Anti-cheating Engine for Android

Unknown macro: 'html'

Excuse the ads! We need some help to keep our site up. Unknown macro: 'html'

List

- Anti-cheating Engine for Android
  - Check modification of binaries
  - Check debug
  - Check Speed hacking
  - Check Virtual Machine
  - Related site

Anti-cheating Engine for Android

- 2016 Anti-cheating Engine for Android .
  - https://github.com/Lazenca/Lazenca-S
- .
- Cheat tool .

Check modification of binaries

- .
  - Hash Hash .
- .
  - , .
    - Reverse engineering .
  - Hash CRC .

CheckBinaryModification.h

```
bool ModFindHash(char *hash) {

        long lSize;
        size_t result;
        int result_val;

        char *buffer;
        char patternFile[MAX_PATH] = "/mnt/sdcard/pattern";

        FILE *fp = fopen(patternFile, "rb");

        if (fp) {
                fseek(fp, 0, SEEK_END);
                lSize = ftell(fp);
                rewind(fp);

                buffer = (char*) malloc(sizeof(char) * lSize);
                if (buffer == NULL) {
                        fputs("Memory error", stderr);
                        exit(2);
                }

                result = fread(buffer, 1, lSize, fp);
                if (result != lSize) {
                        fputs("Reading error", stderr);
                        exit(3);
```

```
                }

                if (result > 0) {
                        if(strstr(buffer,hash)){
                                result_val = true;
                        }else{
                                result_val = false;
                        }
                }
                fclose(fp);
                free(buffer);
        } else {
                return false;
        }
        return result_val;
}

void CheckModBinary(){
        pid_t pid= getpid();

        char sha256[65] = "";

        char packageName[MAX_PATH];
        char apkFilePath[MAX_PATH];
        char libFilePath[MAX_PATH];
        char tmpLibFilePath[MAX_PATH];

        int zipcount = 0;
        unsigned int crc;

        DbgPrint("Modification Check");

        getCmdline(pid,packageName,sizeof(packageName));
        getApkFilePath(packageName,&apkFilePath);

        sprintf(libFilePath, "/data/data/%s/lib", packageName);

        while(1)
        {
                fileToSha256(&sha256,apkFilePath);

                if(ModFindHash(sha256) == false)
                {
                        DbgPrint("File Path : %s, SHA256 : %s - Modfication",apkFilePath,sha256);
                        DbgPrint("Modfication.");
                        sleep(10);
                        exit(0);
                }

                DIR *dirInfo;
                struct dirent *dirEntry;

                dirInfo = opendir(libFilePath);
                if (NULL != dirInfo) {
                        while (dirEntry = readdir(dirInfo))
                        {
                                sprintf(tmpLibFilePath, "/data/data/%s/lib/%s", packageName,dirEntry->d_name);

                                fileToSha256(&sha256,tmpLibFilePath);
                                if (ModFindHash(sha256) == false) {
                                        DbgPrint(".SO File Hash Check - Modfication.");
                                        sleep(10);
                                        exit(0);
                                }
                        }
                        closedir(dirInfo);
                }
                sleep(1000);
        }
}
```

Check debug

- .
  - **"/proc/pid/cmdline" .**
  - GDB PPID gdb "cmdline" gdb .

bool isCheckCmdline()

```
bool isCheckCmdline() {
   char filePath[32], fileRead[128];
   FILE* file;

   snprintf(filePath, 24, "/proc/%d/cmdline", getppid());
   file = fopen(filePath, "r");

   fgets(fileRead, 128, file);
   fclose(file);

   if(!strcmp(fileRead, "gdb")) {
           DbgPrint("Debugger(gdb) detected\n");
       return true;
   }
   DbgPrint("Clear(Debug)\n");
   return false;
}
```

  - **"/proc/pid/status" .**
  - status , , , .
  - Debug "TracerPid" .
    - PID .
    - '0' .

bool isCheckTracerPid()

```
bool isCheckTracerPid() {
        int TPid;
        char buf[512];

        const char *str = "TracerPid:";
        size_t strSize = strlen(str);

        FILE* file = fopen("/proc/self/status", "r");

        while (fgets(buf, 512, file)) {
                if (!strncmp(buf, str, strSize)) {
                        sscanf(buf, "TracerPid: %d", &TPid);
                        if (TPid != 0) {
                                DbgPrint("Debugger detected\n");
                                return true;
                        }
                }
        }
        fclose(file);
        DbgPrint("Clear(Debug)\n");
        return false;
}
```

Check Speed hacking

- **Memory cheat Speed hack (gettimeofday, ...) .**
- **Speed hack .**
  - (start time) .
  - sleep() .
  - sleep() (end time) .

        - (end time - start time) sleep() .
        - sleep() Speed hack .
        - Ex)
            - gettimeofday() sleep(100) 101000 10000 .
            - clock_gettime() sleep(100) 100001000 100000000 .
- **Memory cheat .**

CheckSpeedHack.h

```
int getTime1()
{
        struct timeval tv;
        struct timezone tz;
        gettimeofday (&tv, &tz);
        return (tv.tv_sec*1000 + tv.tv_usec/1000);
}

long getTime2()
{
        struct timespec now;
        clock_gettime(CLOCK_MONOTONIC,&now);
        return now.tv_sec*1000000 + now.tv_nsec/1000;
}

void CheckSpeedHack(){
        int start = 0, end = 0, time_data = 0;
        int start2 = 0, end2 = 0, time_data2 = 0;
        int cmp=1;

        while(cmp){
                start = getTime1();
                start2 = getTime2();

                sleep(100);
                end = getTime1();
                end2 = getTime2();

                time_data = end - start;
                time_data2 = end2 - start2;
                DbgPrint("time_data : %d, end : %d , start : %d",time_data,end,start);
                DbgPrint("time_data2 : %d, end : %d , start : %d",time_data2,end2,start2);
                if(time_data > 101000 || time_data < 10000){
                        cmp = 0;
                        DbgPrint("SpeedHackDetect");
                        sleep(10);
                        exit(0);
                }
                if(time_data2 > 100001000 || time_data2 < 100000000){
                        cmp = 0;
                        DbgPrint("SpeedHackDetect");
                        sleep(10);
                        exit(0);
                }
        }
}
```

Check Virtual Machine

- **Virtual Machine .**
    - "/system/build.prop" .
    - "/system/build.prop" Android .
    - Virtual Machine Virtual Machine , Virtual Machine .
        - "ro.product.manufacturer"
        - "ro.product.model"
        - "ro.product.name"
        - "ro.product.device"
    - **.**
- **.**
    - Virtual Machine

- Virtual Machine
-

CheckVirtualMachine.h

```
void virtualMachine(char *filePath, char *searchStr,char *findStr){
        FILE *fp;
        char str[81];
        fp = fopen(filePath, "r");

        while(!feof(fp))
        {
                fgets(str, 80, fp);
                if(strstr(str,searchStr) != NULL){
                        str[strlen(str)-1] = ',';
                        strcat(findStr,str);
                }
        }
        fclose(fp);
}

bool IsBlackDevice(char *deviceName)
{
        if(strstr(deviceName,"BlueStacks") != NULL)
        {
                DbgPrint(" BlueStacks : %s,%p",deviceName,deviceName);
                return false;
        }

        if(strstr(deviceName,"Genymotion") != NULL)
        {
                DbgPrint(" Genymotion : %s,%p",deviceName,deviceName);
                return false;
        }
        return true;
}

void CheckVirtualMachine(){
        char findSearch[300];

        virtualMachine("/system/build.prop","ro.product.manufacturer",&findSearch);
        virtualMachine("/system/build.prop","ro.product.model",&findSearch);
        virtualMachine("/system/build.prop","ro.product.name",&findSearch);
        virtualMachine("/system/build.prop","ro.product.device",&findSearch);

        if(IsBlackDevice(findSearch))        {
                DbgPrint(" BlueStacks : Not Detect");
        }else{
                DbgPrint(" BlueStacks : Detect");
                sleep(10);
                exit(0);
        }
}
```

Related site

- https://github.com/Lazenca/Lazenca-S

Unknown macro: 'html'