DFN CERT

UNIX and Linux based Kernel Rootkits

DIMVA 2004
Andreas Bunten

Agenda

- Introduction
- Classification of rootkits
- Countermeasures
- Examples
- Conclusions

The Setting

Situation:

- A system is scanned for vulnerable services
- Remote and local exploits are used to break in
- The system is compromised and the attacker gained the access privileges of the administrator

What does the attacker want?

- Reconnect without having to use the exploit again
- Stay unnoticed as long as possible

Definition

DFN CERT

A **Rootkit** enables an attacker to stay unnoticed on a compromised system so he can use it for his purposes.

Traditional rootkit 'features':

- Hide files, processes and network connections
- Filter logfiles
- Provide a hidden backdoor into the system

Timeline

1990

2000

- *Hiding out under UNIX*, Black Tie Affair, Phrack 25, 1989
- System Binaries are exchanged on SunOS 4 systems (*Trojan Horses*)
- Linux Rootkits appear
- *Abuse of the Linux Kernel for Fun and Profit*, Halflife, Phrack 50, 1997
- Kernel Rootkits appear for all popular UNIX versions and Microsoft Windows

Classification of kernel rootkits

Classification

Different criteria of a rootkit can be used for classification.

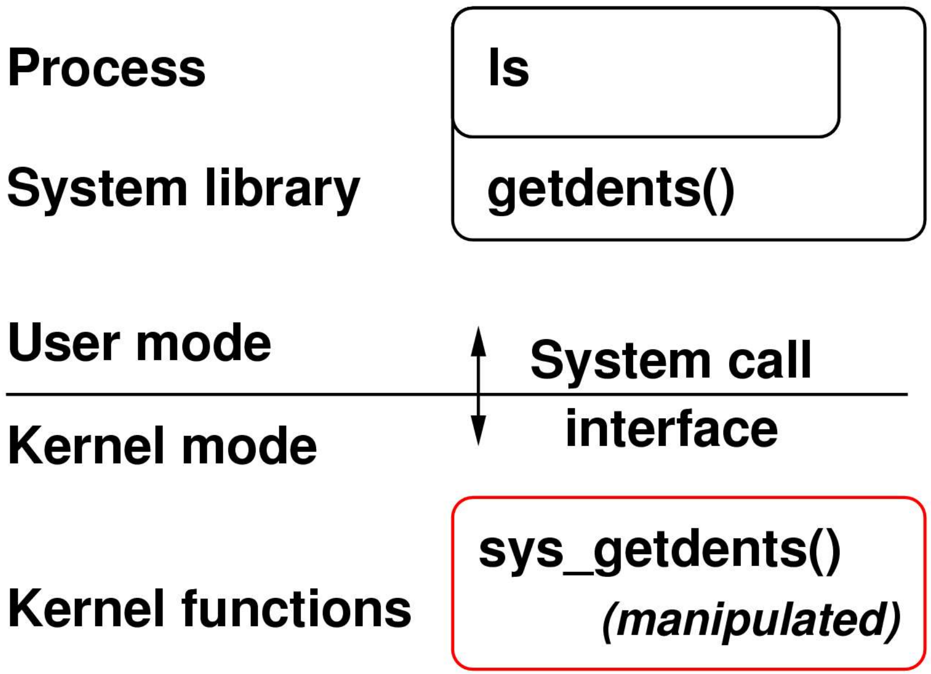
Example: How is the flow of execution intercepted?

- The flow of execution needs to be intercepted or modified at some point
- The manipulation can take place at many different levels in user or kernel space. This determines:
  - What features the rootkit can provide
  - How the rootkit can be detected

Where does a rootkit intercept 'ls' to hide files?

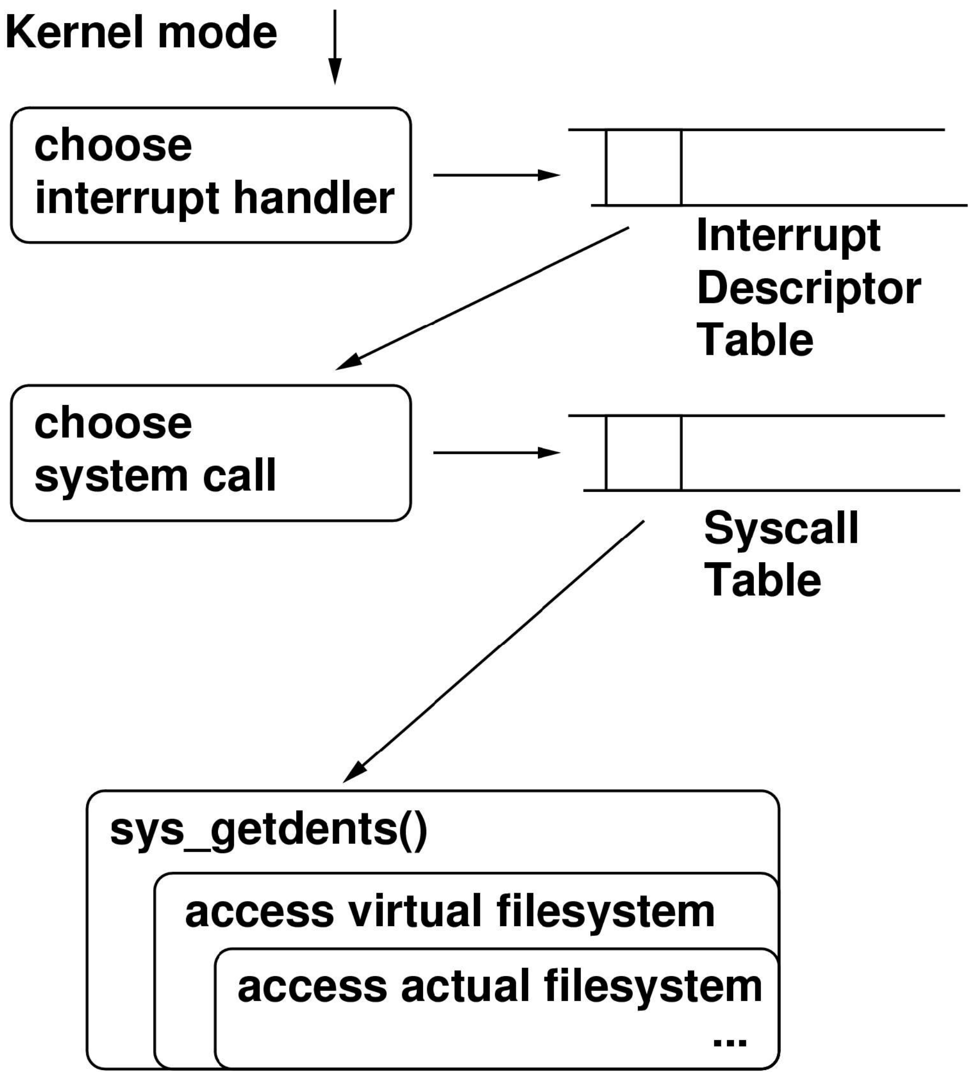
Intercepting the flow of execution

The flow of execution



- Process 'ls' uses library, which makes system call
- The system changes into kernel mode and calls function in kernel
- Every user process is affected when the kernel is manipulated

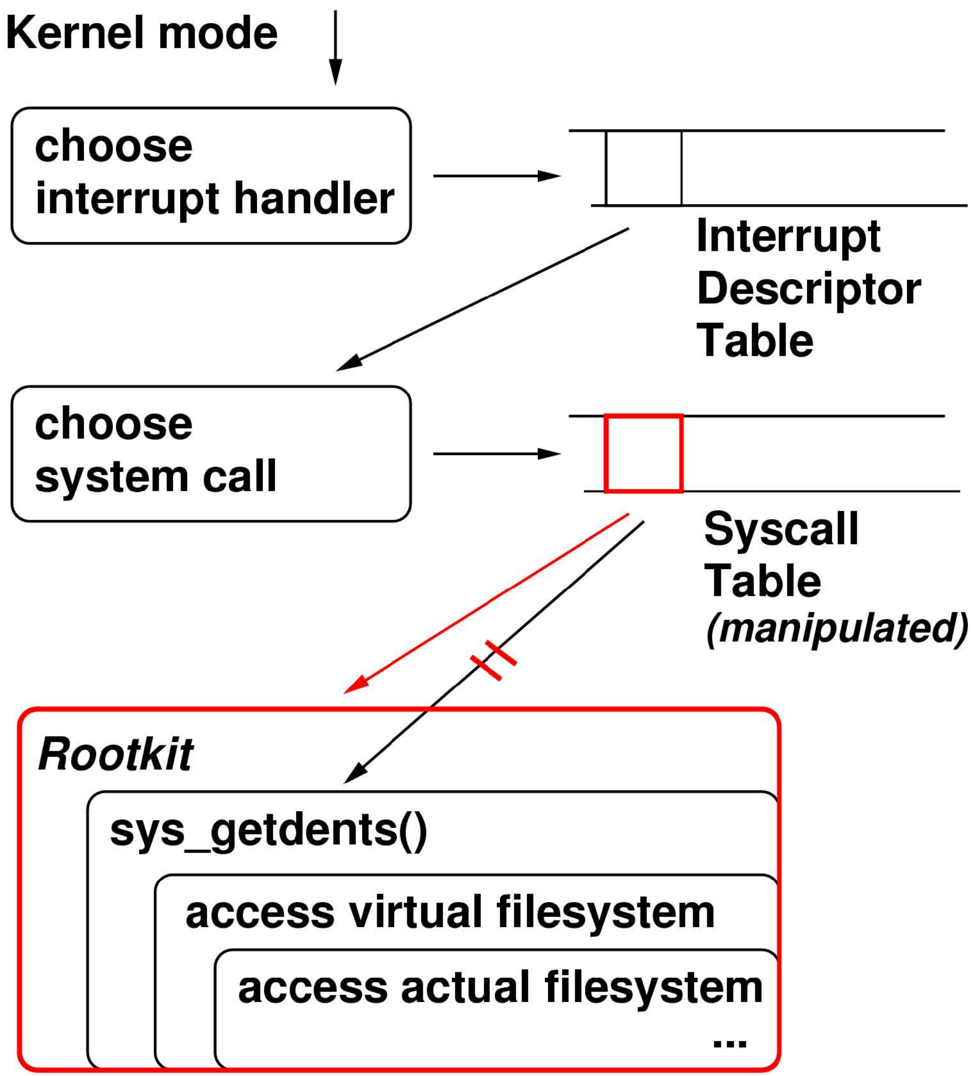
Inside the kernel



Executing a syscall in the kernel:

- Interrupt handler consults the IDT
- System call handler consults Syscall Table
- Function implementing the system call is executing other kernel functions

Inside the kernel



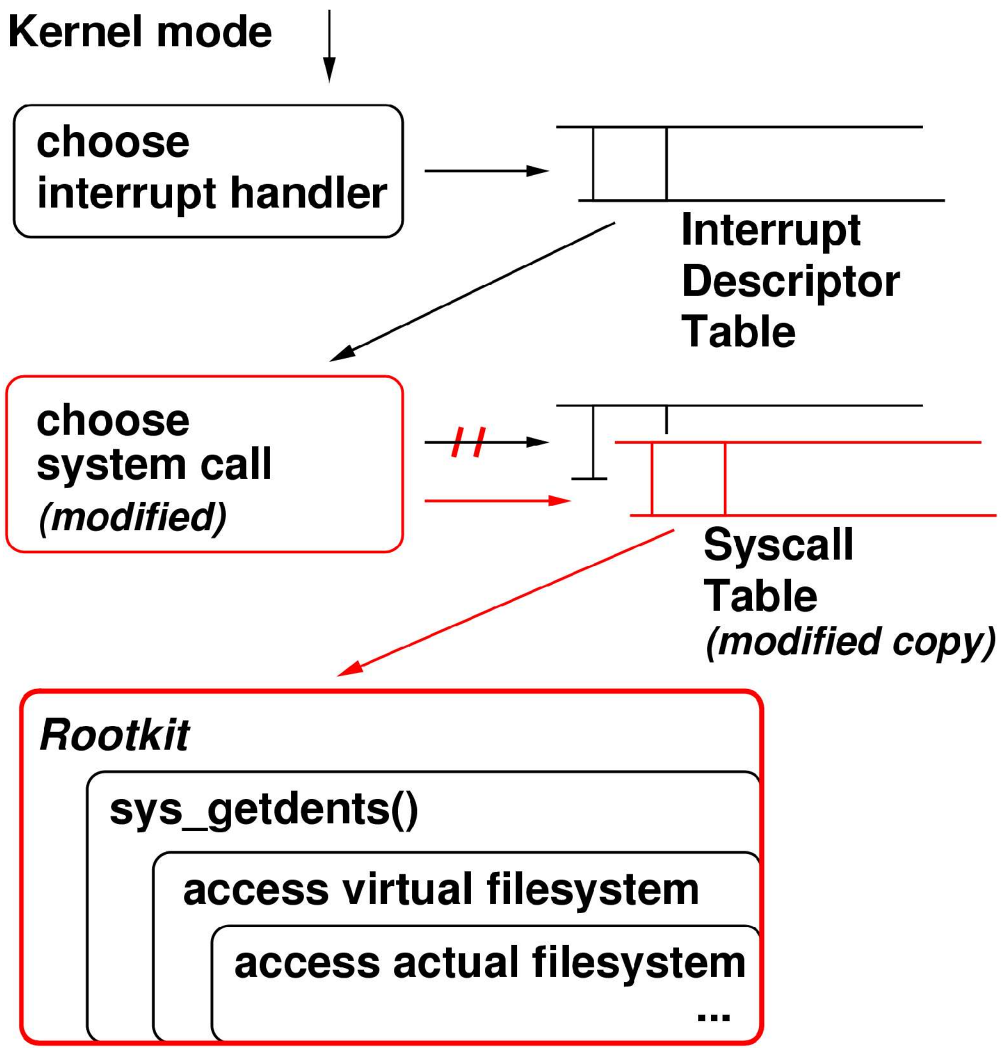
Manipulating the Syscall Table:

- The rootkit is called instead of original function
- Rootkit acts as a wrapper
- Method used by first kernel rootkits

Examples:

Adore, KIS, ...

Inside the kernel



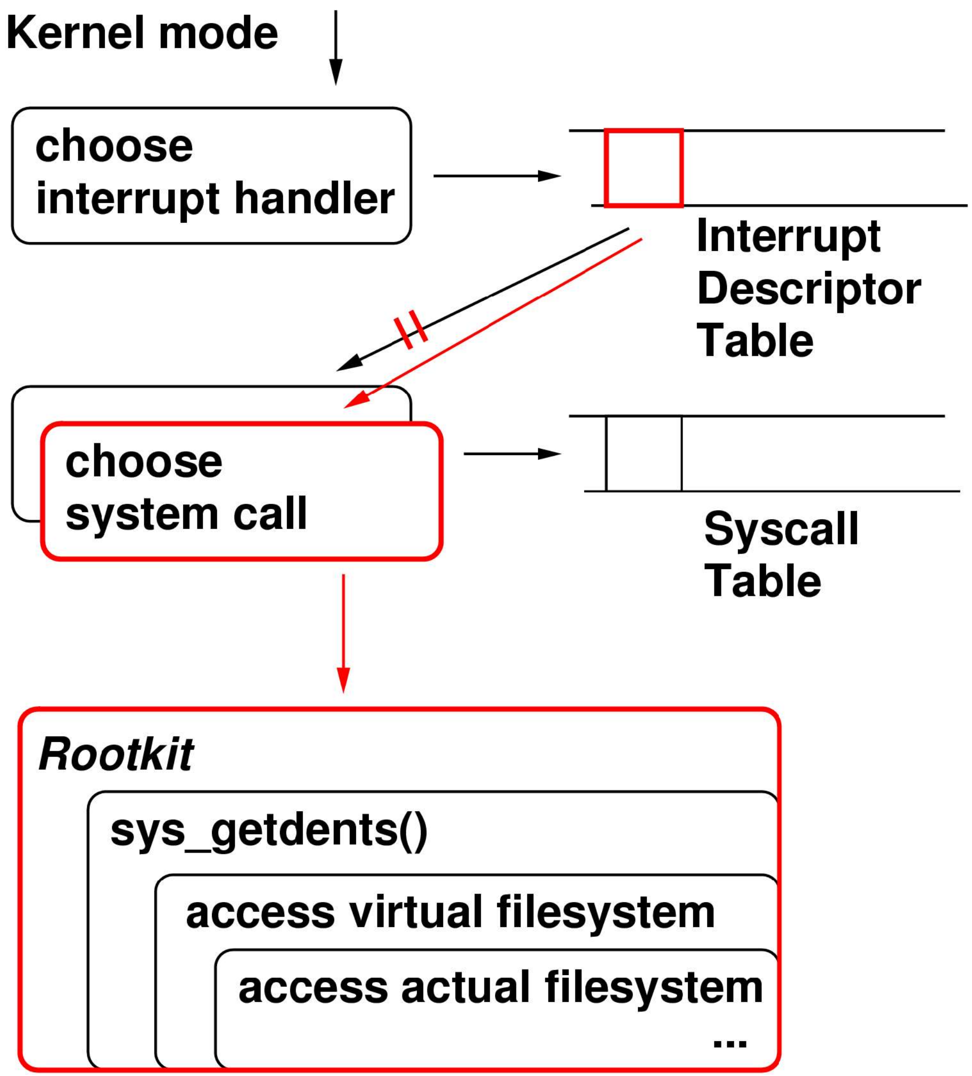
Copying the syscall table / handler:

- Original syscall table is not modified
- Modified syscall handler uses manipulated copy

Examples:

SucKIT

Inside the kernel



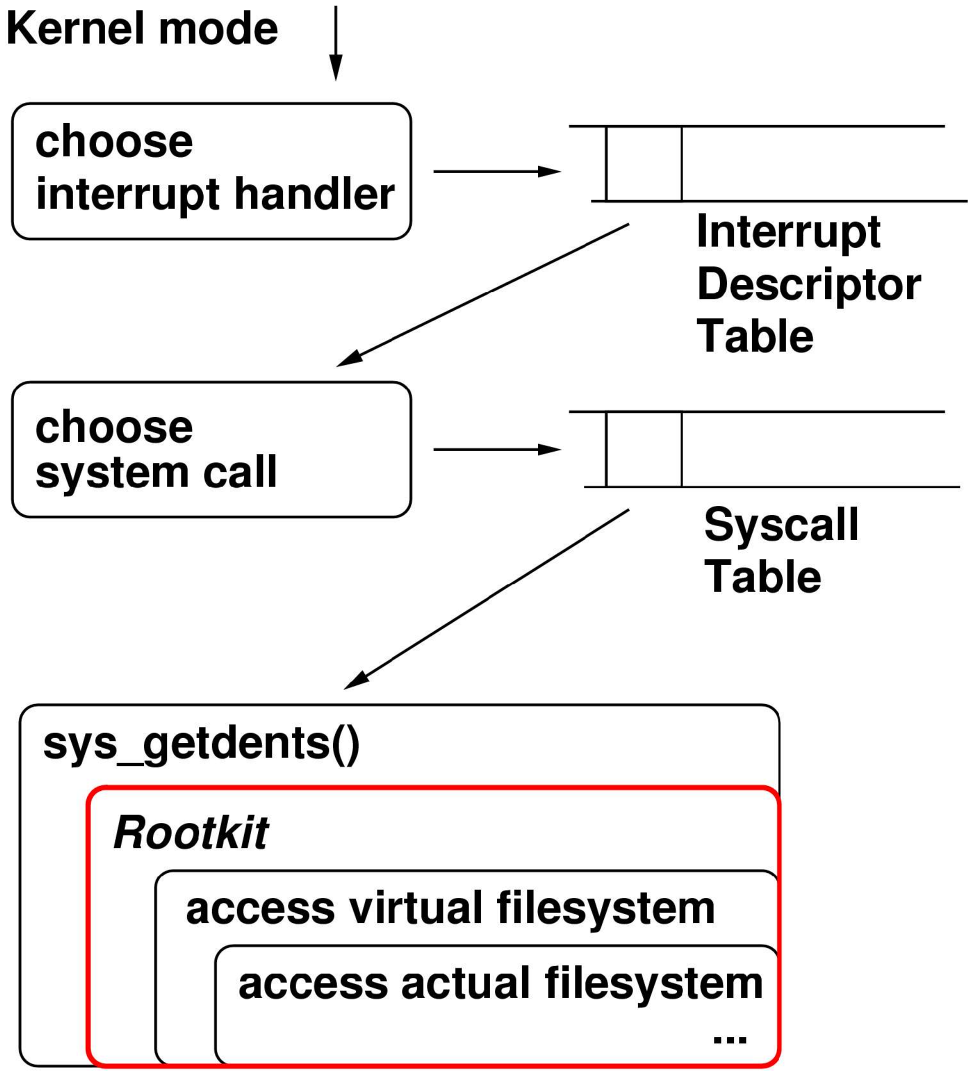
Manipulating the IDT:

- A different syscall handler is used, which calls rootkit
- No need to modify syscall handler or syscall table

Examples:

Concept rootkits

Inside the kernel



Manipulation deeper inside the kernel:

- Less central kernel structures are manipulated
- Hard to detect since many kernel structures need to be monitored

Examples:

Adore-NG manipulates /proc using virtual filesystem (VFS)

Classification

Intercepting the flow of execution:

- User space:
  - Exchange system binaries
  - Infect library
- Manipulation in kernel space:
  - Interrupt Descriptor Table
  - Syscall Handler
  - Syscall Table
  - VFS layer
  - ...

Classification

DFN CERT

Further criteria useable for classification:

- How is a backdoor provided?
- How is the rootkit loaded at restart of the system?
- What features are provided?
  - E.g. automatic log filtering of hidden processes (KIS)
- How is code transferred into the kernel?
  - Official API for kernel modules (Adore, knark, ...)
  - Raw memory device (e.g. /dev/kmem or kernel exploit) (SucKIT)

Classification

DFN CERT

Classification of example rootkits:

	Adore 0.34	SucKIT 1.3b	Adore-NG 1.31
Intercepting the flow of execution	syscall table	syscall handler	VFS
Code transfer into the kernel	module	raw memory access	module
Remote backdoor included	-	yes	-
Reload mechanism	-	/sbin/init	tool to infect existing modules

Countermeasuers for current kernel rootkits

Countermeasures

DFN CERT

Typical methods to detect a rootkit:

- Checksums of important files (aide, tripwire, ...)
- Rootkit detector programs using signatures (chkrootkit, rootkit hunter, ...)
- Backups of central kernel structures (kstat)
- Runtime measurement of system calls (patchfinder)
- Anti-rootkit kernel modules (St Michael)
- Offline / forensic analysis (TCT, ...)
- Watching the network traffic / flows from 3rd system
- Manual logfile analysis and search

Countermeasures

Applying runtime detection methods:

	Adore 0.34	SucKIT 1.3b	Adore-NG 1.31
Checksums aide 0.7	✓	✓ ✗	✓ ✗
Process list chkproc	✓	✓	✗
Kernel structures kstat 2.4	✓	✓ ✗	✓ ✗
Rootkit detector chkrootkit 0.43	✓	✓	✓ ✗
Runtime measurements ...	✓	💣	💣

Examples

Rootkits seen by DFN-CERT

Examples

DFN CERT

Rootkits seen in real incidents:

- Plattforms: mostly Linux, MS Windows and Solaris; occasionally BSD, Tru64, HP-UX, AIX, ...
- Attackers using different misconfigured rootkits together on one system
- Attackers combining sophisticated methods:
  - multistage attacks
  - obfuscated rootkits

Examples

Example incident with obfuscated rootkit:

- Rootkit was installed on SSH gateway of research site
- Logins were sniffed / ~ 30 research sites involved
- Rootkit SucKIT was combined with burneye tool
  - Rootkit loader (/sbin/init) was obfuscated (no encryption)
  - Output of 'strings' was empty
  - Obfuscation could be reversed with free tools
- As soon as rootkit was known:
  - Remote scanner for this version of SucKIT can be used
  - Local detection became very easy

Examples

```
linux:/sbin # ls -al init*
-rwxr-xr-x  1 root root  392124  Jan 6 2003 init
linux:/sbin # mv init initX
linux:/sbin # ls -al init*
-rwxr-xr-x  1 root root   28984  Jan 6 2003 initX
linux:/sbin # ./initX
/dev/null
Detected version: 1.3b
use:
./init.bak <uivfp> [args]
u       - uninstall
i       - make pid invisible
v       - make pid visible
f [0/1] - toggle file hiding
p [0/1] - toggle pid hiding
linux:/sbin #
```

Conclusions

- Many criteria can be used for the classification of rootkits - e.g. the interception of the flow of execution
- Most detection tools are based on specific features of rootkits; few use general mechanisms for detection
- Experience shows that identifying the type of rootkit helps dealing with the incident
- Tools for generic detection of malware are needed

Questions?

???

Feedback / rootkits: bunten@dfn-cert.de