HPC security for non-experts

Hinnerk Stüben

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

NEC User Group Meeting

Hamburg

12–14 June 2024

Disclaimer

- This talk is by a non-expert.

Introduction

- motivation
  - growing need for computer security
  - in 2020 there was a wave of attack on HPC systems in Germany and EU

- possible targets for theft
  - compute power (e.g. for crypto mining, cracking decryption)
  - data
    (less of a problem in fundamental science, more in applied science and engineering)
  - credentials

Background / History

- A security concept for HPC was developed in *HLRN-Verbund*.
- The concept was discussed with NEC in the negotiation phase for our new HPC cluster.

HLRN – The North German Supercomputing Alliance (in 2016)

Security philosophy

Use your brain

- "Have courage to use your own reason." *Immanuel Kant*

- be different from what others are doing
  - can be a consequence of what Kant demanded
  - can be a trick when you mostly follow the mainstream

Consider that the world is paradoxical

Example: admin PC

- naive approach
  - apply the same security measures as on an ordinary PC
  - run all OS updates
  - use a virus scanner including its automatic updates
- paradox
  - automatic scanner (and OS) updates can be a gateway for attackers
- consequence
  - an admin PC should be configured differently from a user PC

Keep Murphy's law in mind

- "Anything that can go wrong will go wrong."
- "Anything that can go wrong will go wrong, and at the worst possible time."
- "If there are two or more ways to do something and one of those results in a catastrophe, then someone will do it that way."

"2nd" Murphy law

- "Wenn etwas eigentlich nicht schief gehen kann, wird es trotzdem schief gehen."
- "If something virtually cannot go wrong, it will go wrong, too."
- "If you think that something cannot go wrong (because that is too unlikely or because that would be too stupid), it will go wrong, too."
- Never rely on the assumption that an operating error is so unlikely that it will not happen.
- example: Chernobyl disaster
- consequence: secure yourself against yourself

Keep it simple

- my main lesson from the 2nd great nuclear accident
  - things are becoming too complicated, too complex
  - obviously risk and effect of tsunamis were not considered in the planning

- consequence
  - reduce complexity / keep it to a minimum
  - reduce functionality / keep it to a minumum
  - maximizing performance can have security impacts, too

- *principle of higher simplicity* $\rightarrow$ simplicity at *all* levels
  - concept
  - usage
  - program code

- consequence
  - If easy usage is only achievable with (too) complicated code, usage must become more inconvenient,

Major problem: everything is connected with everything

- internet level
  - internet of things
  - network virtualization
- cluster level
  - cluster management software configures switches
  - RDMA (remote direct memory access)
- node level
  - BIOS / firmware can be accessed from OS
  - virtual machines
- CPU level
  - spectre etc.
- software level
  - active documents / macros
  - shared libs

Security and convenience are mutually exclusive

- one must find a balance
  - risk must be considered
  - example: is admin access from remote acceptable?

Functionality can breed problems

Examples:

- car electronics accessible from the mobile phone network
- USB: what looks like a memory stick could act as a keybord
- web access, e.g. JupyterHub

Try to keep possible damage to a minumum

- modularity
- separation
- no single point of control

Security practice

Example: non-expert view on user authentication (I)

- password
  - can be stolen via cyber attack *without* breaking into the user's computer
    - via phishing
    - on a cracked computer where the password is entered

- password + one-time password
  - can be stolen via cyber attack like a simple password
  - can be used by the attacker only a single time

Example: non-expert view on user authentication (II)

- SSH keys
  - can be stolen via cyber attack *only* if the local computer has a security weakness (e.g. a mal-functioning web browser)

- SSH keys
  - should always be protected by a passphrase
  - should never be stored on a computer that can be accessed by more than one person (e.g. a server or a PC in a pool)

Example: non-expert view on user authentication (III)

- SSH key on a *resident* hardware security token
  - *cannot* be stolen via cyber attack
  - the whole secret information is on the device an is available to a thief if no passphrase/PIN was set

- SSH key on a *non-resident* hardware security token
  - *cannot* be stolen via cyber attack
  - a thief has not gained the whole secret information even if no passphrase/PIN was set

Examples: Topics for experts

- Are good encryption algorithms employed?
  - algorithms might/will become weak over time
- Is the implementation / Linux distribution ok?
  - secrets could be guessed from execution-time measurements
  - quality of random numbers
  - recently: *xz* attack to ssh
- Is a hardware security token robust?
  - can the secret stored on it never leave it?

Main aspects of our security concept

- login hardening
  - replace password authentication by *public key authentication*
- system hardening
  - apply the *principle of least privilege* as often as possible
  - build some *defense in depth*
- disaster recovery
  - demand that system recovery is possible with 3 days

Principle of least privilege

- Examples
  - no SUID or GUID bits
  - minimal Linux image / minimal number of services
  - rescrictive export and mount options
  - non-root installation of application software

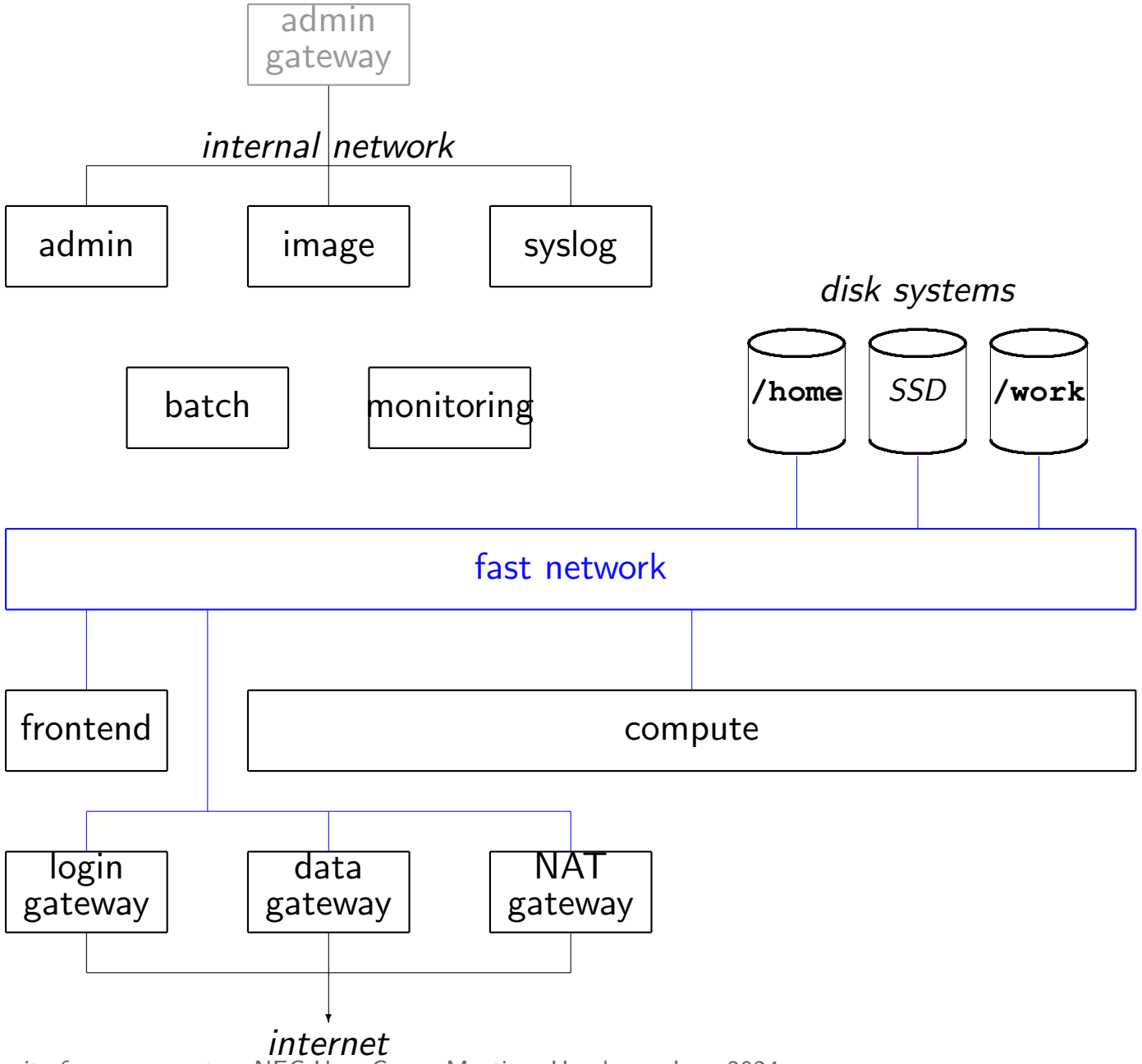
Disaster recovery

- careful generation of operating system images
- (almost) completely diskless system

Some practical guidelines

- whereever possible:
  - do not harden an access path $\rightarrow$ remove it
    (prefer a wall over a secured door)
- separation
  - use dedicated computers (or even infrastructure) for system administration
- principle of least functionality ("keep it simple")
  - minimal software installation on admin computers
- user and admin training

HPC cluster configuration



Foreseeable problems

- file transfer between HPC systems
  - one should never get shell access from a computer used by more that one person
- web applications opening ports to the world
- application software that is downloading and directly using code/libs from the internet