Technology Day

Development Romania

Nmap Scripting Engine

Florin Micu

Agenda

- Introduction to Nmap
  - Port Scanning Techniques and Algorithms
  - Service and Application Version Detection
  - Remote OS Detection
- Nmap scripting engine
  - Intro to Lua, NSE libraries
  - Scripts
    - Type
    - Format
    - Writing tutorial

Introduction to Nmap

- free, open-source port scanner
- purpose:
  - discover computers and services on a computer network, thus creating a “map” of the network
- written by Gordon Lyon (a.k.a Fyodor); first release: September 1, 1997
- runs on Linux, Windows, BSD, Mac OS X, Solaris
- console and graphical versions are available
- legal issues:
  - port scanning is not a crime

Port Scanning Basics

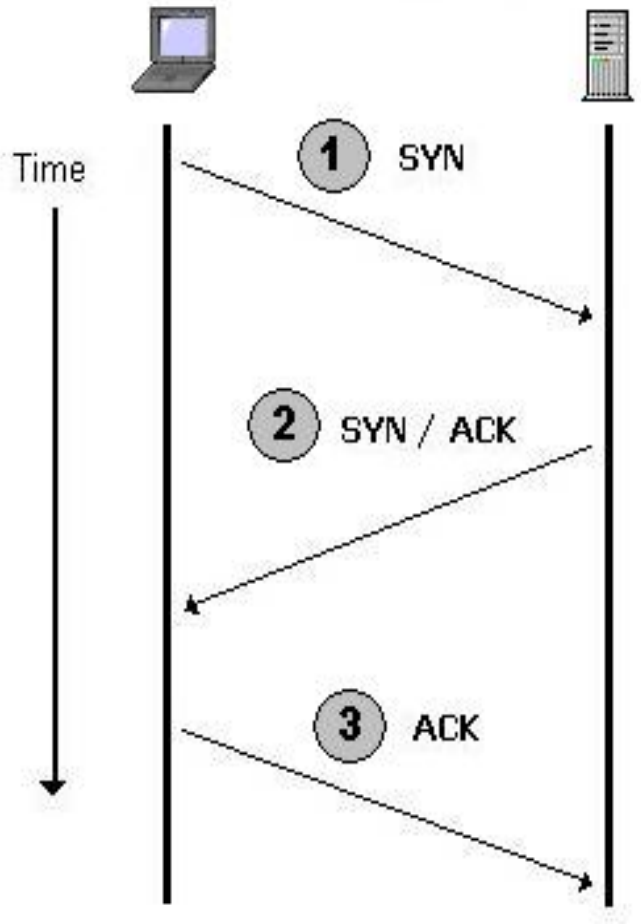
- six port states recognized by Nmap:
  - open
    - an app is actively accepting TCP connections, UDP datagrams
  - closed
    - accessible but no app is listening on it
  - filtered
    - cannot determine if it's open or closed
  - unfiltered
    - accessible but cannot tell if open or closed
  - open | filtered
  - open | closed

Scan type: TCP

- option: -sT
- uses connect()
- it's very easy to detect

  and logged
- example:

nmap -sT -p 30000 172.28.124.39

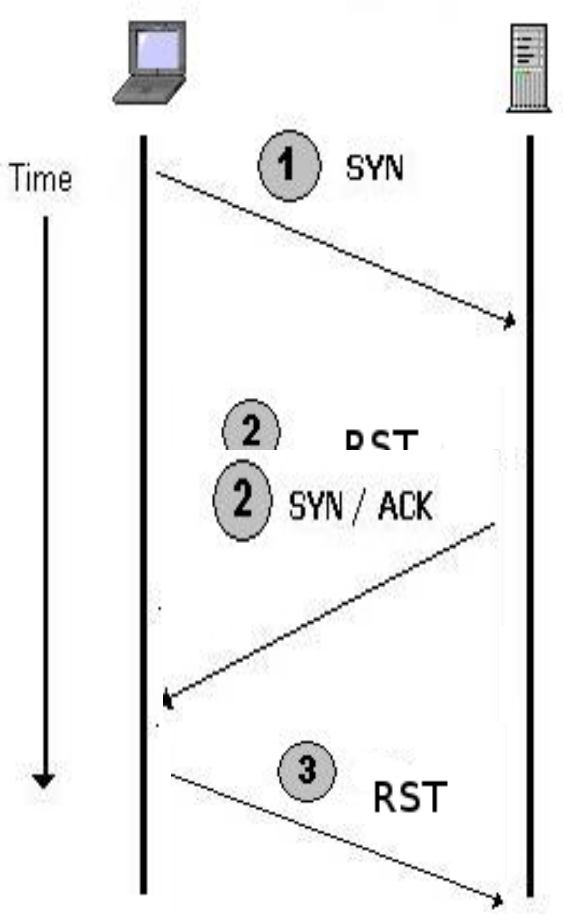


3-way handshake

Scan type: SYN

- option: -sS (stealth)
- sends only SYN packets
- less likely to be detected
- example:

nmap -sS -p 30000 172.28.124.39



Other scan types

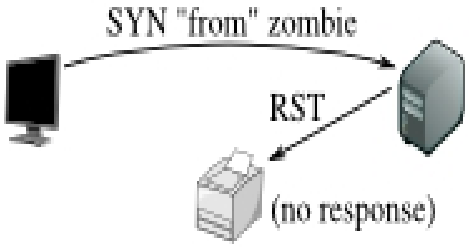
- -sF, -sN, -sX = FIN, null, XMAS
  - refers to the flags set in the TCP header
  - less likely to appear in logging system
  - less reliable (not all systems follow RFC to the letter)
- -sP = ping scan
- -sU = UDP scan

Scan type: Idle scan

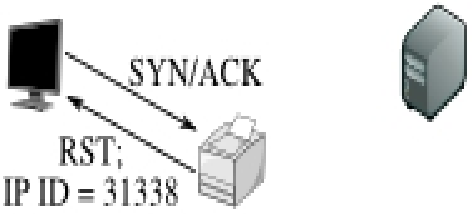
- option: -sI
- sends spoofed packets, impersonating another computer ("zombie")
  - every IP packet has an unique ID
  - host with sequential (and predictable sequence) number can be used as “zombie”
  - the latest versions of Linux, Solaris and OpenBSD, Windows Vista are not suitable as zombie
    - IPID is randomized

Step 1: Probe the zombie's IP ID.

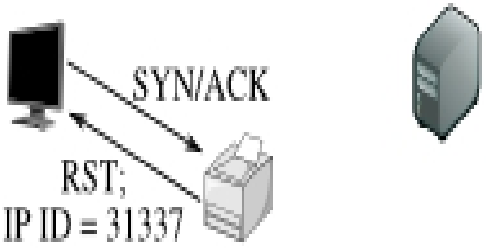
Step 2: Forge a SYN packet from the zombie.



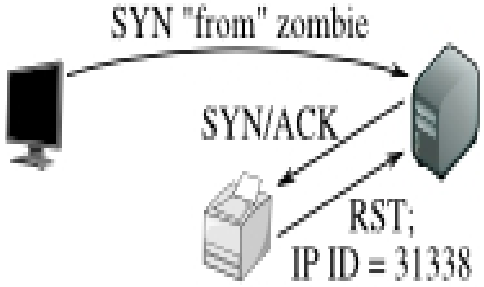
Step 3: Probe the zombie's IP ID again.



Step 1: Probe the zombie's IP ID.



Step 2: Forge a SYN packet from the zombie.



Step 3: Probe the zombie's IP ID again.

Service and Application Version Detection

- purpose
  - detect which exploits a server is vulnerable to
  - detect whether services are run on wrong ports or share the same port
- technique:
  - connect TCP, listen 5 sec, get greeting banner, start UDP probes (specific port, string) etc.
- example
  - nmap -A -T4 -F 172.28.124.39
    - A = enables OS detection and Version detection, Script scanning and Traceroute

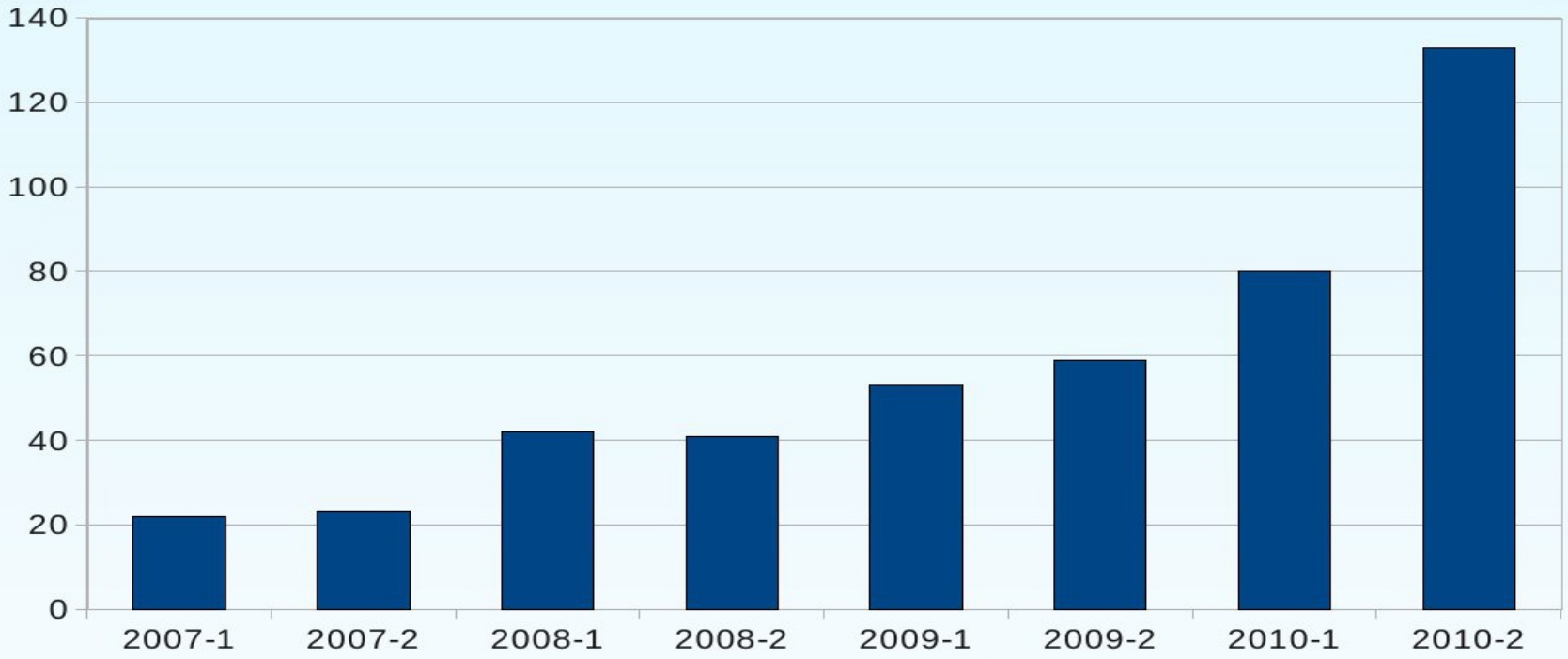
Remote OS Detection

- reasons:
  - determining vulnerability of target hosts
    - Rwho daemon on Solaris 7-9 is vulnerable, on Solaris 10 not
  - network inventory and support
  - detecting unauthorized and dangerous devices
    - detect a wireless access point (WAP), webcams, game console
- technique:
  - TCP/IP stack fingerprinting: send a series of valid and invalid TCP and UDP packets to the remote host and examine practically every bit in the responses
- example:
  - nmap -sV -O 172.28.124.39, nmap -O 172.28.124.39

Nmap Scripting Engine

- automate a wide variety of networking tasks
  - network discovery
  - more sophisticated version detection (e.g Skype2)
  - vulnerability detection
  - backdoor detection
- scripts are executed in parallel
- scripts are written in Lua scripting language
- example:
  - nmap -sC 172.28.124.39

Script Collection Growth



Why Lua?

- NSE had to be: easy to use, small in size, scalable, fast and parallelizable

- Lua is:
  - lightweight programming language
    - easy to learn, has minimalist syntax and features
    - small memory footprint
    - tiny to embed
  - safe and secure
    - no buffer overflows, no strings vulns
  - portable
  - interpreted
  - used:
    - In video game development (Warcraft, Mafia, FarCry etc)
    - Wireshark network packet analyzer, VLC media player, Snort
  - excellent documentation, actively developed

Components of NSE

- NSE libraries
  - 57 modules written in Lua
    - „bit” (bitwise op on integers) and “pcre”(perl compatible regex) are written in C/C++
  - make script writing more powerful and convenient
    - scripts need only to require the default libraries in order to use them
- NSE Scripts
  - 161 scripts: http://nmap.org/nsedoc/
- Nmap API
  - an interface to the Nmap's internal functions (Nsock library for efficient network I/O etc.)

Script categories

- every script belongs to at least one category
- auth
  - try to determine authentication credentials on the target system
- default
  - the default set and are run when using the -sC or -A options
- dos
  - may cause denial of service (they crash a service as a side effect of testing it)
- intrusive
  - the risks are too high that they will crash the target system
- safe
  - scripts that don't crash services, use large amounts of network bandwidth or other resources, or exploit security holes

Command-line arguments

- --script <filename>|<category>|<directory>|<expression>|all[,...]
- --script-args: provides args to script
- --script-trace: for debug

Script Selection

- supports logical operator: not, and, or
  - nmap --script "http-*"
  - nmap --script "not intrusive"
  - nmap --script "default and safe": scripts both in "default" and in "safe"
  - nmap --script "default or safe": scripts in "default" or "safe" category
- examples:
  - nmap --script default,safe 172.28.124.37-45

Script format

- Head
  - meta information
- Rules
  - a rule is a Lua function that returns either „true“ or „false”
- Action
  - the instructions to be executed if the script's rule is evaluated to „true”

Script format: Head

- description field
- catagories field:
  - categories = {“default”, “discovery”, “safe”}
- author field
- license field
- dependencies field
  - an array containing the names of scripts that should run before this script
  - listing a script in dependencies doesn't cause that script to be run but merely forces an ordering among the scripts that are selected
    - » dependencies = {"smb-brute"}

Script format: Rules

- prerule
  - script runs before any of Nmap's scan phases
  - for tasks which don't depend on specific scan targets
- hostrule(host)
  - runs after Nmap has performed host discovery, port scanning, version detection
  - invoked once against each target host which matches its "hostrule function"
- portrule(host, port)
  - the most common Nmap script type
  - decides if a script should run against a service
    - » portrule = shortport.port_or_service(22, "ssh")
- postrule()
  - runs after Nmap has scanned all of its targets
  - useful for formatting

Script format: Action

- the instructions to be executed when the script's prerule, portrule, hostrule or postrule triggers
- NSE scripts generally only return messages when they succeed

– a script must contain one or more rules
– a „prerule“ or a „postrule“ will always evaluate to true

Script writing tutorial

- find IP address from SSH host key
- find IP address from MAC address

Final notes

- Download Nmap from: http://nmap.org/
- NSEDoc portal: http://nmap.org/nsedoc
- NSE systems docs:

  http://nmap.org/book/nse.html
- Q&A ?

Nmap Scripting Engine

Florin Micu

Agenda

- Introduction to Nmap
  - Port Scanning Techniques and Algorithms
  - Service and Application Version Detection
  - Remote OS Detection
- Nmap scripting engine
  - Intro to Lua, NSE libraries
  - Scripts
    - Type
    - Format
    - Writing tutorial

Introduction to Nmap

- free, open-source port scanner
- purpose:
  - discover computers and services on a computer network, thus creating a “map” of the network
- written by Gordon Lyon (a.k.a Fyodor); first release: September 1, 1997
- runs on Linux, Windows, BSD, Mac OS X, Solaris
- console and graphical versions are available
- legal issues:
  - port scanning is not a crime

Port Scanning Basics

- six port states recognized by Nmap:
  - open
    - an app is actively accepting TCP connections, UDP datagrams
  - closed
    - accessible but no app is listening on it
  - filtered
    - cannot determine if it's open or closed
  - unfiltered
    - accessible but cannot tell if open or closed
  - open | filtered
  - open | closed

Scan type: TCP

- option: -sT
- uses connect()
- it's very easy to detect

  and logged
- example:

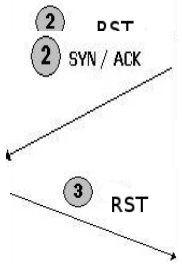
nmap -sT -p 30000 172.28.124.39

3-way handshake

Scan type: SYN

- option: -sS (stealth)
- sends only SYN packets
- less likely to be detected
- example:

nmap -sS -p 30000 172.28.124.39



Other scan types

- -sF, -sN, -sX = FIN, null, XMAS
  - refers to the flags set in the TCP header
  - less likely to appear in logging system
  - less reliable (not all systems follow RFC to the letter)
- -sP = ping scan
- -sU = UDP scan

Scan type: Idle scan

- option: -sI
- sends spoofed packets, impersonating another computer ("zombie")
  - every IP packet has an unique ID
  - host with sequential (and predictable sequence) number can be used as “zombie”
  - the latest versions of Linux, Solaris and OpenBSD, Windows Vista are not suitable as zombie
    - IPID is randomized

be the zombie's
D.

N/ACK

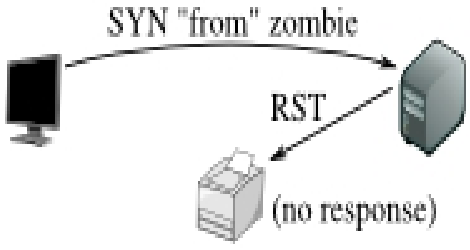
7

IP ID.

SYN/ACK

RST;
IP ID = 31337

Step 2: Forge a SYN packet from the zombie.



from the zombie.

IP ID again.

Service and Application Version Detection

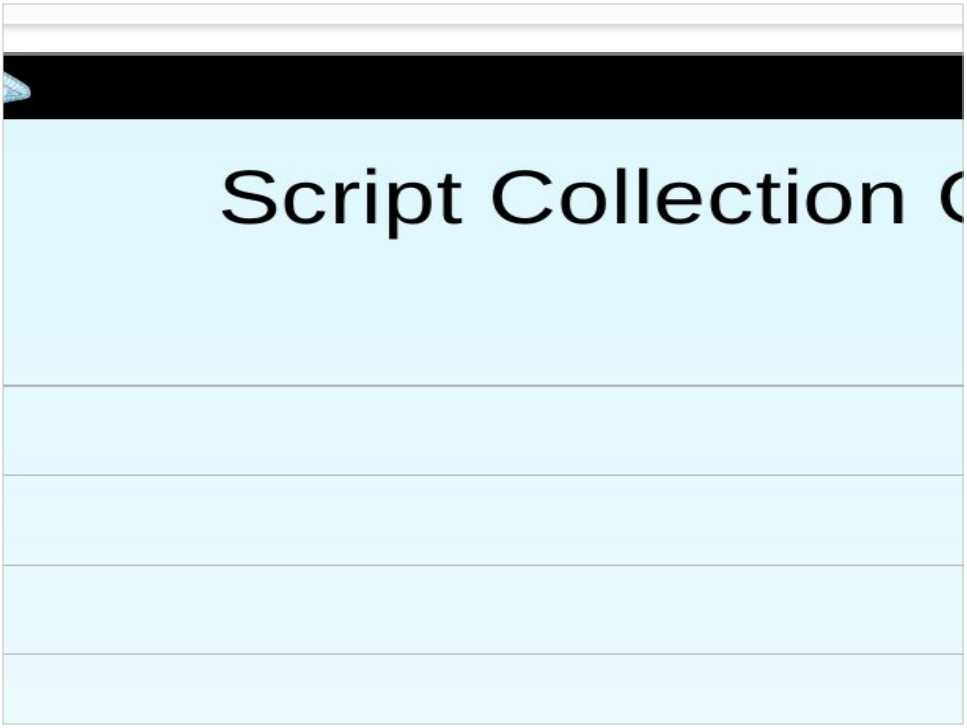
- purpose
  - detect which exploits a server is vulnerable to
  - detect whether services are run on wrong ports or share the same port
- technique:
  - connect TCP, listen 5 sec, get greeting banner, start UDP probes (specific port, string) etc.
- example
  - nmap -A -T4 -F 172.28.124.39
    - A = enables OS detection and Version detection, Script scanning and Traceroute

Remote OS Detection

- reasons:
  - determining vulnerability of target hosts
    - Rwho daemon on Solaris 7-9 is vulnerable, on Solaris 10 not
  - network inventory and support
  - detecting unauthorized and dangerous devices
    - detect a wireless access point (WAP), webcams, game console
- technique:
  - TCP/IP stack fingerprinting: send a series of valid and invalid TCP and UDP packets to the remote host and examine practically every bit in the responses
- example:
  - nmap -sV -O 172.28.124.39, nmap -O 172.28.124.39

Nmap Scripting Engine

- automate a wide variety of networking tasks
  - network discovery
  - more sophisticated version detection (e.g Skype2)
  - vulnerability detection
  - backdoor detection
- scripts are executed in parallel
- scripts are written in Lua scripting language
- example:
  - nmap -sC 172.28.124.39


Script Collection C

Why Lua?

- NSE had to be: easy to use, small in size, scalable, fast and parallelizable

- Lua is:
  - lightweight programming language
    - easy to learn, has minimalist syntax and features
    - small memory footprint
    - tiny to embed
  - safe and secure
    - no buffer overflows, no strings vulns
  - portable
  - interpreted
  - used:
    - In video game development (Warcraft, Mafia, FarCry etc)
    - Wireshark network packet analyzer, VLC media player, Snort
  - excellent documentation, actively developed

Components of NSE

- NSE libraries
  - 57 modules written in Lua
    - „bit” (bitwise op on integers) and “pcre”(perl compatible regex) are written in C/C++
  - make script writing more powerful and convenient
    - scripts need only to require the default libraries in order to use them
- NSE Scripts
  - 161 scripts: http://nmap.org/nsedoc/
- Nmap API
  - an interface to the Nmap's internal functions (Nsock library for efficient network I/O etc.)

Script categories

- every script belongs to at least one category
- auth
  - try to determine authentication credentials on the target system
- default
  - the default set and are run when using the -sC or -A options
- dos
  - may cause denial of service (they crash a service as a side effect of testing it)
- intrusive
  - the risks are too high that they will crash the target system
- safe
  - scripts that don't crash services, use large amounts of network bandwidth or other resources, or exploit security holes

- Command-line arguments
  - --script <filename>|<category>|<directory>|<expression>|all[,...]
  - --script-args: provides args to script
  - --script-trace: for debug
- Script Selection
  - supports logical operator: not, and, or
    - nmap --script "http-*"
    - nmap --script "not intrusive"
    - nmap --script "default and safe": scripts both in "default" and in "safe"
    - nmap --script "default or safe": scripts in "default" or "safe" category
  - examples:
    - nmap --script default,safe 172.28.124.37-45

Script format

- Head
  - meta information
- Rules
  - a rule is a Lua function that returns either „true“ or „false”
- Action
  - the instructions to be executed if the script's rule is evaluated to „true”

Script format: Head

- description field
- catagories field:
  - categories = {“default”, “discovery”, “safe”}
- author field
- license field
- dependencies field
  - an array containing the names of scripts that should run before this script
  - listing a script in dependencies doesn't cause that script to be run but merely forces an ordering among the scripts that are selected
    - » dependencies = {"smb-brute"}

Script format: Rules

- prerule
  - script runs before any of Nmap's scan phases
  - for tasks which don't depend on specific scan targets
- hostrule(host)
  - runs after Nmap has performed host discovery, port scanning, version detection
  - invoked once against each target host which matches its "hostrule function"
- portrule(host, port)
  - the most common Nmap script type
  - decides if a script should run against a service
    - » portrule = shortport.port_or_service(22, "ssh")
- postrule()
  - runs after Nmap has scanned all of its targets
  - useful for formatting

Script format: Action

- the instructions to be executed when the script's prerule, portrule, hostrule or postrule triggers
- NSE scripts generally only return messages when they succeed

– a script must contain one or more rules
– a „prerule“ or a „postrule“ will always evaluate to true

Script writing tutorial

- find IP address from SSH host key
- find IP address from MAC address

Final notes

- Download Nmap from: http://nmap.org/
- NSEDoc portal: http://nmap.org/nsedoc
- NSE systems docs:

  http://nmap.org/book/nse.html
- Q&A ?