SGX开发实验

TEE

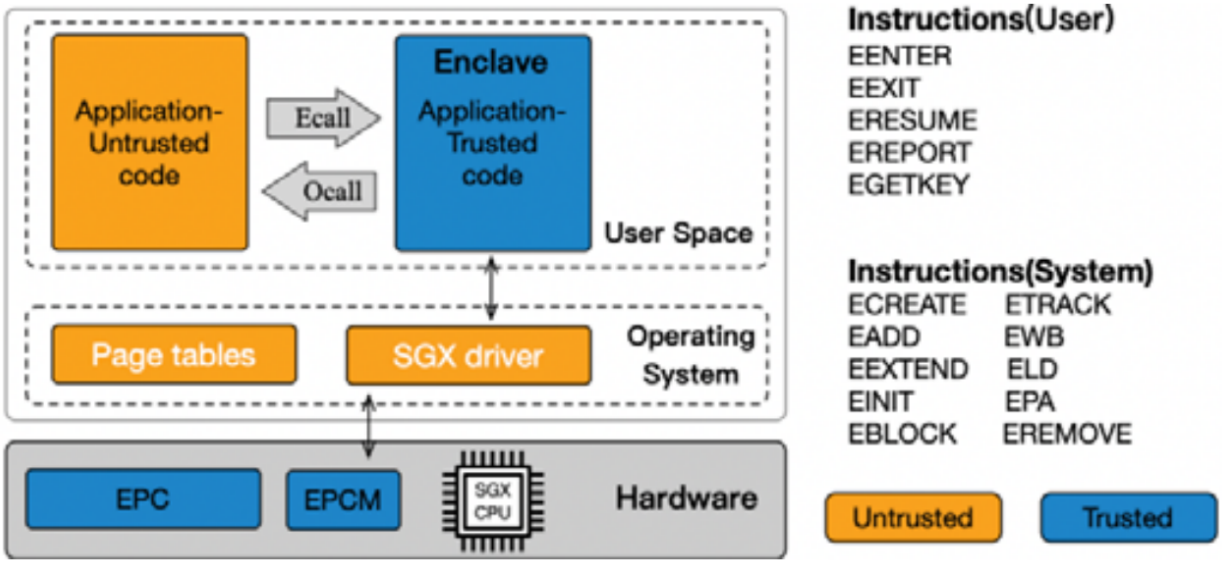
可信执行环境是计算平台上的一个安全区域，其最主要提供的一个属性是隔离，保护安全区域内部的数据代码的机密性和完整性。目前各个架构上都有实现自己的可信执行环境，如Intel的SGX，AMD的SEV，ARM的TrustZone等。

Intel SGX

Intel SGX（software guard extensions） 是Intel推出的指令集扩展，是基于内存和证明的硬件机制，为需要保护的应用程序提供了可信的执行环境Enclave，将应用程序与外部环境隔离开来，保障了应用程序的关键代码和敏感数据的机密性和完整性。Enclave本质上是一块被保护的物理内存区域，所有的Enclave代码和数据都存储在EPC（enclave page cache）中，Enclave中的代码数据在写入内存前，会使用处理器内存加密单元加密。

官方网址：https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

Github项目：https://github.com/intel/linux-sgx



SGX指令集添加了17条新的指令 ,其中12条由系统使用，5条由用户使用。常用的指令包括进入Enclave中执行代码（EENTER），退出Enclave（EEXIT），获取一个128-bit的

key（EGETKEY）等。

Super.	Description	User	Description
EADD	Add a page	EENTER	Enter an enclave
EBLOCK	Block an EPC page	EEXIT	Exit an enclave
ECREATE	Create an enclave	EGETKEY	Create a cryptographic key
EDBGRD	Read data by debugger	EREPORT	Create a cryptographic report
EBDGWR	Write data by debugger	ERESUME	Re-enter an enclave
EINIT	Initialize en enclave		
ELDB	Load an EPC page as blocked		
ELDU	Load an EPC page as unblocked		
EPA	Add a version array		
EREMOVE	Remove a page from EPC		
ETRACE	Activate EBLOCK checks		
EWB	Write back/invalidate an EPC page		

SGX SDK

Intel官方为了简化SGX应用程序的开发，提供了SGX SDK，SDK最主要的功能是实现了ECALL和OCALL这两个接口，ECALL准备执行环境并调用EENTER指令进入Enclave，用于Enclave外的不可信代码调用Enclave里的代码；OCALL执行EEXIT指令返回到不安全的环境下执行，用于Enclave里的代码调用Enclave外的代码。

Enclave/Enclave.edl

EDL文件中声明了在Enclave中使用的ECALL和OCALL接口，trusted部分是ECALL，untrusted部分是OCALL，语法类似于C语言的语法。在传输buffer时需要在参数前指明传输的方向和数据的长度。

- 方向：在ECALL中in指数据从Enclave外传到Enclave里，out指Enclave里传到Enclave外（在OCALL中相反）。
- 长度：通过size和count指定，size指定一个单位的数据结构的大小，可以是一个整型常量也可以是函数参数；count指定传输的单位数，默认为1；lenth = size * count

需要注意的是ECALL和OCALL的返回值通过第一个参数传递。

数据类型					
char	short	int	float	double	void
int8_t	int16_t	int32_t	int64_t	size_t	wchar_t
uint8_t	int16_t	int32_t	int64_t	unsigned	struct
union	enum	long			
指针参数处理					
in	out	user_check	count	size	readonly
isptr	string	wstring			
其他					
enclave	from	import	trusted	untrusted	include
public	allow	isary	const	propagate_errno	transition_using_threads
函数调用约定					
cdecl	stdcall	fastcall	dllimport		

```
enclave {
    trusted {
        public void ecall_hello_from_enclave([out, size=len] char* buf, size_t len);
    };
};
```

Enclave/Enclave.cpp（可信代码）

```
#include "Enclave.h"
#include "Enclave_t.h"
#include <string.h>

void ecall_hello_from_enclave(char *buf, size_t len)
{
    char *hello = malloc(len);
    read(0,hello,len);
    size_t size = len;
    if(strlen(hello) < len)
    {
        size = strlen(hello) + 1;
    }
    memcpy(buf, hello, size - 1);
```

```
    buf[size-1] = '\0';
}
```

App/App.cpp（不可信代码）

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <unistd.h>
#include <pwd.h>
#define MAX_PATH FILENAME_MAX
#include "sgx_urts.h"
#include "App.h"
#include "Enclave_u.h"
/* Global EID shared by multiple threads */
sgx_enclave_id_t global_eid = 0;

int initialize_enclave(void)
{
    sgx_status_t ret = SGX_ERROR_UNEXPECTED;
    /* 调用 sgx_create_enclave 创建一个 Enclave 实例 */
    /* Debug Support: set 2nd parameter to 1 */
    ret = sgx_create_enclave(ENCLAVE_FILENAME, SGX_DEBUG_FLAG, NULL, NULL, &global_eid, NU
LL);
    if (ret != SGX_SUCCESS) {
        printf("Failed to create enclave, ret code: %d\n", ret);
        return -1;
    }
    return 0;
}

/* 应用程序入口 */
int SGX_CDECL main(int argc, char *argv[])
{
    (void)(argc);
    (void)(argv);
    const size_t max_buf_len = 100;
    char buffer[max_buf_len] = {0};

    /* 创建并初始化 Enclave */
    if(initialize_enclave() < 0){
        printf("Enter a character before exit ...\n");
        getchar();
        return -1;
    }

    /* ECALL 调用 */
    ecall_hello_from_enclave(global_eid, buffer, max_buf_len);
    printf("%s\n", buffer);

    /* 销毁 Enclave */
```

```
    sgx_destroy_enclave(global_eid);
    printf("Info: SampleEnclave successfully returned.\n");
    printf("Enter a character before exit ...\n");
    getchar();
    return 0;
}
```

实验要求

在SGX里实现RC4加密算法，设计三个ECALL函数，分别为S盒生成，流密钥生成，解密函数，其中key作为全局变量存放在Enclave中，密文通过ECALL从App传入Enclave中解密，将解密完成的结果通过OCALL输出。

密文HEX：1c7b53616e81ce8a45e7af3919bc94aba41258

key：gosecgosec

可以基于/home/test/sgxsdk/SampleCode/SampleEnclave目录下的例子开发。

编译之前需要执行命令 source /home/test/sgxsdk/environment

使用make SGX_MODE=sim编译项目。