2012-09-30

Powiedzieć Blah...

Secure Systems and Networks

Tomasz R. Surmacz

Institute of Computer Engineering, Control and Robotics (CECR)
Wrocław University of Technology

November 2010

Choose yourself and new technologies

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

Wrocław University of Technology

EUROPEAN SOCIAL FUND

Project co-financed from the EU European Social Fund

Course overview
Lecture topics
- User/group identifier and access rights
- Memory protection and special programs
- Writing safe programs – system calls caveats and loopholes
- System administrator and access permissions
- Security threats in layer 3 of IP protocols (ICMP, UDP, TCP)
- Security threats in specific services of TCP/IP and UDP/IP protocols (SMTP, FTP, etc.)
- Secure Sockets Layer (SSL) and other mechanisms in layers 3-7 (S-HTTP, PGP)
- Principles of cryptography
- Viruses, trojan horses, worms, system configuration errors
- Defensive programming
- Port scanning and methods of active network checking
- Packet filters
- Firewall systems

Source file: *RCSfile* : 00 — *MPIE* — 2010 — *en.txt*, *v*
- NoSubsection: Lecture topics

Course overview

Lecture topics

- User/group identifier and access rights
- Memory protection and special programs
- Writing safe programs – system calls caveats and loopholes
- System administrator and access permissions
- Security threats in layer 3 of IP protocols (ICMP, UDP, TCP)
- Security threats in specific services of TCP/IP and UDP/IP protocols (SMTP, FTP, etc.)
- Secure Sockets Layer (SSL) and other mechanisms in layers 3-7 (S-HTTP, PGP)
- Principles of cryptography
- Viruses, trojan horses, worms, system configuration errors
- Defensive programming
- Port scanning and methods of active network checking
- Packet filters
- Firewall systems

Course overview
Literature
- Tomasz Surmacz – Secure Systems and Networks
- Garfinkel & Spafford – Practical Unix and Network Security
- Silberschatz, Abraham – Operating Systems Concepts

Additional literature
- Schneier, Bruce – Practical Cryptography
- Clifford Stoll – Cuckoo's Egg
- Bach, Maurice J. – Design of the UNIX Operating System
- Stevens, W. Richard – UNIX Network Programming

Source file: *RCSfile* : 00 – *MPIE* – 2010 – *en.txt*, *v* (from NoSubHead)
- NoSubsection: Literature
Citation: *Garfinkel* : 2003 : *PUI*
Citation: *Garfinkel* : 2003 : *PUI*
Citation: *silberschatz – en*
- Subsection: Additional literature
Citation: *schneier – en*
Citation: *cuckooen*
Citation: *bachen*
Citation: *Stevens* : 1999 : *UNP*
Citation: *Stevens* : 1997 : *UNP*

Course overview

Literature

- Tomasz Surmacz – Secure Systems and Networks
- Garfinkel & Spafford – Practical Unix and Network Security
- Silberschatz, Abraham – Operating Systems Concepts

Additional literature

- Schneier, Bruce – Practical Cryptography
- Clifford Stoll – Cuckoo's Egg
- Bach, Maurice J. – Design of the UNIX Operating System
- Stevens, W. Richard – UNIX Network Programming

Security of computer systems and networks

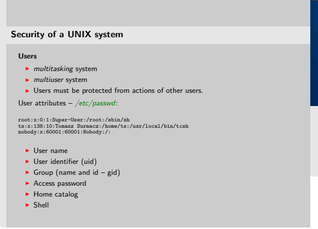
What do we mean by 'security'?
- Security of individual computers
- Security of server systems
- Routers security
- Security of physical connections
- Security of the protocols used for data exchange
- Physical security of the equipment and the backed up data
- Security of data in a computer system
- Access control to the data and to the system

Source file: *RCSfile* : *sec* – *generalintro* – *en.txt*, *v*
Citation: *Garfinkel* : 2003 : *PUI*
- NoSubsection: What do we mean by 'security'?

Security of computer systems and networks

What do we mean by 'security'?

- Security of individual computers
- Security of server systems
- Routers security
- Security of physical connections
- Security of the protocols used for data exchange
- Physical security of the equipment and the backed up data
- Security of data in a computer system
- Access control to the data and to the system



Source file: *RCSfile* : *unix* — *users* — *en.txt*, *v*
- NoSubsection: Users

Security of a UNIX system

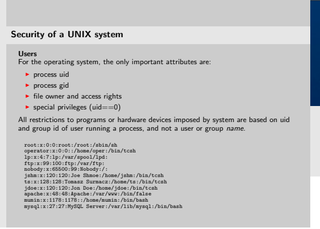
Users

- *multitasking* system
- *multiuser* system
- Users must be protected from actions of other users.

User attributes – */etc/passwd*:

```
root:x:0:1:Super-User:/root:/sbin/sh
ts:x:138:10:Tomasz Surmacz:/home/ts:/usr/local/bin/tcsh
nobody:x:60001:60001:Nobody:/:
```

- User name
- User identifier (uid)
- Group (name and id – gid)
- Access password
- Home catalog
- Shell


Security of a UNIX system

Users
For the operating system, the only important attributes are:
- process uid
- process gid
- file owner and access rights
- special privileges (uid==0)

All restrictions to programs or hardware devices imposed by system are based on uid and group id of user running a process, and not a user or group *name*.

```
root:x:0:0:root:/root:/sbin/sh
operator:x:0:0::/home/oper:/bin/tcsh
lp:x:4:7:lp:/var/spool/lpd:
ftp:x:99:100:ftp:/var/ftp:
nobody:x:65500:99:Nobody:/:
jshm:x:120:120:Joe Shmoe:/home/jshm:/bin/tcsh
ts:x:128:128:Tomasz Surmacz:/home/ts:/bin/tcsh
jdoe:x:120:120:Jon Doe:/home/jdoe:/bin/tcsh
apache:x:48:48:Apache:/var/www:/bin/false
mumin:x:1178:1178::/home/mumin:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
```

Source file: *RCSfile* : *unix* — *users* — *en.txt*, *v* (from NoSubHead)
- NoSubsection: Users

Security of a UNIX system

Users

For the operating system, the only important attributes are:

- process uid
- process gid
- file owner and access rights
- special privileges (uid==0)

All restrictions to programs or hardware devices imposed by system are based on uid and group id of user running a process, and not a user or group *name*.

```
root:x:0:0:root:/root:/sbin/sh
operator:x:0:0::/home/oper:/bin/tcsh
lp:x:4:7:lp:/var/spool/lpd:
ftp:x:99:100:ftp:/var/ftp:
nobody:x:65500:99:Nobody:/:
jshm:x:120:120:Joe Shmoe:/home/jshm:/bin/tcsh
ts:x:128:128:Tomasz Surmacz:/home/ts:/bin/tcsh
jdoe:x:120:120:Jon Doe:/home/jdoe:/bin/tcsh
apache:x:48:48:Apache:/var/www:/bin/false
mumin:x:1178:1178::/home/mumin:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
```

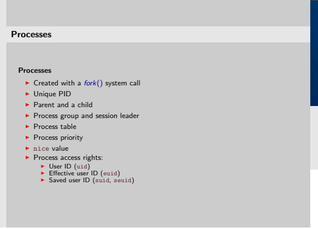
Processes

- In a multitasking environment, many tasks (processes) may run simultaneously.
- Each process may be run by a different user.
- Processes should not disturb each other. Errors in one process should not affect running of other processes.
- Process protection is required:
  - Processes of different users – protection against removing a process or modifying its data.
  - Processes of the same user – protection against data modification.
  - Access control to the private data of a process.
  - Access control to shared resources.

Processes

Source file: *RCSfile* : *unix* – *processes* – *en.txt*, *v*

- In a multitasking environment, many tasks (processes) may run simultaneously.
- Each process may be run by a different user.
- Processes should not disturb each other. Errors in one process should not affect running of other processes.
- Process protection is required:
  - Processes of different users – protection against removing a process or modifying its data.
  - Processes of the same user – protection against data modification.
  - Access control to the private data of a process.
  - Access control to shared resources.



Source file: *RCSfile* : *unix* – *processes* – *en.txt*, *v* (from NoSubHead)
- NoSubsection: Processes

Processes

Processes

- Created with a *fork*() system call
- Unique PID
- Parent and a child
- Process group and session leader
- Process table
- Process priority
- `nice` value
- Process access rights:
  - User ID (`uid`)
  - Effective user ID (`euid`)
  - Saved user ID (`suid`, `seuid`)

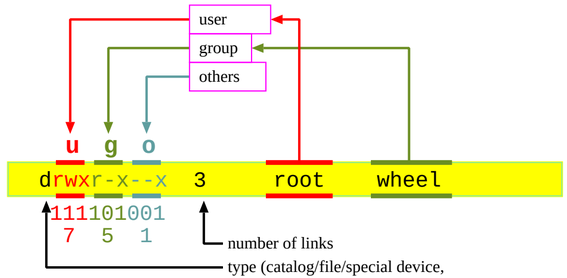
File access permissions
Each file in a filesystem has its owner and group owner
Each file has access attributes grouped in 3 categories:
for user
for group
for others
basic access permissions:
(r)ead
(w)rite
e(x)ecute

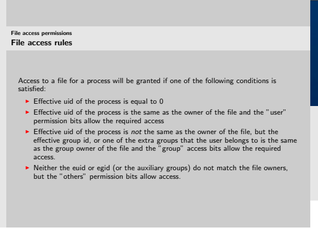
File access permissions

Source file: *RCSfile* : *unix* — *filemodes* — *en.txt*, *v*

- Picture: prdost

- Each file in a filesystem has its owner and group owner
- Each file has access attributes grouped in 3 categories:
  - for user
  - for group
  - for others
- basic access permissions:
  - (r)ead
  - (w)rite
  - e(x)ecute




File access permissions
File access rules

Access to a file for a process will be granted if one of the following conditions is satisfied:
- Effective uid of the process is equal to 0
- Effective uid of the process is the same as the owner of the file and the "user" permission bits allow the required access
- Effective uid of the process is *not* the same as the owner of the file, but the effective group id, or one of the extra groups that the user belongs to is the same as the group owner of the file and the "group" access bits allow the required access.
- Neither the euid or egid (or the auxiliary groups) do not match the file owners, but the "others" permission bits allow access.

Source file: *RCSfile* : *unix* – *filemodes* – *en.txt*, *v* (from subhead)
- Subsection: File access rules

File access rules

Access to a file for a process will be granted if one of the following conditions is satisfied:

- Effective uid of the process is equal to 0
- Effective uid of the process is the same as the owner of the file and the "user" permission bits allow the required access
- Effective uid of the process is *not* the same as the owner of the file, but the effective group id, or one of the extra groups that the user belongs to is the same as the group owner of the file and the "group" access bits allow the required access.
- Neither the euid or egid (or the auxiliary groups) do not match the file owners, but the "others" permission bits allow access.

Traditional passwords
Comparison of password security
- Comparison of strength given by passwords containing different categories of characters.
- For time considerations, we assume 8000 cracks per second, which is true for a single Pentium 300 MHz processor.
- 10 computers cracking the same password in parallel decrease the required time 10 times.

Characters used	no. of passwords	breaking time required
4-digit password (PIN code)	10000	1.2 seconds
typical language vocabulary	100000	12 s
4-letter password	456976	1 minute
8-digits password	$1 \cdot 10^8$	3 hours 25 mins
6-letter password	$3 \cdot 10^8$	11 hours
8-letter password, lowercase	$2 \cdot 10^{10}$	302 days
8-character password, lowercase and digits	$2.8 \cdot 10^{12}$	11 years
8-letter password, mixed case	$5 \cdot 10^{13}$	212 years
8 characters, all ASCII 32-127	$7 \cdot 10^{15}$	28594 years
8 characters, all keyboard 1-127	$6.7 \cdot 10^{16}$	268246 years
All possibilities for MD5 hash function	$3.4 \cdot 10^{38}$	$1.3 \cdot 10^{27}$ years
Estimated age of the Universe		10^{10} years

Source file: *RCSfile* : *sec* – *passwd* – *en.txt*, *v* (from subhead)
- Subsection: Comparison of password security

1

Traditional passwords

Comparison of password security

- Comparison of strength given by passwords containing different categories of characters.
- For time considerations, we assume 8000 cracks per second, which is true for a single Pentium 300 MHz processor.
- 10 computers cracking the same password in parallel decrease the required time 10 times.

Characters used	no. of passwords	breaking time required
4-digit password (PIN code)	10000	1.2 seconds
typical language vocabulary	100000	12 s
4-letter password	456976	1 minute
8-digits password	$1 \cdot 10^8$	3 hours 25 mins
6-letter password	$3 \cdot 10^8$	11 hours
8-letter password, lowercase	$2 \cdot 10^{10}$	302 days
8-character password, lowercase and digits	$2.8 \cdot 10^{12}$	11 years
8-letter password, mixed case	$5 \cdot 10^{13}$	212 years
8 characters, all ASCII 32-127	$7 \cdot 10^{15}$	28594 years
8 characters, all keyboard 1-127	$6.7 \cdot 10^{16}$	268246 years
All possibilities for MD5 hash function	$3.4 \cdot 10^{38}$	$1.3 \cdot 10^{27}$ years
Estimated age of the Universe		10^{10} years

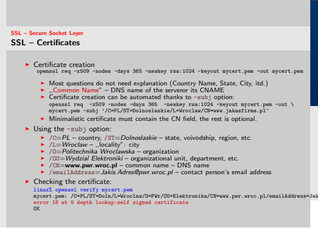
Time-restricted passwords
- Password changes automatically every 60 seconds.
- Centralized authentication server.
- A special device generating these passwords for user is needed (looking as a credit card or a key fob).
- A password shown on the display has to be prepended with a PIN
- Password is invalidated immediately after logging in and cannot be used to login to another system, even if the 60-seconds time has not expired yet.
- Some existing solutions: SecureID and VASCO tokens.
- More sophisticated tokens have a pin pad and get activated only after entering the right PIN.
- Mechanisms exist to guarantee time synchronization between the token and the authentication server.

Time-restricted passwords

Source file: *RCSfile* : *sec* – *passwd* – *otp* – *secid* – *en.txt*, *v*

- Picture: rsa-1

- Password changes automatically every 60 seconds.
- Centralized authentication server.
- A special device generating these passwords for user is needed (looking as a credit card or a key fob).
- A password shown on the display has to be prepended with a PIN
- Password is invalidated immediately after logging in and cannot be used to login to another system, even if the 60-seconds time has not expired yet.
- Some existing solutions: SecureID and VASCO tokens.
- More sophisticated tokens have a pin pad and get activated only after entering the right PIN.
- Mechanisms exist to guarantee time synchronization between the token and the authentication server.


SSL – Secure Socket Layer
SSL – Certificates
- Certificate creation
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
  - Most questions do not need explanation (Country Name, State, City, itd.)
  - „Common Name" – DNS name of the serveror its CNAME
  - Certificate creation can be automated thanks to -subj option:
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out \
mycert.pem -subj '/C=PL/ST=Dolnoslaskie/L=Wroclaw/CN=www.jakasfirma.pl'
  - Minimalistic certificate must contain the CN field, the rest is optional.
- Using the -subj option:
  - /C=*PL* – country, /ST=*Dolnoslaskie* – state, voivodship, region, etc.
  - /L=*Wroclaw* – „locality": city
  - /O=*Politechnika Wroclawska* – organization
  - /OU=*Wydzial Elektroniki* – organizational unit, department, etc.
  - /CN=**www.pwr.wroc.pl** – common name – DNS name
  - /emailAddress=*Jakis.Adres@pwr.wroc.pl* – contact person's email address
- Checking the certificate:
linux% openssl verify mycert.pem
mycert.pem: /C=PL/ST=Doln/L=Wroclaw/O=PWr/OU=Elektronika/CN=www.pwr.wroc.pl/emailAddress=Jakis.Adres@pwr.wroc.pl
error 18 at 0 depth lookup:self signed certificate
OK

Source file: *RCSfile* : *sec* — *ssl* — *cert* — *creation* — *en.txt*, *v* (from subhead)
- Subsection: SSL – Certificates

SSL – Certificates

- Certificate creation
  `openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem`
  - Most questions do not need explanation (Country Name, State, City, itd.)
  - „Common Name" – DNS name of the serveror its CNAME
  - Certificate creation can be automated thanks to `-subj` option:
    ```
    openssl req  -x509 -nodes -days 365  -newkey rsa:1024 -keyout mycert.pem -out \
    mycert.pem -subj '/C=PL/ST=Dolnoslaskie/L=Wroclaw/CN=www.jakasfirma.pl'
    ```
  - Minimalistic certificate must contain the CN field, the rest is optional.
- Using the `-subj` option:
  - `/C=`*PL* – country, `/ST=`*Dolnoslaskie* – state, voivodship, region, etc.
  - `/L=`*Wroclaw* – „locality": city
  - `/O=`*Politechnika Wroclawska* – organization
  - `/OU=`*Wydzial Elektroniki* – organizational unit, department, etc.
  - `/CN=`**www.pwr.wroc.pl** – common name – DNS name
  - `/emailAddress=`*Jakis.Adres@pwr.wroc.pl* – contact person's email address
- Checking the certificate:
  ```
  linux% openssl verify mycert.pem
  mycert.pem: /C=PL/ST=Doln/L=Wroclaw/O=PWr/OU=Elektronika/CN=www.pwr.wroc.pl/emailAddress=.
  error 18 at 0 depth lookup:self signed certificate
  OK
  ```


SSL – Secure Socket Layer
SSL – certificate signing
1. Certificate creation means generating „certificate requests" (more precicesly: „certificate signing request")
openssl req -new -key privkey.pem -out cert.csr
2. Checking the contents of this request:
openssl req -in cert.csr -noout -text
3. Such a „certificate reqest" (plik cert.csr) may be sent to the CA for signing or one can sign it oneself (especially when having someone's own CA)
Individual CA in demoCA subdirectory (demoCA/cacert.pem, demoCA/private/cakey.pem)
openssl ca -policy policy_anything -out newcert2.pem -infiles newreq.pem
1. Creation of a self-signed certificate:
openssl genrsa -des3 -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
- First command generates the key, the second one - the certificate.
- The -des3 option locks the certificate with a password (otherwise – a paswordless certificate)
- If the password-protected certificate is to be used by the WWW server, it will be necessary to manually unlock the certificate each time the server is restarted.
4. Verification:
openssl x509 -text -in cacert.pem
- Checking selected information – with -issuer, -subject, -dates, -hash and -fingerprint options

Source file: *RCSfile* : *sec* — *ssl* — *cert* — *selfsign* — *en.txt*, *v* (from subhead)
- Subsection: SSL – certificate signing

SSL – certificate signing

1. Certificate creation means generating ,,certificate requests" (more precicesly: ,,certificate signing request")

```
openssl req -new -key privkey.pem -out cert.csr
```

✪ Checking the contents of this request:

```
openssl req -in cert.csr -noout -text
```

2. Such a ‚certificate reqest" (plik *cert.csr*) may be sent to the CA for signing or one can sign it oneself (especially when having someone's own CA)

```
# Individual CA in demoCA subdirectory (demoCA/cacert.pem, demoCA/private/cakey.pem)
openssl ca -policy policy_anything -out newcert2.pem -infiles newreq.pem
```

1. Creation of a self-signed certificate:

```
openssl genrsa -des3 -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
```

- First command generates the key, the second one - the certificate.
- The `-des3` option locks the certificate with a password (otherwise – a paswordless certificate)
- If the password-protected certificate is to be used by the WWW server, it will be necessary to manually unlock the certificate each time the server is restarted.

✪ Verification:

```
openssl x509 -text -in cacert.pem
```

- Checking selected information – with `-issuer`, `-subject`, `-dates`, `-hash` and `-fingerprint` options

Bibliography I

Aleph One.
Smashing the stack for fun and profit.
Phrack, 49, April 2001.
http://www.phrack.org/archives/49/

Maurice J. Bach.
The Design of the UNIX Operating System.
Prentice Hall, Inc., 2 edition, 1986.

W.R. Cheswick.
An evening with Berferd, in which a cracker is lured, endured, and studied.
In Proceedings of the Winter USENIX Conference (San Francisco), January 1992.

Morris Dworkin.
Recommendation for block cipher modes of operation, December 2001.
NIST 800-38A, http://csrc.nist.gov/publications/nistpubs/800-38A/nist-sp-800-38E.pdf

Federal Information Processing Standards.
Passwords usage, May 1985.
FIPS 112, http://www.itl.nist.gov/fipspubs/fip112.htm

Bibliography I

Aleph One.
Smashing the stack for fun and profit.
Phrack, 49, April 2001.
http://www.phrack.org/archives/49/.

Maurice J. Bach.
The Design of the UNIX Operating System.
Prentice Hall, Inc., 2 edition, 1986.

W.R. Cheswick.
An evening with Berferd, in which a cracker is lured, endured, and studied.
In *Proceedings of the Winter USENIX Conference (San Francisco)*, January 1992.

Morris Dworkin.
Recommendation for block cipher modes of operation, December 2001.
NIST 800-38A, http:
//csrc.nist.gov/publications/nistpubs/800-38A/nist-sp-800-38E.pdf.

Federal Information Processing Standards.
Passwords usage, May 1985.
FIPS 112, http://www.itl.nist.gov/fipspubs/fip112.htm.

Bibliography II

Simson Garfinkel, Gene Spafford, and Alan Schwartz.
Practical Unix & Internet Security.
O'Reilly & Associates, Inc., third edition, 2003.

Mark Graff and Kenneth R. Van Wyk.
Secure coding: principles and practices.
O'Reilly & Associates, Inc., 2003.
http://etutorials.org/Programming/secure+coding/

Robert Haskins and Dale Nielsen.
Slamming Spam: A Guide for System Administrators.
Addison Wesley Professional, 1 edition, 2004.

Gordon Lyon.
Top 10 password crackers.
http://sectools.org/crackers.html, 2006.

National Institute of Standards and Technology (NIST).
DES encryption standard, January 1977.
FIPS 46.

Bibliography II

Simson Garfinkel, Gene Spafford, and Alan Schwartz.
Practical Unix & Internet Security.
O'Reilly & Associates, Inc., third edition, 2003.

Mark Graff and Kenneth R. Van Wyk.
Secure coding: principles and practices.
O'Reilly & Associates, Inc., 2003.
http://etutorials.org/Programming/secure+coding/.

Robert Haskins and Dale Nielsen.
Slamming Spam: A Guide for System Administrators.
Addison Wesley Professional, 1 edition, 2004.

Gordon Lyon.
Top 10 password crackers.
http://sectools.org/crackers.html, 2006.

National Institute of Standards and Technology (NIST).
DES encryption standard, January 1977.
FIPS 46.

Bibliography III

National Institute of Standards and Technology (NIST).
DES modes of operation, December 1980.
FIPS 81, http://www.itl.nist.gov/fipspubs/fip81.htm

Bruce Schneier.
Applied cryptography (2nd ed.): protocols, algorithms, and source code in C.
John Wiley & Sons, Inc., New York, NY, USA, 1995.

Abraham Silberschats, James L. Peterson, and Peter B. Galvin.
Operating System Concepts.
Addison-Wesley Publishing Company, Inc., 1991, 1991.

William Stallings.
Cryptography and Network Security, Principles and Practice.
Prentice Hall, 3rd edition, 2003.

W. Richard Stevens.
UNIX Network Programming, Volume 1: Networking APIs: Sockets and XTI.
Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1997.

Bibliography III

National Institute of Standards and Technology (NIST).
DES modes of operation, December 1980.
FIPS 81, http://www.itl.nist.gov/fipspubs/fip81.htm.

Bruce Schneier.
Applied cryptography (2nd ed.): protocols, algorithms, and source code in C.
John Wiley & Sons, Inc., New York, NY, USA, 1995.

Abraham Silberschats, James L. Peterson, and Peter B. Galvin.
Operating System Concepts.
Addison-Wesley Publishing Company, Inc., 1991, 1991.

William Stallings.
Cryptography and Network Security, Principles and Practice.
Prentice Hall, 3rd edition, 2003.

W. Richard Stevens.
UNIX Network Programming, Volume 1: Networking APIs: Sockets and XTI.
Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1997.

Bibliography IV

W. Richard Stevens.
UNIX Network Programmings, Volume 2: Interprocess Communications.
Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

Clifford Stoll.
The Cuckoo's Egg.
Pocket, 1st edition, 1990.

Ken Thompson.
Reflections on trusting trust.
Communication of the ACM, 27(8):761–763, August 1984.
http://cm.bell-labs.com/who/ken/trust.html.

Wietse Venema.
TCP Wrapper network monitoring, access control, and booby traps.
ftp://ftp.porcupine.org/pub/security/tcp_wrapper.pdf, July 1995.

Tim Waugh.
When is a command line not a line?
http://freshmeat.net/articles/view/337/, 2001.

Bibliography IV

W. Richard Stevens.
UNIX Network Programmings, Volume 2: Interprocess Communications.
Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

Clifford Stoll.
The Cuckoo's Egg.
Pocket, 1st edition, 1990.

Ken Thompson.
Reflections on trusting trust.
Communication of the ACM, 27(8):761–763, August 1984.
http://cm.bell-labs.com/who/ken/trust.html.

Wietse Venema.
TCP Wrapper network monitoring, access control, and booby traps.
ftp://ftp.porcupine.org/pub/security/tcp_wrapper.pdf, July 1995.

Tim Waugh.
When is a command line not a line?
http://freshmeat.net/articles/view/337/, 2001.

Bibliography V

David A. Wheeler.
Secure programming for linux and unix HOWTO – creating secure software.
http://www.dwheeler.com/secure-programs/, March 2003.

Bibliography V

David A. Wheeler.
Secure programming for linux and unix HOWTO – creating secure software.
http://www.dwheeler.com/secure-programs/, March 2003.

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!
Wrocław University of Technology
EUROPEAN SOCIAL FUND
Project co-financed from the EU European Social Fund