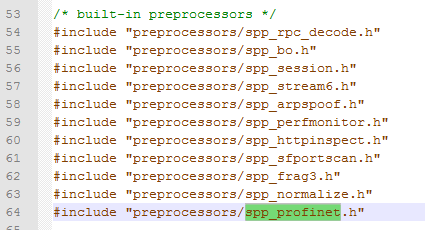
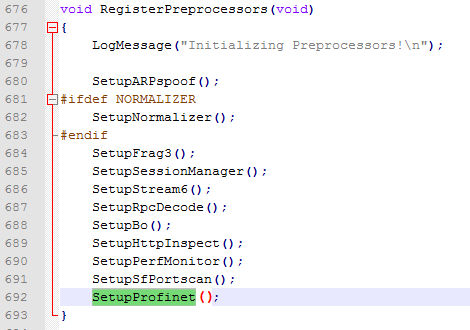
# Snort预处理器开发流程

使用Snort版本为2.9.8.2

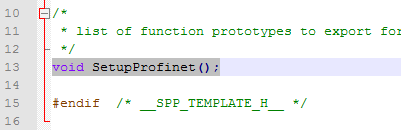
1. 复制snort-2.9.8.2/templates/spp\_template.c snort-2.9.8.2/templates/spp\_template.h到snort-2.9.8.2/src/preprocessors，重命名为spp\_profinet.c spp\_profinet.h （名字任意）
2. 修改snort-2.9.8.2/src/plugbase.c 文件，将插件的头文件spp\_profinet.h 包含到plugbase.c 中如图64行



1. 将插件的 SetupProfinet()函数插入plugbase.c 的 RegisterPreprocessors()函数中



1. 修改spp\_profinet.h文件，定义初始化函数原型void SetupProfinet();



1. 修改spp\_profinet.c文件
2. 修改引用的头文件 #inlcude “spp\_template.h”改为 #inlcude “spp\_profinet.h”
3. SetupProfinet()函数即为模板中的函数SetupTemplate()，用来调用注册预处理器函数RegisterPreprocessor，修改为

void SetupProfinet()

{

#ifndef SNORT\_RELOAD

RegisterPreprocessor("Profinet\_RT", ProfinetRTInit);

#else

RegisterPreprocessor("Profinet\_RT", ProfinetRTInit, ProfinetRTReloadFuction,NULL, NULL, NULL);

#endif

}

Profinet\_RT为预处理器名称

ProfinetRTReloadFuction这个函数可以暂时为NULL，本例中也只是声明了这个函数，并没有使用。

1. ProfinetRTInit()函数即为模板中TemplateInit()函数。用来初始化预处理器，调用AddFuncToPreprocList把此预处理器加入预处理器列表。

static void ProfinetRTInit(struct \_SnortConfig \*sc,u\_char \*args)

{

AddFuncToPreprocList(sc, ProfinetRTFunct,0x01, PP\_Profinet\_RT, PROTO\_BIT\_\_PROFINET);

printf("ProfinetRTInit is setup\n\n");

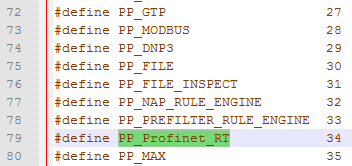
RegisterPreprocStats("Profinet\_RT", ProfinetPrintStats);

//AddFuncToCleanExitList(PreprocCleanExitFunction, NULL);

//AddFuncToRestartList(PreprocRestartFunction, NULL);

}

* ProfinetRTFunct为预处理器执行函数 0x01为预处理器优先级
* PP\_Profinet\_RT需在源文件 /src/preprocids.h中定义的最后添加(把PP\_MAX增加了1)



* PROTO\_BIT\_\_PROFINET为本项目中对新增的解码类型自己定义的标志。一般情况下想对IP数据进行预处理，使用标志PROTO\_BIT\_\_IP即可，这个数值定义在Decode.h文件中。**不要使用**PROTO\_BIT\_\_PROFINET**这个参数。**
* RegisterPreprocStats为Snort退出时预处理器输出总结信息的模块添加函数，即添加到此函数中的函数ProfinetPrintStats()会在程序结尾执行一次。不需要总结输出的话可以注释掉
* args为预处理器的配置信息

1. ProfinetRTFunct()函数，自由发挥。对每一个数据包的处理可以写在此函数中
2. PreprocCleanExitFunction PreprocRestartFunction未修改
3. 修改snort-2.9.8.2/src/preprocessors/Makefile.am文件

把此处理器的文件添加到libspp\_a\_SOURCES中去 spp\_profinet.c spp\_profinet.h \

1. 回到snort-2.9.8.2路径下automake自动修改makefile（src/preprocessors/路径下）
2. 重新make、 make install
3. 修改snort.conf文件，即启动snort IDS模式时-c 后面的那个文件

在Setp 5中添加

preprocessor Profinet\_RT

即启动Profinet\_RT预处理器

1. Over

备注：

1. 想要启动预处理器报警需修改/etc/snort/snort.conf文件，Step8

# include $PREPROC\_RULE\_PATH/preprocessor.rules 去掉前面的井号

1. 定制自己的预处理器报警需在/etc/snort/preproc\_rules/preprocessor.rules中添加报警规则，例如

alert ( msg: "Profinet Test 1"; sid:**1**; gid:146; rev: 1; metadata: rule-type preproc; classtype:protocol-command-decode; )

每一个预处理器使用一种gid

预处理器的每一种报警信息，使用一个sid

1. 预处理器中调用

SnortEventqAdd(146, **1**, 1, 0, 0,"aaaaaaaaaaaaaaaaaaaaaaaaaaaaa", 0);即可

1. 最好在/etc/snort/gen-msg中添加对应的预处理器与报警信息的对应信息，便于参考