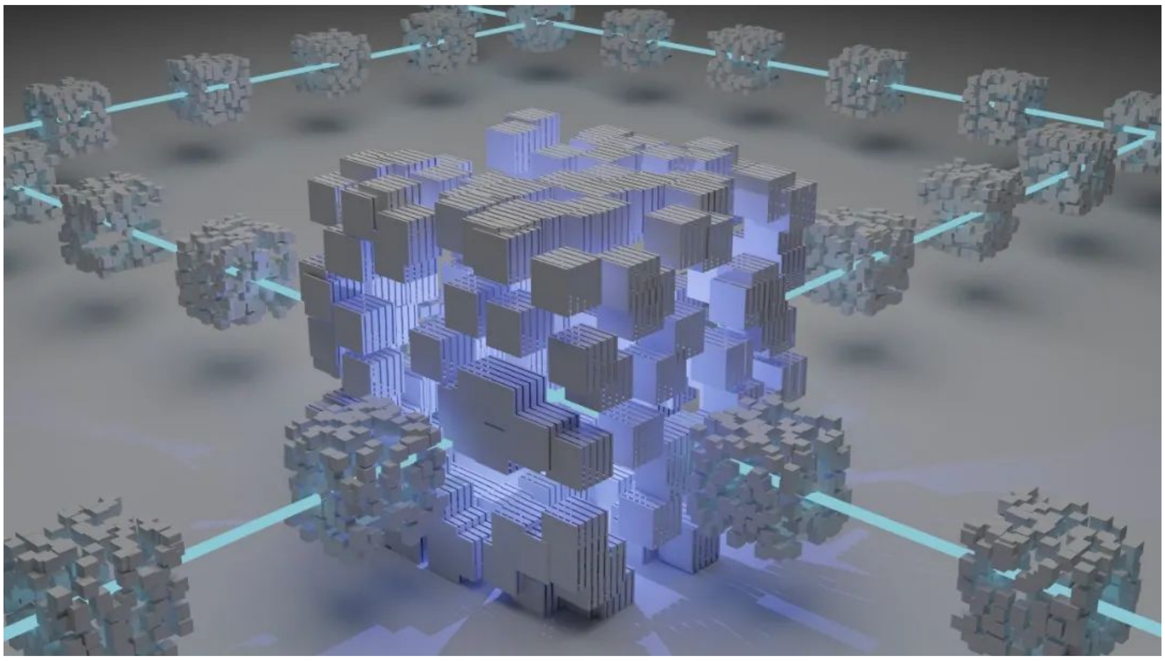
Anti-token保护分析心得

原创 那年没下雪 看雪学苑 2022-06-10 18:24 发表于上海

收录于合集
#“雪花”创作激励计划 78个

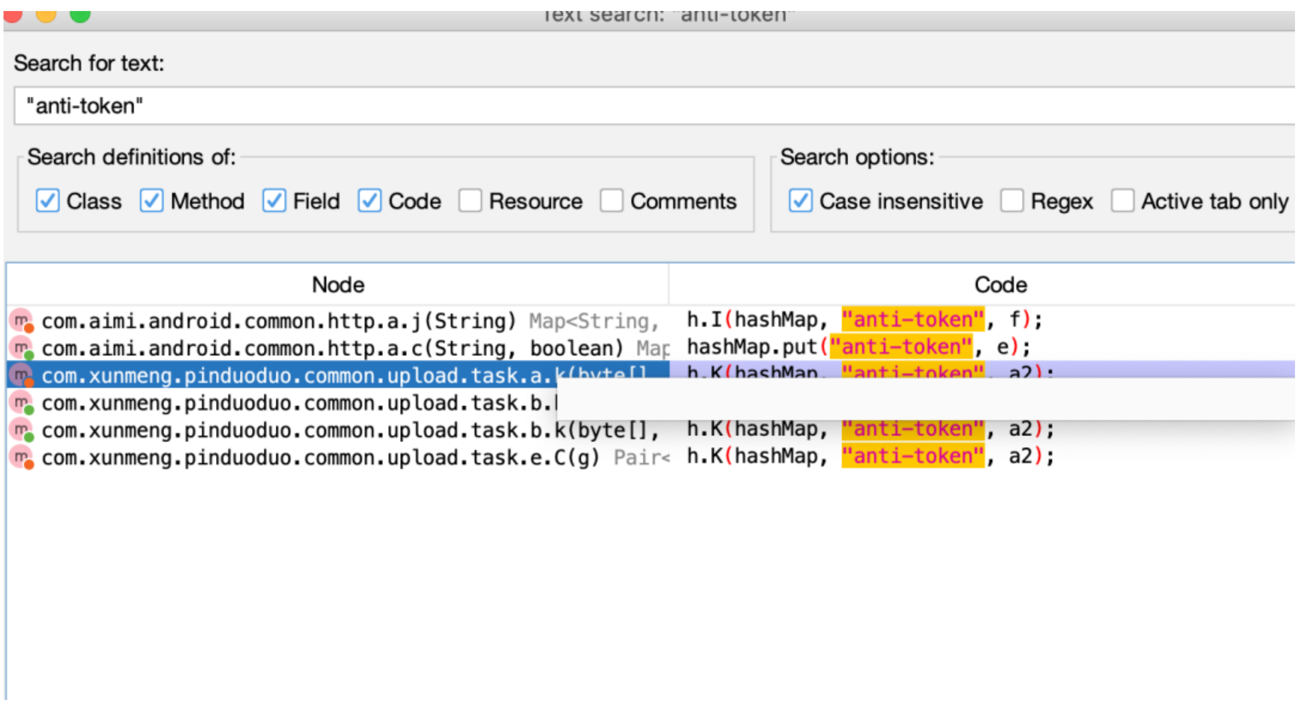


本文为看雪论坛优秀文章

看雪论坛作者ID：那年没下雪

某软件charles抓包分析发现跟商品相关的请求头里都带了一个anti-token的字段且每次都不一样，那么下面的操作就从分析anti-token开始了。

1.jadx反编译直接搜索



Text search: "anti-token"

Search for text:

"anti-token"

Search definitions of: [✓ Class] [✓ Method] [✓ Field] [✓ Code] ☐ Resource ☐ Comments

Search options: [✓ Case insensitive] ☐ Regex ☐ Active tab only

Node	Code
com.aimi.android.common.http.a.j(String) Map<String,	h.I(hashMap, "anti-token", f);
com.aimi.android.common.http.a.c(String, boolean) Map	hashMap.put("anti-token", e);
com.xunmeng.pinduoduo.common.upload.task.a.k(byte[]	h.K(hashMap, "anti-token", a2);
com.xunmeng.pinduoduo.common.upload.task.b.l	
com.xunmeng.pinduoduo.common.upload.task.b.k(byte[],	h.K(hashMap, "anti-token", a2);
com.xunmeng.pinduoduo.common.upload.task.e.C(g) Pair<	h.K(hashMap, "anti-token", a2);

选中跟http相关的类对这个方法进行打印堆栈。

结合堆栈方法调用的情况找到具体anti-token是由拦截器类f.a方法调用的,在http.a.c()方法中生成并且http.p.e()方法中加入请求头。

在http.a.c()方法中有个一个判断条件如果为true则走d.a().e()方法生成anti-token。

如果为false则走j()方法生成anti-token。

hook这个i()方法返回值可知获取商品详情接口返回值为false所以走的是j()方法进行计算anti-token。

SecureNative.deviceInfo3()方法生成,传入的str为pdd生成的固定id 一个字符串。

根据hook_libart 得到info3()方法是在libodd_secure.so中,那么ida打开看看这个so包。

2.这部分我们采用unidbg+jnitrace+frida相结合的方式

unidbg前期准备的代码这里就不发了直接调用这个info3方法。

这里提示调用gad()方法返回一个字符串那么frida hook这个方法拿到这个值 如下图 一个固定的字符串。

16位长度看着像AES的密钥。

继续补环境 这里简单补环境就不发了。

补完简单的环境代码后,再次运行报这个错误 看错误应该是缺少文件 ,那么看看日志需要补那个文件。

继续运行,没有返回值报空指针。execve()函数执行的时候程序exit了这里我们返回对象本身。

execve()函数执行的时候程序exit了。

execve filename=/system/bin/sh, args=[sh, -c, cat /proc/sys/kernel/random/boot_id]
这个函数相当于查看 boot_id这个文件信息。

捋顺下逻辑应该就是先fork进程 然后在子进程中读取这个文件 然后把他写入pip中。

那么自定义syscallhandler后 再次运行成功拿到结果。

全部代码如下：

```java
package pdd;

import com.github.unidbg.AndroidEmulator;
import com.github.unidbg.Emulator;
import com.github.unidbg.Module;
import com.github.unidbg.file.FileResult;
import com.github.unidbg.file.IOResolver;
import com.github.unidbg.file.linux.AndroidFileIO;
import com.github.unidbg.linux.android.AndroidARMEmulator;
import com.github.unidbg.linux.android.AndroidEmulatorBuilder;
import com.github.unidbg.linux.android.AndroidResolver;
import com.github.unidbg.linux.android.dvm.*;
import com.github.unidbg.linux.file.ByteArrayFileIO;
import com.github.unidbg.memory.Memory;
import com.github.unidbg.memory.SvcMemory;
import com.github.unidbg.spi.SyscallHandler;
import com.github.unidbg.unix.UnixSyscallHandler;

import java.io.File;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

public class Pddmain extends AbstractJni implements IOResolver<AndroidFileI

```

```
    private AndroidEmulator androidEmulator;
    private static final String APK_PATH = "/Users/Downloads/com.xunmeng.pi
    private static final String SO_PATH = "/Users/Downloads/com.xunmeng.pin
    private Module moduleModule;
    private VM dalvikVM;

    public static void main(String[] args) {
        Pddmain main = new Pddmain();
        main.create();
    }

    private void create() {
        AndroidEmulatorBuilder androidEmulatorBuilder = new AndroidEmulator
            @Override
            public AndroidEmulator build() {
                return new AndroidARMEmulator("com.xunmeng.pinduoduo",rootD
                    @Override
                    protected UnixSyscallHandler<AndroidFileIO> createSysca
                        return new PddArmSysCallHand(svcMemory);
                    }
                };
            }
        };
        androidEmulator = androidEmulatorBuilder.setProcessName("").build()
        androidEmulator.getSyscallHandler().addIOResolver(this);
        Memory androidEmulatorMemory = androidEmulator.getMemory();
        androidEmulatorMemory.setLibraryResolver(new AndroidResolver(23));
        dalvikVM = androidEmulator.createDalvikVM(new File(APK_PATH));
        DalvikModule module = dalvikVM.loadLibrary(new File(SO_PATH), true)
        moduleModule = module.getModule();
        dalvikVM.setJni(this);
        dalvikVM.setVerbose(true);
        dalvikVM.callJNI_OnLoad(androidEmulator, moduleModule);
        callInfo3();
    }

    @Override
    public void callStaticVoidMethodV(BaseVM vm, DvmClass dvmClass, String 
        if ("com/tencent/mars/xlog/PLog->i(Ljava/lang/String;Ljava/lang/Str
            return;
        }
        super.callStaticVoidMethodV(vm, dvmClass, signature, vaList);
    }

    private void callInfo3() {
        List<Object> argList = new ArrayList<>();
        argList.add(dalvikVM.getJNIEnv());
        argList.add(0);
        DvmObject<?> context = dalvikVM.resolveClass("android/content/Conte
        argList.add(dalvikVM.addLocalObject(context));
        argList.add(dalvikVM.addLocalObject(new StringObject(dalvikVM, "api/
        argList.add(dalvikVM.addLocalObject(new StringObject(dalvikVM, "dIr
        Number number = moduleModule.callFunction(androidEmulator, 0xb6f9, a
        String toString = dalvikVM.getObject(number.intValue()).getValue().t
        System.out.println(toString);
    }

    @Override
    public DvmObject<?> callStaticObjectMethodV(BaseVM vm, DvmClass dvmClas
        if ("com/xunmeng/pinduoduo/secure/EU->gad()Ljava/lang/String;".equal
            return new StringObject(vm, "cb14a9e76b72a627");
        } else if ("java/util/UUID->randomUUID()Ljava/util/UUID;".equals(sig
            UUID uuid = UUID.randomUUID();
            DvmObject<?> dvmObject = vm.resolveClass("java/util/UUID").newOb
            return dvmObject;
        }
        return super.callStaticObjectMethodV(vm, dvmClass, signature, vaList
    }

    @Override
    public DvmObject<?> callObjectMethodV(BaseVM vm, DvmObject<?> dvmObject,
        if ("java/util/UUID->toString()Ljava/lang/String;".equals(signature)
            UUID uuid = (UUID) dvmObject.getValue();
            return new StringObject(vm, uuid.toString());
        } else if ("java/lang/String->replaceAll(Ljava/lang/String;Ljava/lan
            String obj = dvmObject.getValue().toString();
            String arg0 = vaList.getObjectArg(0).toString();
            String arg1 = vaList.getObjectArg(1).toString();
            String replaceAll = obj.replaceAll(arg0, arg1);
            return new StringObject(vm, replaceAll);

        }
        return super.callObjectMethodV(vm, dvmObject, signature, vaList);
    }

    @Override
    public int callIntMethodV(BaseVM vm, DvmObject<?> dvmObject, String sigr
        if ("java/lang/String->hashCode()I".equals(signature)) {
            return dvmObject.getValue().toString().hashCode();
        }
        return super.callIntMethodV(vm, dvmObject, signature, vaList);
    }

    @Override
    public FileResult<AndroidFileIO> resolve(Emulator<AndroidFileIO> emulatc
        if ("/proc/stat".equals(pathname)) {
            String info = "cpu  15884810 499865 12934024 24971554 59427 3231
                    "cpu0 6702550 170428 5497985 19277857 45380 1821584 5294
                    "cpu1 4438333 121907 3285784 1799772 3702 504395 255852
                    "cpu2 2735453 133666 2450712 1812564 4626 538114 93763 (
                    "cpu3 2008473 73862 1699542 2081360 5716 367109 66860 0
                    "intr 1022419954 0 0 0 159719900 0 16265892 4846825 5 5
                    "ctxt 1572087931\n" +
                    "btime 1649910663\n" +
                    "processes 230673\n" +
                    "procs_running 6\n" +
                    "procs_blocked 0\n" +
                    "softirq 374327567 12481657 139161248 204829 7276312 227
            return FileResult.success(new ByteArrayFileIO(oflags, pathname,
        }
        return null;
    }
}
```

看雪ID：那年没下雪

https://bbs.pediy.com/user-home-884888.htm https://bbs.pediy.com/user-home-884888.htm

*本文由看雪论坛 那年没下雪 原创，转载请注明来自看雪社区

# 往期推荐

1.反射式DLL注入实现

2.angr符号变量转LLVM IR

3.记录一次对某CMS漏洞挖掘

4.逆向角度看C++部分特性

5.CVE-2014-4113提权漏洞学习笔记

6.Go语言模糊测试工具：Go-Fuzz

球分享 球点赞 球在看

点击“阅读原文”，了解更多！

收录于合集 #“雪花”创作激励计划 78

下一篇 · STM32固件逆向

阅读原文

喜欢此内容的人还喜欢

STM32固件逆向
看雪学苑