CLR 这种中间指令。通常不会手动将代码翻译成 RTD 指令，而是使用 RTD 编译器。

   例如：

   ```
   UnitCreate(1, "DCM6200", c_unitCreateIgnorePlacement, EventPlayer(),
   CameraGetTarget(EventPlayer()), libNtve_gf_RandomAngle());
   ```

   在 RTD 编译器翻译之后如下：

   ```
   T*C*EventPlayer*1*
   T*C*EventPlayer*2*
   T*C*CameraGetTarget*3*1*
   T*C*libNtve_gf_RandomAngle*4*
   T*W*i*5*1
   T*W*s*6*DCM6200
   T*W*i*7*2
   T*C*UnitCreate*7*5*6*7*2*3*4*
   ```

   混淆器会扫描库引用，并且把库参与进脚本的逻辑分析，会拼接所有的变量和可访问的功能。这包含库中的变量和脚本中的变量，以及库中的函数和脚本中的函数，RTD 都可以对齐进行访问/修改/调用

   RTD 对每一个类型的数据类型都有相应的缓冲池，具备 50 个元素。

   请注意：RTD 的调用是自动根据参数的定义，从对应的缓冲池中取元素。

   RTD 的多模式性

   RTD 拥有几个扩展组件，增强 RTD 在多种情况下的兼容性

   RTD 可以拥有如下执行方式：

   a. 聊天指令

      你必须在混淆器的设置中预先定义管理员的 PlayerHandle，这使得你可以使用聊天命令操作 RTD

   b. RTD Pipe 2.0 – LEA RTD Pool

      RTD 可以通过特殊的内存调制，从而在游戏中显示一个对话框，使得你可以使用图形界面输入 RTD 指令，然而这个对话框在录像中是不可见的。除了使用内存调制器以外，没有其他办法呼出此对话框。

c. RTD Pipe 3.0 – Internal Quantum Teleportation
RTD 管道的 T3 科技允许远程 RTD 指令传送，缩写 RTD_ITP
这可以允许远程操纵地图运行，以及不会留下任何迹象。
RTD_ITP 中包含一个总线控制，最多会有 2 秒的延时。

2. LC4 卡顿检测
请参考其他文档中的 LC4 卡顿检测介绍，该份文档已经足够详细，而且包含混淆器中的配置说明。

3. CSP-RNG 随机器
混淆器可以额外生成一个 CSP-RNG 随机数表，用于忽略星际 2 的随机器
请注意该随机器有这非常好的随机器分布，但是这只会影响脚本中的随机
不过说实在的，随机这东西一言难尽，就算这玩意足够随机，也难逃命运的安排，DreamProtocol REMILIA 没有办法解释这个现象。

4. 本混淆器还包含一些特殊的功能，那些功能只能通过调试模式才能激活，这里就不再赘述

混淆器的配置

DreamProtocol 一般不会对这种东西提供图形界面，需要填写 HEADER 来完成配置，例如

```
//DreamProtocol ObFuscator Advance
//TRUSTED ADMIN: 5-S2-1-4233328-5-S2-1-8276085-5-S2-1-9049787-5-S2-1-9218313-5-S2-1-8021785-5-S2-1-9622531-3-S2-2-1108893
//RTD Enable
//LC4 Enable
//CSP-RNG Randomizer
//DreamProtocol ObFuscator Advance End
```

这种被称作 HEADER 的东西需要粘贴在脚本的开头，然后放入混淆器，再点击 LAUNCH
HEADER 由`//DreamProtocol ObFuscator Advance` 作为开始
HEADER 由`//DreamProtocol ObFuscator Advance End` 作为结束
选项分为两种，一种是单行选项，还有一种是多行选项
此外还有一种可以在脚本添加的注释

请注意所有的冒号都不是中文冒号，冒号之后都有个空格，该空格必须保留

一、单行选项

1. 让混淆器加载 RTD 模组至脚本

```
//RTD Enable
```

2. RTD 功能调用例外

```
//RTD Function Exclude: bit_field_transformer_t
```

RTD 的代码扫描模块非常古老，拥有一些缺陷，对于 funcref<T>中的虚函数，及只有定义没有实际执行说明的函数，RTD 也会尝试加载此功能，然而这将造成虚函数调用错误。需要手动添加这种行，以添加例外。

3. 强制重命名对

```
//Force Rename KeyPair: gf_AllowPhantasy-IllIlI1l
```

格式如上面所示，这将会让 `gf_AllowPhantasy` 重命名成 `IllIlI1l`，从而不是随机分配随机名称，以及 `IllIlI1l` 不会在其他的重命名结果中出现。

4. 禁用 Syscall 行为隐藏转换

```
//Disable Syscall API Transition
```

行为隐藏中，所有的 natives 功能会使用虚拟化的 Syscall 进行隐藏，该转化会对性能有一定的影响，如果有些 natives 功能每秒钟被调用上万次，则会出现一些卡顿，通常情况下脚本从来不会是造成游戏卡顿的因素。然而对于特殊的代码，这是个解决方案。这将会严重降低脚本的混淆强度！

5. 禁用字符串坍缩（关闭字符串常量保护）

```
//Disable Cosmic String Collapse
```

关闭这个东西会显著减小脚本的大小，然而对性能并没有什么帮助，因为本来就没有啥性能开销，然而会严重降低脚本的混淆强度！

6. 禁用 natives 常量转化

```
//Disable Hide Natives Constants
```

关闭这个东西不会对性能有任何有感觉的帮助，然而会严重降低脚本的混淆强度！

请注意，5 和 6 仅应当作为故障排除时候使用，如果你不具备很强的脚本阅读能力所有的故障排除应当由 DreamProtocol REMILIA 完成

7. 禁止对于 PlayerHandle 字符串的运行时解密（加强型保护）

```
//String Encryption Disable Decrypt on using
```

如果你的代码并不能很好的安排好玩家的权限，在运行时有大量的 PlayerHandle 比较，可以考虑启动此选项，这会稍微降低些许安全性。

8. 地图管理员

```
//TRUSTED ADMIN: 5-S2-1-4233328-5-S2-1-9622531
```

使用-分开不同的 PlayerHandle，这里的玩家可以控制 LC4 和 CSP-RNG 以及 RTD。

9. 启用 LC4

```
//LC4 Enable
```

10. 启用 CSP-RNG

```
//CSP-RNG Randomizer
```

11. 指定脚本的入口函数（默认为 InitMap）

```
//Script Entry: InitMain
```

通知混淆器，脚本的入口函数名为 `InitMain`

12. Syscall 排除的 natives

```
//Syscall Exclude: EventUnitDamageSourceUnit
```

将 natives 中的 EventUnitDamageSourceUnit 排除在 Syscall 虚拟化之外

在有些时候这可以缓解性能问题，例如地图中的伤害追踪器，一些每秒钟被调用上千次的 natives 功能，请注意，这个应该使用触发器调试来找出性能瓶颈，而不是通过猜测。

13. 使用泡泡重命名模式

```
//Use Bubbles Name Renaming
```

使用深渊模式下的重命名模式 o0O 组合来重命名变量，从而不使用默认的 I1l 组合

14. 拆分 InitTriggers 函数，禁止脚本编译时候内联其中函数

```
//Natives Hide Option: Split InitTriggers
```

星际的脚本编译会积极内联，然后发生语法检查时发生 e_jumpOutOfBounds 错误。此问题只会在触发器非常多的地图中发生，添加此选项可以避免此问题。

二、多行选项

其中的 Block 通常以//XXXXX 开始//XXXXX End 作为结束，Block 中每一行一个元素

1. Int 数组安全读写

```
//Int Array Move
//gv_playerExperience
//Int Array Move End
```

将 gv_playerExperience 添加到 Int 数组安全读写

2. 同理//Bool Array Move 和//String Array Move
3. 重命名例外//Rename Conditions

   混淆器的重命名不会对指定名称的变量进行重命名
4. 重命名额外的前缀//Idt A Additional Start 和//Idt A Additional End

   例如添加 libDCMV_，这样整个 LibDCMV 库都会参与重命名，亦可指定一个变量，例如 ImperialBank

三、脚本内注释

1. 仅允许执行一次

```
//Single Hit Limiter
```

放置该行在一个或者多个的函数内部，使其产生一个“屏障”，使得整场游戏只能通过这个屏障一次，当有第二次到达这处地方，地图聊天区域会显示一行警告。

Core Debug: Single Hit Limiter Blocked an Exception!

2. 禁止使用 Syscall 的区域

```
//Start Non Syscall Virtualization Region
//End Non Syscall Virtualization Region
```

相比较禁用掉整个 Syscall 行为隐藏以及禁用掉一些 natives 函数的隐藏，这可能是个更好的选项，这将禁用这片区域之间的所有 natives Syscall 虚拟化，这个标记可以在地图中有多处，但是有开始必须有结束。

疑难解答

联系 DreamProtocol REMILIA 以了解详细信息

你必须详细备注。添加 QQ 好友 1308080807