Short review of modern vulnerability research

Michal Bucko
michal.bucko@eleytt.com
Warsaw, 2008

Abstract

This paper introduces quite a new approach to modern threat management in the context of advanced vulnerability research. The author makes an attempt to point out security threats and highlight possible exploitation opportunities as well as into describes the proposed mitigation and prevention system which would allow deployment of security QoS. Security trends, new weaknesses and vulnerabilities are all analyzed in the context of global threat management system make the work pinpoint the real-world problem and suggest the solution.

Table of Contents

1 Introduction to the problem

1.1 Cutting-edge traditional vulnerabilities and the trends

Nowadays, the Internet security reaches such a point that risk assessment is very important. Zero day vulnerabilities exploited in the wild, malware, rootkits – those problems are definitely not going to disappear. Moreover, all of the problems concerning these issues are going to exacerbate. Mobile security, business web application security, mobile network security are just other targets for the future exploitation. Each day new malware comes to this world and numerous unknown vulnerabilities are exploited in the wild. Online e-facilities and overall application of electronic solutions result in building stronger connection between humanity and electronic world, which more and more accurately defines current notion of every-day living. The electronic world is, however, not secure (and prone to weaknesses) and this will be exploited mercilessly. Criminals may hide inside the network and earn money the way they always wanted, but much faster as the information world is getting smaller and smaller.

Global network does not provide security quality of service and application of global security management would be an interesting mitigation and protection. The prognosis for the security market is appealing as IT security is currently undergoing a real blossoming due to the fact that our world is slowly turning into even a more volatile electronic world, where everyone uses a mobile phone and takes advantage of online electronic facilities. The trends are clear – new merges and acquisitions in the field of IT security, malware boost and new kinds of weaknesses and vulnerabilities. The legislation won't be capable of handling effectively electronic crimes, therefore unexpected judgment may occur. Mobile phones and embedded devices are going to be put under stress conditions by hackers. It may be possible that telecom infrastructure won't soon remain

untouched. The problems with identity theft, e-voting, digital signatures and SCADA systems in conjunction with the mobility factor are definitely going to shape the IT world.

The techniques involved in malware analysis, risk management and protection must be developed to become more effective. Moreover, the overall mobility enforces the requirement for easier and standardized risk identification and handling, thus, the proposal of global security and threat management network. Such a global threat management system would allow the standardization and SQoS deployment in global networks (note that there are programs such as EuQoS, which concern the QoS enforcement but have nothing to do with security). Security level's improvement and standardized security measurement would allow better risk management. Risk management involves knowledge of new threats and vulnerabilities. This is the reason why an introduction to advanced vulnerability research is covered in this paper.

1.2 Cutting-edge knowledge and new kinds of weaknesses

We already see the problems related to the exploitation of traditional weaknesses and vulnerabilities. We, however, have not taken into consideration the rapid technological development and the aspect of mobility. New kinds of vulnerabilities and weaknesses are going to be exploited and new mitigation and prevention measures shall be used. The attackers obtain high level of knowledge and may perform successful attacks at critical systems. New weaknesses and vulnerabilities may allow more focused and specialized attacks, e.g. the mobility factor may allow tracking, jamming, and eavesdropping whereas an advanced attack at the telecom infrastructure may result in severe service disruption. Moreover, the severity of the attack (especially today when the telecommunications is that expensive) is not only dependent on the knowledge

differences, but also on the fact that deployment of security solutions is very costly and slows down the network (e.g. communication between RLC and Node-B may be encrypted, but generally remains plain text, which allows MITM). Knowing the trends and analyzing the current situation, one can quite easily notice that new kinds of weaknesses are going to be exploited in the near future. This means new risks and, as many times mentioned, we are now facing the era of electronic crime and cyber terrorism. The mitigation and prevention lies in the foundation of the project described within this paper.

The paper also covers an introduction to advanced vulnerability research. The examples and know-how described in the paper are based on hands-on experience and research. The techniques involved in vulnerability assessment and analysis shown in the article may help disclose unknown weaknesses in popular systems as well as enable the readers to conduct further research and discover new kinds of system weaknesses. As mentioned in the paragraph's title, an in-depth knowledge of attacked OSes as well as exploitation techniques (which are far beyond the scope of this paper) is required to conduct most advanced attacks.

2 Vulnerability research

2.1 Current technical aspects of vulnerability research

Nowadays, the vulnerability research process involves a wide variety of procedures and knowledge of vulnerability assessment and exploitation is broad. The discovery of the vulnerability requires knowledge of reverse engineering, vulnerabilities (buffer overflows, format string vulnerabilities, race conditions, NULL-pointer dereference etc.), software penetration (how to conduct the testing to assess the vulnerability, how to apply fuzzing or manual testing, also knowledge of system internals is required to understand certain vulnerabilities). The final part

in the process is the exploitation, which requires significant know-how in the field of stack smashing protections and bypassing techniques.

Buffer overflow vulnerabilities are currently not only simply related to string-copying facilities. There are many functions, which may lead to memory corruption conditions, and therefore to exploitable vulnerabilities. It shall be borne in mind that detection of weaknesses cannot be currently fully automated due to required intelligence level. Buffer overruns occur in often unexpected and hard to verify conditions. For instance, the use of a standard non-compliant function used within the application could under some circumstances result in the unexpected behavior of the very application.

The process of vulnerability research may involve fuzzing, reverse engineering, source code auditing etc. There is, however, still much space for the discovery of much more sophisticated vulnerabilities. The vulnerability research process might also involve protocol analysis, research into operating systems' internals and much more. In this paragraph, I am going to give a few short examples of the technical aspect of vulnerability research related to the traditional system penetration, which allows the compromise of the remote machine. It must be noted that the exploitation techniques won't be discussed in this paper. Some of the vulnerabilities shown may be difficult to exploit under various systems. There is a number of known anti-exploitation solutions as well as the exploitation of different vulnerabilities varies significantly.

Two pictures below show how improperly memory is allocated and how EIP can be simply overwritten to execute the arbitrary code remotely. The pictures are based on [15] and the research conducted using OllyDbg – this method includes the use of reverse engineering techniques. Anti-reverse engineering is skipped in this paper.

Figure 1, Use of memcpy ([15])

In the following picture (on the right), we can see that EIP has been overwritten. After the *memcpy* function has been used to copy the contents of the specially crafted file into the buffer of a fixed size (typical buffer overflow vulnerability), the exploitation seems quite simple.

Figure 2, EIP overwritten ([15])

Below we can see an even more typical example with *strcpy* function – no bounds-checking results in exploitable memory corruption conditions, which may be successfully exploited to conduct remote privilege escalation.

Figure 3, More typical buffer overflow ([15])

One can, however, find a number of examples where *strncpy* function is improperly used, thus arbitrary code execution may be possible.

Current vulnerability research is, however, far away from the situation when we dealt with typical buffer overruns. There are many vulnerabilities, which may be exploited to achieve system compromise. No longer do we look for the known functions, but also for other kinds of vulnerabilities, such as: race conditions, format string vulnerabilities, NULL-pointer dereference weaknesses, cross-zone and other privilege escalation vulnerabilities.

There are currently many known issues, which may result in severe security-related problems:

- lack of standard compliance (e.g. AfxParseUrl),
- facilities that require changing the state to the elevated or less privileged mode
- path spoofing vulnerability (e.g. basename)
- functions prone to problems with TOCTOU (e.g. chmod, chown)
- dangerous functions when passing parametres (e.g. BytesToUnicode) or requiring additional knowledge (e.g. chroot, CreateProcess, CreateThread)
- problems with format strings (e.g. CHString)
- functions, which may lead to unhandled exceptions (e.g. EnterCriticalSection)
- NULL-termination issues with e.g. getc

- vulnerable to internal problems (e.g. some implementation of getopt were vulnerable to buffer overflow vulnerabilities)

The aforementioned problems are only a few examples, which may be discovered during the vulnerability research process. Security software is also often prone to a wide variety of weak protection of registry keys, weak logic problems, improper use of *LoadLibrary* etc. On other side, firewalls are tested using a variety of other methods (e.g. substitution, launching, DLL or process injection) but also may be prone to regular security weaknesses. There are also many applications, which are prone to the vulnerabilities strongly associated with web applications' security. For instance, a number of client applications is prone to SQL injection or directory traversal due to improper handling of given user-provided input.

The process of vulnerability research cannot be yet fully automated as this would require very sophisticated tools and built-in anomaly detection. Many of the attempts have been made (e.g. bugscam, zzuf, Sulley and Peach), but the process of vulnerability research still requires much analytical skills. For instance, auditing ActiveX controls have been made simpler due to the use of fuzzing tools available (still, there are many problems with this), whereas auditing of RPC interfaces still takes much hand-work and manual analysis. The exploitation of the vulnerabilities reachable via DCE/RPC endpoint (bound to certain services) requires enumeration of IDL stub (we could achieve this using MIDA plugin for IDA Pro, we could use Metasploit's module or python and impacket scripts) and, then, reverse engineering (or fuzzing) of the very function. Tipping Point have built an IDL parsing facility and an NDR library, which makes the work a little bit simpler. When it comes to auditing file parsing vulnerabilities, the fuzzing process is also deployed. The problem, however, remains as the fuzzer may not reveal a significant number of vulnerabilities and manual advanced reverse engineering may be required.

```
/* opcode: 0x1E, address: 0x76E3F6CA */

long  sub_76E3F6CA (
 [in][unique][string] wchar_t * arg_1,
 [in] /* enum16 */ short arg_2,
 [in] long arg_3,
 [out] struct struct_1 ** arg_4
);
```

Figure 4, Using MIDA in practice

Immunity Debugger and IDA Pro cut exploit development time as well as improve the effectiveness of vulnerability research. The example below (straight from Immunity's web site) shows how useful for security professionals can ImmDBG be.

Figure 5, Immunity Debugger and more effective vulnerability research

Stack- and heap-related problems can also be nicely analyzed using PaiMei Reverse Engineering Framework. Vulnerability research could also be made more effective using Sulley or Peach frameworks. Both allow the fuzzing of RPC, COM, DLLs, services, file formats.

Figure 6, Sulley and 41414141

Vulnerability research related to driver vulnerability assessment is currently confined to the driver's reverse engineering and simple fuzzing using simple home-made scripts (also Kartoffel from Reversemode could be used) with *CreateFileA* and *DeviceIoControl*. The exploitation is more difficult in real world – as shown by Digit-Labs in IPSecDrv.sys Local Kernel Ring0 Indirect System Call exploit. The vulnerability could be found using reverse engineering techniques. Very interesting examples of the exploitation have been shown in MS06-049, MS05-055 – both written by SoBeIt.

From the other side, the audit of the source code is still much beyond the capabilities of fuzzing tools available. As indicated before, the source code analysis is currently far away from simple string-copy vulnerabilities. There is a wide variety of vulnerabilities, which may be exploited by malicious attackers. Use of Valgrind, flawfinder or zzuf won't eliminate a significant number of vulnerabilities, although is strongly advised. The difficulty related to bug discovery varies - to begin with relatively simple local kernel denial of service vulnerabilities till quite elaborate exploits for race condition vulnerabilities. Source code audit requires an in-depth understanding of system's internals and typical vulnerabilities.

In this paragraph I am going to reveal a few examples of code auditing process, however, without any information about the specific product or vendors. The provided information is more a practical note than an introduction to the foundations of open-source auditing; given examples are real examples of vulnerabilities I have dealt with.

The problems often occur when we deal with multiple loops and an array, which is accessed each iteration. Many times the bug is difficult to detect, but could be used to cause remote out-of-the bounds access. Assuming that this is only a read-access kind of vulnerability, remote denial of service may be exploited (but this is not always the case as there are ways to take advantage of pointer dereference to execute shellcode). Write-access may result in arbitrary code execution.

When auditing source code at kernel level, one may notice that improper IOCTL handling may allow kernel panic, which leads to denial of service conditions. One may also find deeply hidden (non-obvious) integer overflow/underflow security issues, explore a race condtion vulnerability or detect a problem with the low-level

function. Source code audit requires the knowledge of typical problems with *unlink*, *chown*, *chmod*, *chroot*, *umask*, *syslog*, *setLocale* and many other functions. Typical vulnerabilities found during web application's security assessment are skipped as this has been described many times before. Web applications are prone to a variety of easily detectable vulnerabilities, e.g. (class) code injection, session issues (injection, fixation), CSRF and XSS, HTTP response splitting, directory traversal, improper error handling, improper privilege, context issues, .database security issues, etc. In conjunction with high kernel/user-mode exploitation, such vulnerabilities may help gain full remote system compromise.

Vulnerability assessment and exploitation requires an in-depth knowledge of OS internals – advanced kernel-level exploitation under Windows involves sophisticated knowledge of kernel pools (paged and non-paged), look-aside lists, advanced memory freeing and allocation whereas advanced exploitation under

Unix mostly requires knowledge of process handling, syscalls, singals etc. Automated source code security assessment involves the use of numerous tools, such as RATS, Valgrind or Flawfinder. Gprof or Gcov may also be useful during the automated analysis of the tested application.

Vulnerability research has been evolving very fast and right now it is time for its application to mobile security research. Mobile phones, smartphones, PDAs – the devices are prone to a number of foreseeable weaknesses, including vulnerabilities within the system installed on the embedded device, communication problems (including low-level issues) and much more. Even the hardware itself could be possibly affected (various aspects of embedded security analysis). For instance, analysis of Symbian code requires much knowledge of system's panic codes, including ViewSrv, USER and KERN-EXEC. When it comes to vulnerabilities in mobile applications, Symbian ones are handled quite nicely. KERN-EXEC informs of possible dereferences, HANG information provides information of possible infinite loops (usually), USER message reveals information of exceeding array boundaries, known "stray event" message is strongly related with asynchronous programming under Symbian etc.

Vulnerability research is not, however, confined to traditional security assessment- new kinds of vulnerabilities may also be disclosed. The attackers may attempt eavesdropping, masquerade, message modification, replay as well as traditional software vulnerability-based attacks- the exploitation of mobile infrastructure is probably inevitable. Knowing the architecture of GSM (mobile stations, BSS, network management and databases), possible weaknesses in message handling (e.g. SMS spoofing, SMSC vulnerabilities, abuse of SMS billing) allows the attackers to take advantage of their knowledge during more sophisticated attacks. To some extent, such attacks may lead to IT infrastructure paralysis. In the next paragraph, I am going to focus a bit on the telecom side of the IT infrastructure. It must be taken into consideration that the telecoms need much advanced security assessment of their infrastructure as the mobility rapidly becomes the requirement. Successful exploitation of open network segments, MMS/SMS gateway

exploitation, and security assessment of certain procedures implemented within the companies (very important) may place significant role. SS7/Sigtran or IMS penetration testing is important and may result in severe vulnerability disclosure, which may influence the mobile business. Vulnerability research in the near future will involve much more use of telecoms. For instance, while dealing with mobile networks, the attacker may more thoroughly analyze the core network (e.g. SGSN, DNS servers, VLR, MSC), GGSN, Radius and DHCP servers, databases (HLR, EIR) as well as procedures such as PDP Context Activation or GPRS Attach. Mobility factor and ubiquitous computing (among with SCADA solutions) are going to shape the future face of vulnerability research.

The assumption has been made that the readers are cognizant of code audit and a limited number of more sophisticated examples confined to more advanced area of system's exploitation will be enough interesting. For more information about the exploitation and vulnerability research, please refer to Eleytt. To learn more about the vulnerability research, please refer to [9]. To learn more about mobile-specific exploitation opportunities, please refer to [14].

Exemplary codes used for vulnerability research and assessment purposes are shown below (also available from CD enclosed):

1. The assessment of an ActiveX control (verification of buffer overrun):

```
<?XML version='1.0' standalone='yes' ?>
<package><job id='DoneInVBS' debug='false' error='true'>
<object classid='clsid:SOME-CLASSID' id='target' />
<script language='vbscript'>

arg1=222147483647
arg2=String(4444,"A")

target.someFunction arg1 ,arg2

</script></job></package>
```

2. Two exemplary codes used in manual RPC auditing:

```
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib(SOME_UID),
  version(1.0),
  helpstring("SOME NAME"),
  helpfile("WANT SOME HELP"),
  helpcontext(00000000)
]
library LIBNAME
{
    importlib("SOMEIMPORT");

    // Forward declare all types defined in this typelib
    dispinterface _int1
    dispinterface _int2

    [
      uuid(SOME_UIC),
      helpstring("some interface")
    ]
    dispinterface _int1 {
        properties:
        methods:
            [id(0xffffdd8)]
            void f1();
            [id(0x00000001), helpstring("method f1")]
            short f2();
            [id(0x00000002), helpstring("method f2")]
            short f3(BSTR op);
            [id(0x00000003), helpstring("method f3")]
```

```
#!/usr/bin/python
```

```
from impacket.dcerpc import transport
from impacket import uuid

host = '127.0.0.1'
pipe = 'browser'
UUID = ('SOME_UID', '1.0')

stringbinding = "ncacn_np:%(host)s[\\pipe\\%(pipe)s]"
stringbinding %= {'host':host, 'pipe':pipe}

trans = transport.DCERPCTransportFactory(stringbinding)


try:
  trans.connect()
except:
```

3. Code used by Spike fuzzing facility:

```
s_block_start("something");
s_string_variable("01");
s_binary("...");
..
s_string_variable("");
s_binary("53 53 53");
s_string_variable("");
s_block_end("something");
```

4. Code involved in Unix driver security assessment:

```
#include unistd.h
#include sys/types.h
#include stdio.h
#include fcntl.h
```

```
int main()
{
	int x;

	    x = open(_{SOMEDEVICE}_, O_RDWR, 0);

	    if (x == -1){
	       perror("open");
	       exit(-1);
	}

	ioctl(x, _{SOMEVALUE}_, &kop);


	return 0;
}
```

2.2 New threats and weaknesses

Technological growing up also means new kinds of technological crimes and unknown threats. We are building mobile solutions and networks, but must take into consideration the threats (see: risk management). Risk management has never been a matter of will (this is not true, but this won't be discussed further in this paper) and should be considered as the process leading to the minimalization of the risk function. Secure online transactions, anonymity, mobile threats, sensitive data storage, mobile viruses, smart card transactions and much more – the technological world is facing new kinds of threat that is in no way imaginary and should be treated seriously.

The problem is going to exacerbate in the near future as the access to an in-depth knowledge is becoming more and more free, which means that theoretically uneducated people might have very powerful knowledge and exploit their know-how for a wide variety of purposes. New vulnerabilities might concern transmission

systems (see: O-CDMA and OOK-based vulnerabilities, O-CDMA and key transmission issues, encryption issues, advanced cryptology etc.), mobile devices (various aspects of new threats, such as: weaknesses in standard compliance, remote denial of service vulnerabilities, tracking issues, privilege escalation resulting in sensitive data theft, mobile vulnerabilities and weaknesses, SMIL exploit, new viruses for embedded devices), IMS (and advanced modern communications technology), ubiquitous computing and disperse wireless sensor networks weaknesses, EMC-based weaknesses (and various similar issues requiring highly specialized knowledge in the field of electromagnetic waves, hardware, mathematics) might be exploited in the wild. Future security risks must be managed and taken into consideration even by the most powerful companies in the IT world. In case of an attack, one might expect an IT infrastructure disaster. Electronic devices might be paralyzed, important servers might get hacked into, and sensitive information might be stolen. We have already learnt much about Bluetooth security, seen or read about the exploitation of RFiD vulnerabilities, experienced the exploitation of most advanced (complex) vulnerabilities in software. We have already seen first mobile viruses and heard of vulnerabilities found in telecom infrastructure. Currently introduced implementations of telecom solutions are also prone to a number of serious weaknesses (this is far beyond the scope of the paper, but it is a very interesting issue). The picture below clearly indicates that the mobility factor is going to become the standard requirement; therefore, the enforcement of a global security policy may be significant. Moreover, the deployment of the global threat management system shall involve the mitigation and prevention procedures against mobile weaknesses and anomalies. The mobility is already widespread and will shape the vulnerability research in the next years.

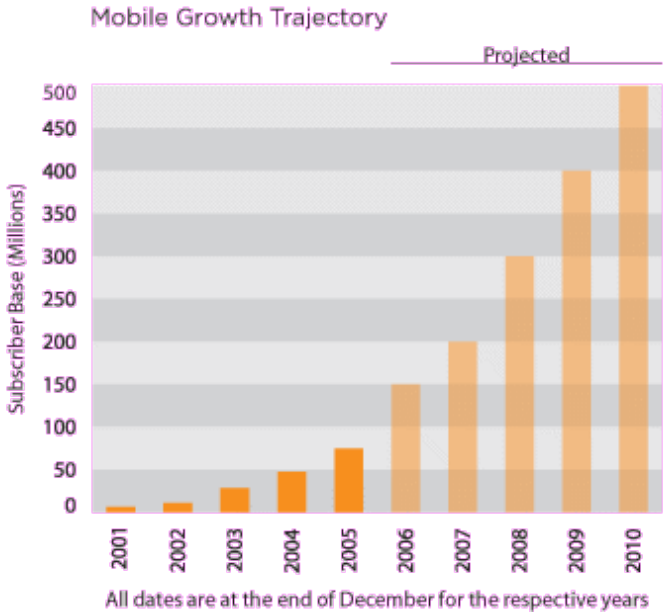

Figure 7, More and more mobile threats, www.plusmo.com

Ubiquitous computing and mobile solutions are going to swamp the market and this will enable new exploitation opportunities. Currently, we refer to traditional vulnerabilities– in the near future the vulnerability list may also include remote resource consumption-related denial of service vulnerabilities, tracking/position disclosure vulnerabilities and many other types of weaknesses (e.g.: taking advantage of GPS and map facilities may allow the attackers localize the target; the SPA/DPA analysis may reveal significant problems in the mobile devices that may then be used for e.g. remote detonation purposes). The legislation procedures change in many countries so that it could be possible to handle the problem, which may appear on the horizon. This must be, however, understood that the legislation won't keep pace with hacking as knowledge is free and just one person may have a very powerful ace in ones pocket.

Not only are going new threats to appear due to new vulnerabilities in mobile software and systems, but also most advanced e-systems (financial institutions, e-commerce, the telecoms and SCADA) are going to be more thoroughly explored.

This means that hackers are going to learn more about the critical facilities, which could – for instance – be exploited by terrorists. New threats are also going to appear due to the fact that the mobility factor is user-friendly and allows access to networks in varied conditions (and places), but the mobility has not been yet verified thoroughly in terms of security. Mobile weaknesses and vulnerabilities could, for instance, lead to creation of mobile botnets. This could be used to influence the functioning of backbone networks, e.g. mobile GWs and DNSes could be attacked and (currently often overlooked) WAP interfaces could be challenged. Mobile weaknesses could also be used to conduct more sophisticated attacks, which may use information of the position to disrupt the normal network functioning (blocking authorized users to use the network, cause denial of service conditions, collect sensitive information etc.). Testing multiple technologies such as SMS, MMS, GPRS, EDGE, NGN may reveal numerous critical weaknesses, some of which will be related to the protocols used in the backbone. The mobility also means that terminals are simply left for the end-users and, if compromised or stolen, may be used by malicious users. Furthermore, mobility also means different routing protocols, including such with selfish nodes or simple encryption algorithms (used due to resource limitations of embedded devices). New understanding of security is also connected with the use of generalized meaning of IP – mobile IP where end-users remain connected to the global network while changing their location. Current situation allows the use of many technologies, applications usually do not require sessions continuity (sessions are not managed), the introduction of mobile IP architecture ([21]) is just one step towards the possible ability to handle mobile traffic and provide (globally managed) security for end-users. The problems related to the use of mobile routing protocols are still not enough satisfactorily solved.

According to Juniper Networks ([22]), there are at least six issues Gp interface (interface between to different mobile operators) is prone to: Border Gateway bandwidth saturation, DNS flood, GTP flood, Spoofed GTP PDP Context Delete, Bad BGP Routing Information and DNS Cache Poisoning. The protection techniques are also described in [22]. This, however, shows how many

vulnerabilities known from the traditional security scene are also applicable to mobile security and also how advanced attacks may be conducted on the mobile telecom infrastructure.

We also know that many attacks exist that involve the use of Bluetooth technology, including BlueSnarf, Bluejacking, BlueBug or BlueBump. The very attacks are publicly recognized and won't be described further in this paper. The importance of these techniques lies in the mobility factor and ease-of-exploitation.

Vulnerability research will also touch mobile operating systems as their primary focus is ease-of-use, ease-of-integration with other devices as well as multiple methods of communication – security is currently put aside. Moreover, due to limited resources, the element of security is even more difficult to properly build-in. For instance, PalmOS/LibertyCrack virus was installed during a HotSync or via IR from another PDA. Other viruses also take advantage of ease-of-communication and ease-of-integration, but future analysis will also let them more effectively become stealth within the attacked handheld. Boot loader issues, kernel problems, in-depth research into file systems, inter-process communication, process and thread protection, authentication issues and many other issues are going to shape the security of mobile systems. When we take a closer look at various mobile operating systems, we can see that each of them has quite different notion of device-locking, encryption, memory management. Memory protection and permission management as well as low-level encryption are very important in case of handhelds.

Mobile devices are also prone to another kind of abuse. The attackers may have quite an easy access to device, which allows them to take advantage of Firewire, USB, JTAG or any other technology to communicate with the device. In [23] there have been numerous embedded security techniques briefly described. It shows how deep the attacker can go to conduct certain attacks. And, as mentioned earlier, more advanced attacks will more often take advantage of electronics.

7 References

References are limited to ones required for the paper.

1. Eleytt Corporation, www.eleytt.com

9. IT Underground 2007, *"Practical security thought – the bughunt"*, Michal Bucko

14. "NGN and Mobile Botnets – the future of cyberterrorism", Michal Bucko

15. Software Vulnerability Information Site, Tan Chew Keong, vuln.sg

21. „Policy Expression and Enforcement for Handheld Devices", NIST (National Institute of Standards and Technology), Wayne Jansen, Tom Karygiannis, Vlad Korolev, Serban Gavrila, Michaela Iorga

22. „GPRS Security Threats And Solution Recommendations", Alan Bavosa, Juniper Networks

23. „Introduction to Embedded Security", Joe Grand, Grand Idea Studio, Inc.

Additional references: US-CERT (BuildSecurityIn)

Eleytt Corporation

www.eleytt.com

Eleytt is specialized in mobility: mobile development and mobile security. Our experts have disclosed numerous traditional vulnerabilities as well as conduced major security audits for the telecoms. Eleytt comprises people specialized in numerous areas of telecommunications and mobile development as well as exploit writers and mobile hackers. We focus on mobility only and our focus is to improve the connectivity within the mobile infrastructures.