Mag. iur. Dr. techn. Michael Sonntag

Collecting information

Institute for Information Processing and
Microprocessor Technology (FIM)
Johannes Kepler University Linz, Austria

E-Mail: sonntag@fim.uni-linz.ac.at
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

Agenda

- NMap
- Google Hacking
  - → Special search operators
  - → Google cache
  - → Robots.txt

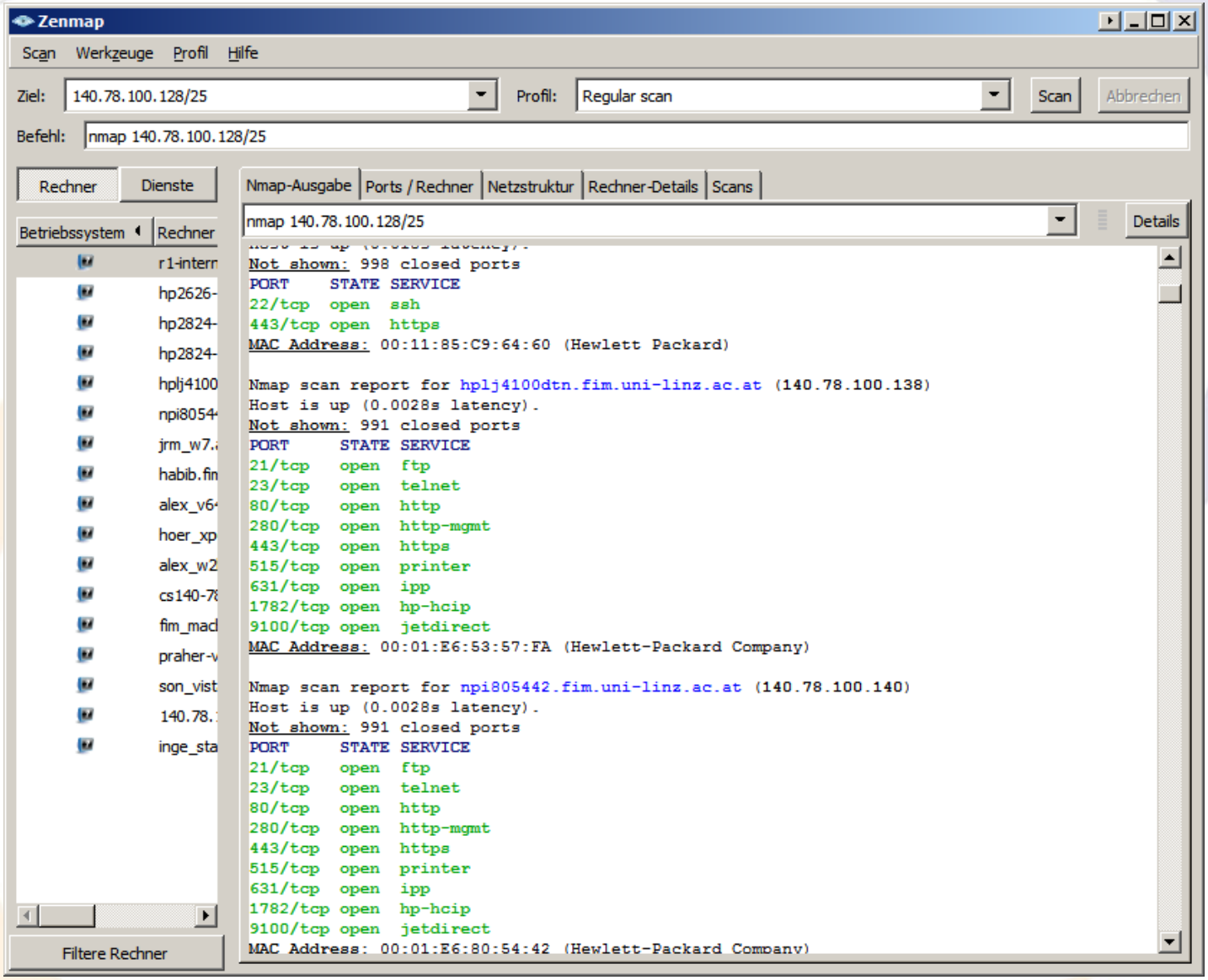
NMap

- NMap (Network MAPper) is a network scanner
  - → It tries to find all computers in a specific network and checks what ports are open, what OS they are running, whether there is a firewall, etc.
- It does not look for specific vulnerabilities!
  - → But it gives recommendations; e.g. services to disable
  - → Some scans + vuln. systems → Lock-up/crash!
- Used as a tool for inventory generation in a network
  - → Are there any computers which should not be there?
  - → Can also be used to gather information for a later attack
    - » Which OS/software and which version is running
- Stages: 1 = Host discovery, 2 = Port scan, 3 = Service/ version detection, 4 = OS detection, 5 = Scripting
  - → Scripting may also include vulnerability/malware detection!

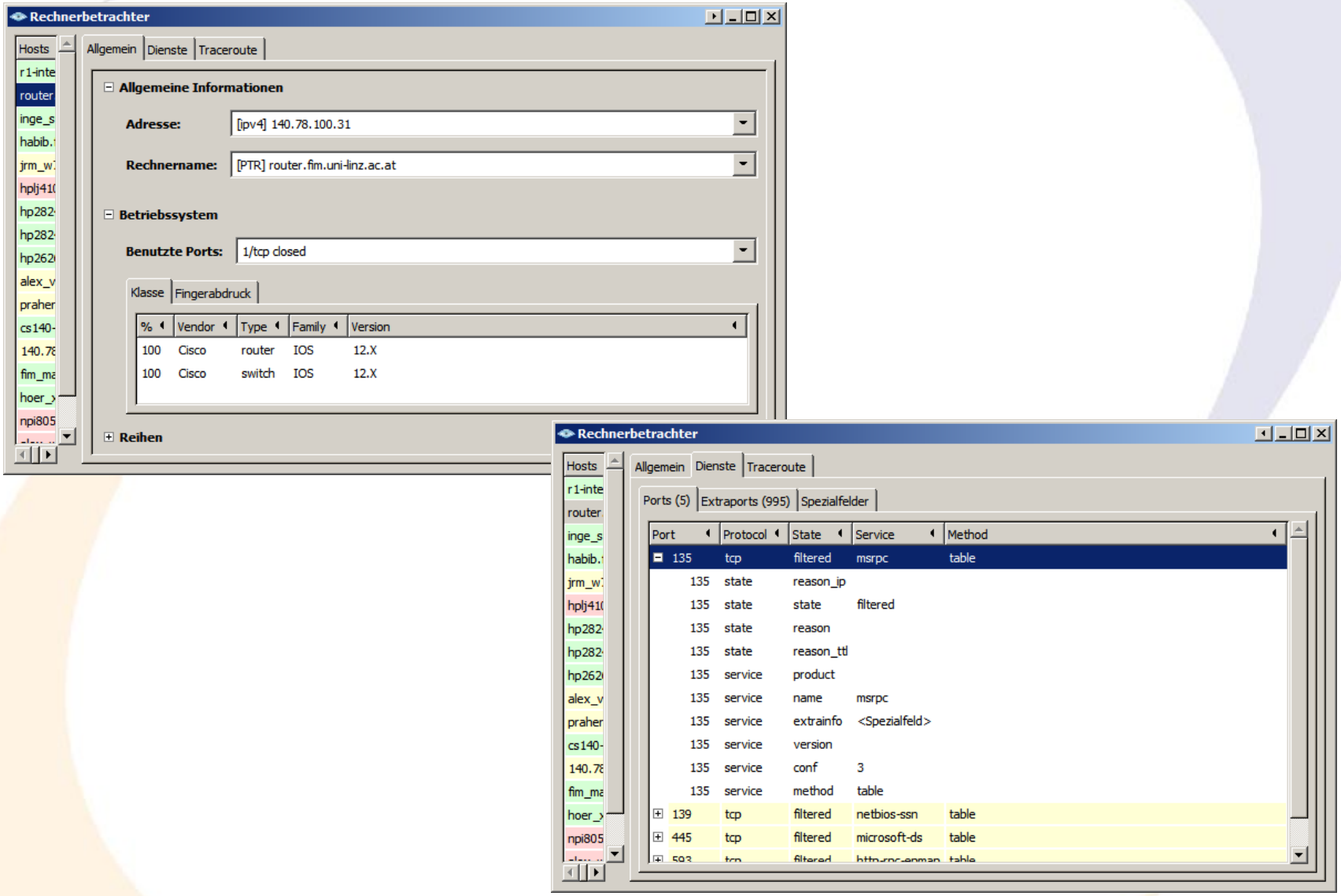
NMap

- Usage:
  - → Start program and enter IP address
  - → Select profile for scanning
    - » Special options only available in the command line version or when constructing a new profile!
- Your tasks:
  - → Install NMap (+ the UI – Zenmap)
  - → Scan the local subnet for hosts
    - » Use a “Quick scan"
  - → Scan the machine of your neighbour
    - » Use a “Regular scan"
  - → Interpret the results
    - » Correct output?
    - » Something surprising/dangerous found?

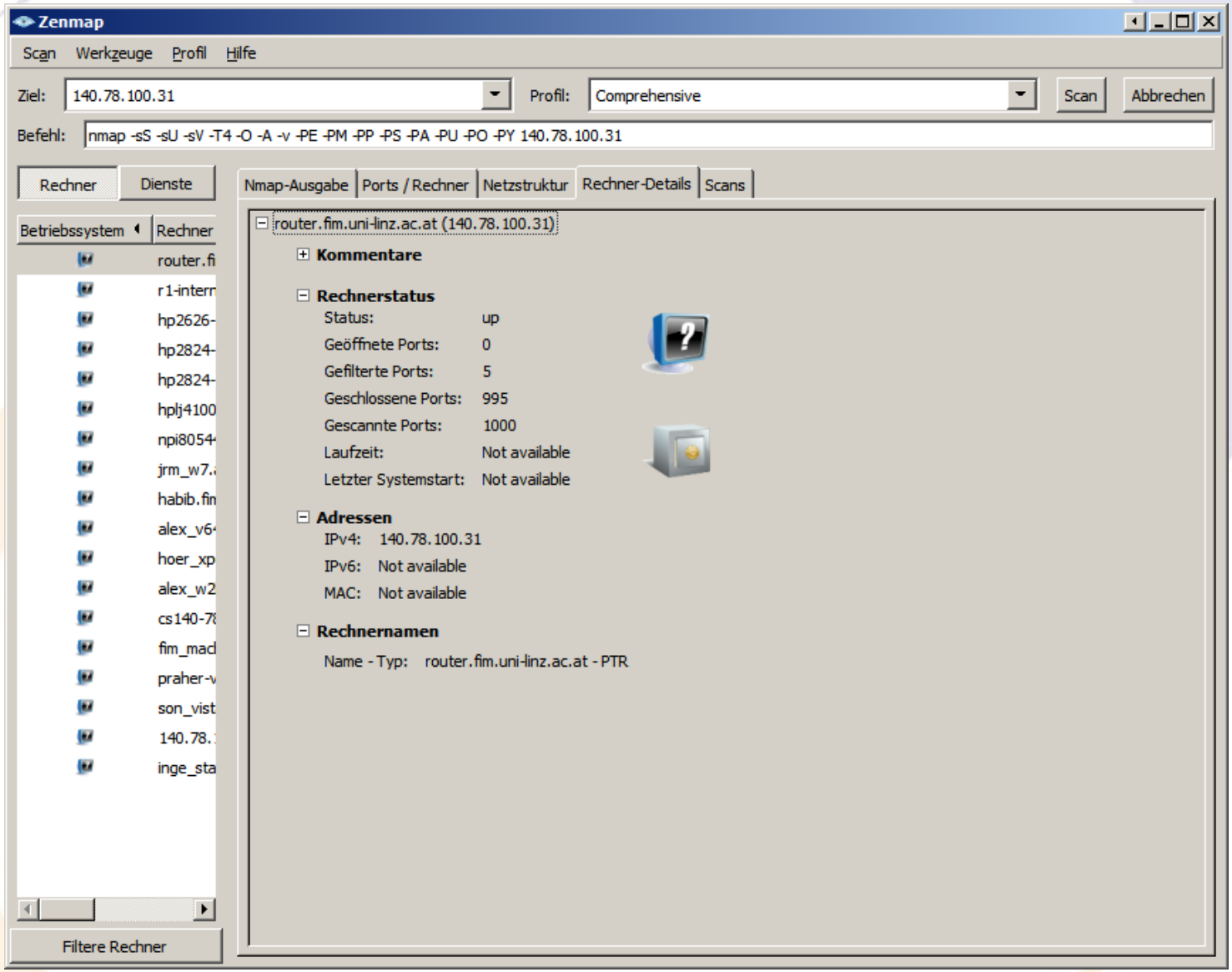
Sample result: NMap local subnet scan



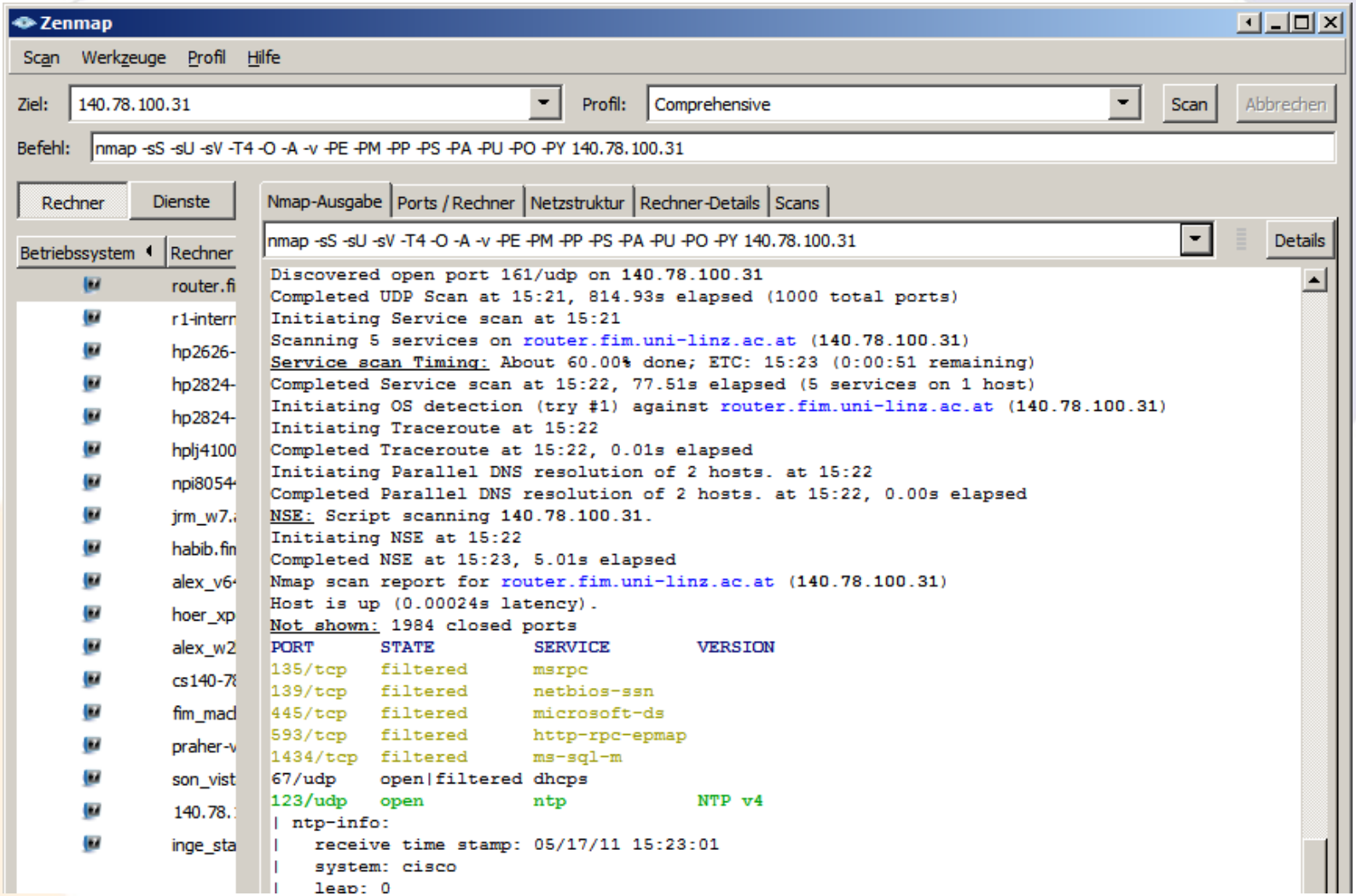
Sample result: NMap info



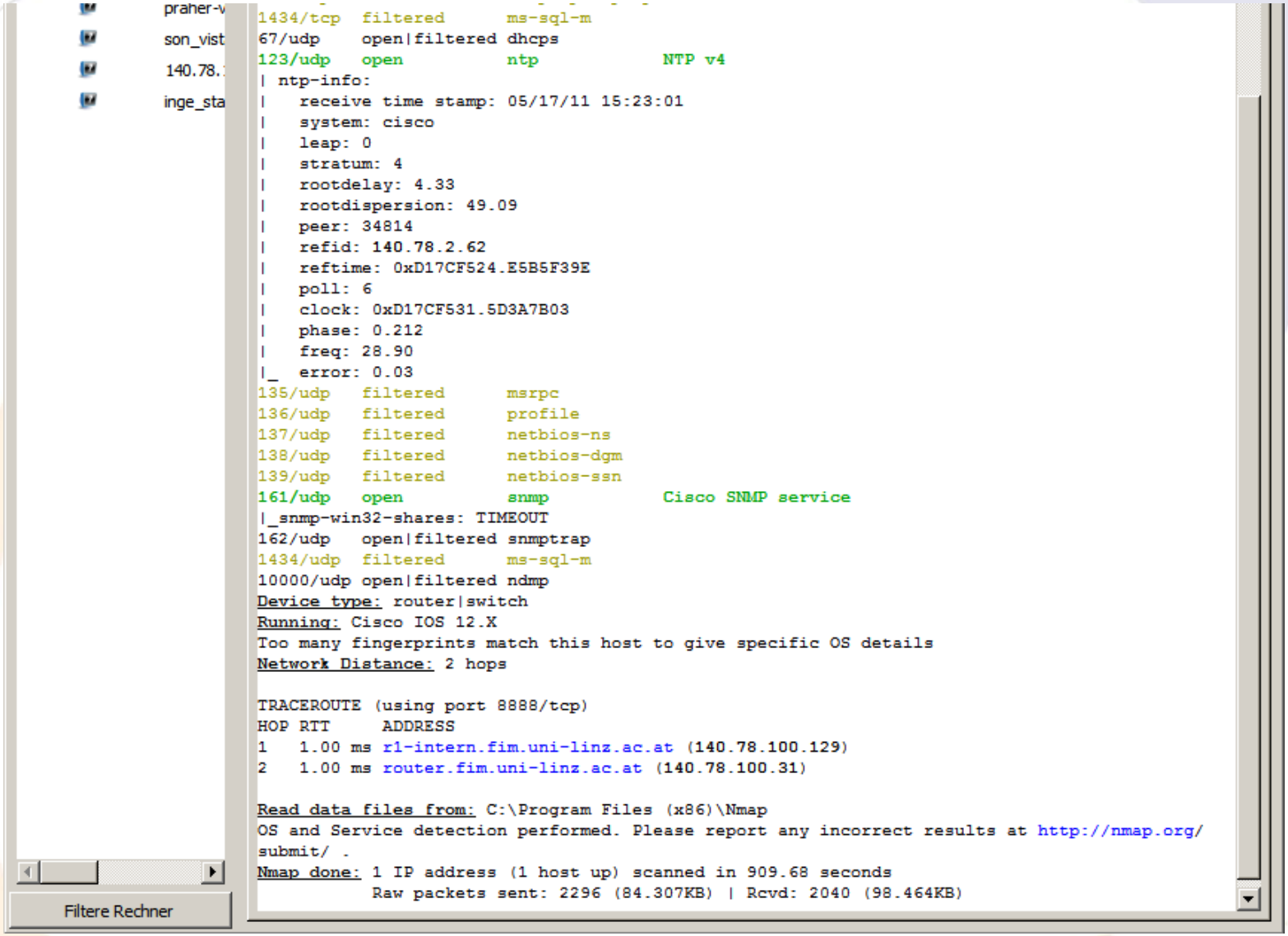
Sample result: NMap info



Sample result: NMap info



Sample result: NMap info



Google hacking

- Not an attack as such, but the preliminaries: Searching for vulnerable systems or vulnerabilities on a site
  - → Using a search engine to look for known weaknesses
- Examples:
  - → Looking for version numbers (vulnerable versions of software are known; websites running them will be prime subjects!)
  - → Looking for "weak" code → "Google Code Search"
  - → Search program comments indicating problems
    - » Like: /* TODO: Fix security problems */
- Note: The subject of the attack has no chance at all of noticing this, as his server is not touched at all!
  - → Attacks come "out of the blue"
    - » But not unprepared: Only pages existing for a “long” time (typical indexing time: 2-3 weeks!) can be found
    - » Usually the vulnerability is older too

Google hacking

- Requires advanced Google operators:
  - → link: Search within hyperlinks
    - » With certain words hinting at interesting pages
  - → cache: Displays the page as it was indexed by Google
    - » Turn off image loading and you will not be logged on the server!
  - → intitle: Within the title tag
    - » Directory listings: intitle:index.of
      - – Better: intitle:index.of “parent directory”; intitle:index.of name size
  - → inurl: Within the URL of the web page
    - » Webcams: inurl:"ViewerFrame?Mode=" inurl:"/axis-cgi/jpg/image.cgi?"
  - → filetype: Only files of a specific type (no colon → filetype:doc)
    - » MS SQL server error: "A syntax error has occurred" filetype:ihtml
- Note: Such operators exist for most search engines
  - → This is **not** a Google-specific problem!

Google Hacking: General targets

- Looking for specific vulnerabilities
  - → Version numbers, strings, URLs, …
- Error messages with too much information
  - → Before “lockdown”, which logs errors and shows a simple message to the user only
- Files containing passwords
  - → For offline breaking
- Logon pages
  - → Where to actually attack
  - → Title/content may give away information about limitations to passwords, method of storage, security precautions, …
- Vulnerability information
  - → All kinds of logs (web servers, firewalls, …)
  - → May also contain information about the internal network

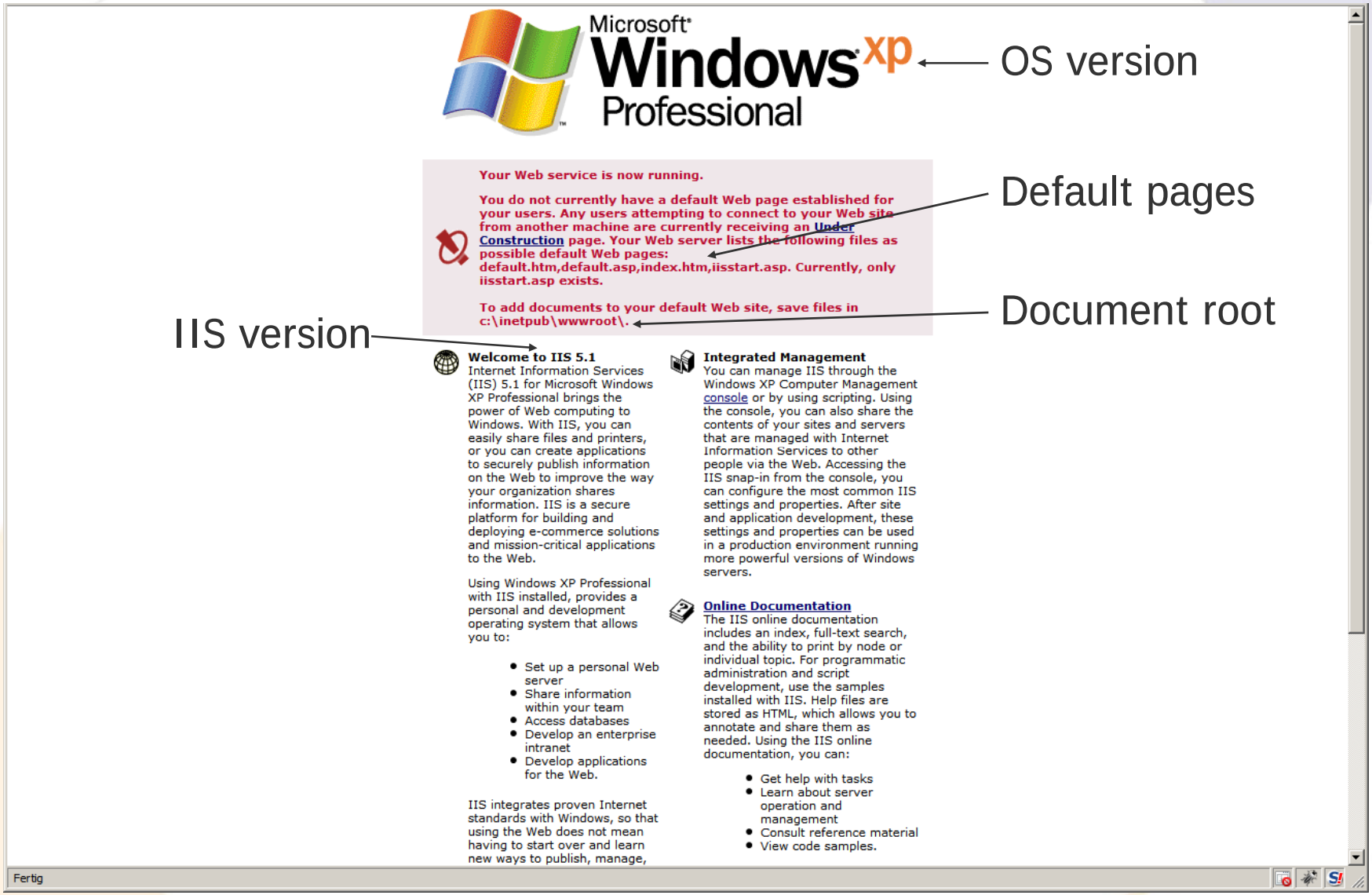
Google hacking: Examples

- Searching for password lists (very old vulnerabilities!):
  - → inurl:/_vti_pvt/users.pwd
  - → inurl:/_vti_pvt/administrators.pwd
  - → inurl:/_vti_pvt/service.pwd
  - → Still requires to break passwords, but this can be done offline!
- HP JetDirect: Printers with an included web server
  - → inurl:hp/device/this.LCDispatcher
    - » Note: These web pages typically cannot be changed at all!
    - » Only access can (and should!) be impossible form the Internet
  - → Searching by title (model numbers) or strings (handbook, questions, …) would not be successful here!
- Login portals of routers
  - → intitle:"Cisco Systems, Inc. VPN 3000 Concentrator“
  - → Only shows where to attack; passwords must still be guessed!
    - » But: Try passwords of producer; often the same for all appliances

Google hacking: Examples

- VNC viewers (Java client: Port 5800; server: Port 5900):
  - → intitle:VNC inurl:5800
    - » Depending on page title the version/product can be distinguished
- Webcams (Axis):
  - → intitle:"Live View / - AXIS"
    - » Title can be used for further restriction, e.g. the model used
- Server version:
  - → intitle:index.of server.at
    - » Example result at bottom of page: “Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g Server at www.????? Port 80”
      - – mod_ssl/OpenSSL version might also be **very** interesting!
  - → Also the default test pages (after installation) often remain accessible even after installing the final website
    - » intitle:welcome.to intitle:internet IIS (see next slide!)
- Looking for know-vulnerable cgi files
  - → inurl:/random_banner/index.cgi

intitle:welcome.to intitle:internet IIS



Google hacking: Examples

- MySQL database dumps
  - → "# Dumping data for table (username|user|users|password)" -site:mysql.com -cvs
- phpMyAdmin: Database administration tools
  - → intitle:phpMyAdmin “Welcome to phpMyAdmin ***” “running on * as root@*”
- Registry dumps
  - → filetype:reg reg HKEY_CURRENT_USER username
- Looking for code/passwords (often contains cleartext pwds!)
  - → filetype:inc intext:mysql_connect
- Printers/Faxes:
  - → inurl:webArch/mainFrame.cgi
- UPS:
  - → intitle:"ups status page"

Google hacking: Cache

- The cache gives you access to old/removed content
  - → Which might still be applicable!
- Attention: Surfing the cache will still touch the server
  - → E.g. images are loaded from the “source”
- Way around: View the text-only version
  - → Add “&strip=1” to the search URL

Google Hacking: Prevention

- Make sure that “private” computers are not accessible from the “public” internet
  - → Use a firewall (packet filter alone might be insufficient)
- Automated tools available : E.g. SiteDigger
  - → Can also be used on your own pages to look for "weaknesses“ (verification)!
- Check what Google (and others) know about your site
  - → site:www.mysite.com
  - → Is this **only** what **should** be accessible to everyone?
- Use "robots.txt" to limit web crawlers to "relevant" pages
- Captchas/Remove from Google index (→ Desirable?)
  - → Not that easy and/or quick!
  - → Requires often extensive measures (removal of page + notification of Google + wait for index)

Google hacking: Legal aspects

- The site is not attacked at all in this stage
  - → Just some information is collected
  - → The information is gathered from public sources
- In contrast to other attacks, this is legal in most countries!
  - → Too far away from a concrete attack
    - » When trying it out on the real server (even if unsuccessful!), this is typically a punishable offence!
  - → Note: UK and USA are notable exception!
    - » "Unauthorized access" is an offence
- BUT: If something happens, this can be used as evidence
  - → Also, it is a very good evidence to prove intentionality
    - » When explicitly looking for weaknesses, you can later hardly claim that you sent a special request "accidentally" …
  - → Note, that finding evidence of Google hacking is difficult
    - » Requires access to your computer or log files of intermediaries (like proxies, wiretapping at the ISP, …)

Google hacking: Tasks

- Try out several of the examples before
  - → E.g. webcams or database examples
  - → Do they always work? What could be the reason?
- Access the Google cache for a website you know to be changing frequently
  - → Check the differences to the current website
  - → How old is the cached version?
    - » Approximately or can you identify the exact date?
  - → Where do external links lead to?
    - » Archived version or live version?
  - → Where are images loaded from?
    - » What difference can this make?
- Bonus task:
  - → What is the “web archive”?
  - → How is it similar to Google cache and what’s the difference?

Robots.txt

- Robot Exclusion Standard
  - → Asking nicely to leave your site alone
    - » "Good" robots check for this file and adhere to it
    - » But technically there is no need!
    - » Example: Austrian National Library has the legal permission to archive website with strong connection to Austria → Ignores this file deliberately (legal permission + obligation!)
  - → No official standard available!
  - → Note: Crawling; indexing is different!
  - → Must reside in the root directory of the site
- Alternative: META tags within the site
  - → Drawbacks:
    - » Robot has already retrieved the site
    - » Works only for HTML pages
  - → Advantage: Local control!
    - » Robots.txt is possible only site-wide!

Robots.txt

- What robots.txt is **NOT**:
  - → A security measure: Anyone can access any page
    - » Retrieving robots.txt is no requirement!
    - » Use password, authentication, … instead
  - → A way of hiding data
    - » The location/its name is publicly visible
  - → A tool to prevent indexing
    - » External URLs may still result in indexing
- What robots.txt **IS**:
  - → A way to reduce the server load and the quality of search results by excluding uninteresting parts of the site
    - » Or those changing too frequently to be useful within the index
  - → A way of providing information about the sitemap

Robots.txt: Difficulties of later removing content

- Adding it to robots.txt
  - → URL is known, so it is accessed and indexed
    - » Or: External links to the site → Again being indexed!
  - → Only from the pages with the links, those links are ignored and are not followed
    - » At some time they might fall out of of the index (several month)
- Potential solution: Add META-Tags
  - → Problem: Doesn't work for .doc, .pdf, …
  - → But then these files MUST NOT be in the robots.txt!
    - » Must be allowed in robots.txt and individually excluded
- Real solution:
  - → Use Google webmaster tools to remove
  - → Use X-Robots-Tag for non-HTML file types
    - » Note: This is a HTTP header! Requires webserver configuration!

Robots.txt: Format

- Simple text file in the website root: "**/**robots.txt"
  - → Attention: Might be case-sensitive (implement.-dependent)
- "User-agent: " For which bot the following lines are intended
  - → Note: Find out first, which one you want to block
    - » Google: "Googlebot", "Googlebot-Image", …
    - » Yahoo: "yahoo-slurp"
    - » Microsoft: "msnbot" (MSN search), "psbot" (images)
    - » "*" as wildcard for all bots
- "Disallow: " What may not be followed
- "Allow: " What may be followed (exceptions from Disallow)
  - → This is no guarantee and doesn't force the bot to follow links!
- "Crawl-delay": How many seconds to wait between retrieves
  - → Note: Google does not follow this (→ Use webmaster tools!)
- "Sitemap: " URL of the sitemap
  - → Only selected bots (Google, Yahoo, MSN, …)

Robots.txt: Format

- Format for Disallow and Allow:
  - → Empty: Ignore it
    - » Example: “Disallow: “ → Whole page may be crawled
  - → Everything starting with the string provided
    - » Example: “Disallow: /” → Nothing may be crawled
    - » Example: “Disallow: /index” → Will not crawl:
      - – “/index” as a file or a directory
      - – “/index.htm”, “/index.html”: Files
      - – “/indexing/”, “/index/”: Directories
  - → “$” end of line anchor
    - » Only Google, Yahoo, MSN
    - » Example: “Disallow: /*.pdf$” → Will not crawl pdf files
    - » Attention: No regular expressions allowed!
- Each command must be a separate line
- At least one “Disallow” line is required
- Empty line before 2nd, 3rd, … User-agent line

Robots.txt

- Example of “hiding” the complete site (= no crawling)
  - → User-agent: *
    Disallow: /
- Example of typical exclusions:
  - → User-agent: *
    Disallow: /cgi-bin/
    Disallow: /tmp/
- Example of allowing only Google, but not Google images
  - → User-agent: Googlebot ← Includes “Googlebot-Mobile”
    Disallow:

    User-agent: Googlebot-Image
    Disallow: /

    User-agent: *
    Disallow: /

Robots.txt Tasks

- Create a robots.txt file with the following restrictions:
  - → Allow Google, Yahoo and MSN access to the whole site
  - → No access for image searching by anyone
  - → No archiving by the web archive
  - → No access to the directory “/news/today/”, but allow access to the subdirectory “/news/today/hot/”
  - → No crawling of Microsoft Office documents
- Check whether these restrictions are possible at all
  - → And whether they are possible with robots.txt
  - → Or how they must be specified more exactly
- Find a verification tool and check your file with it

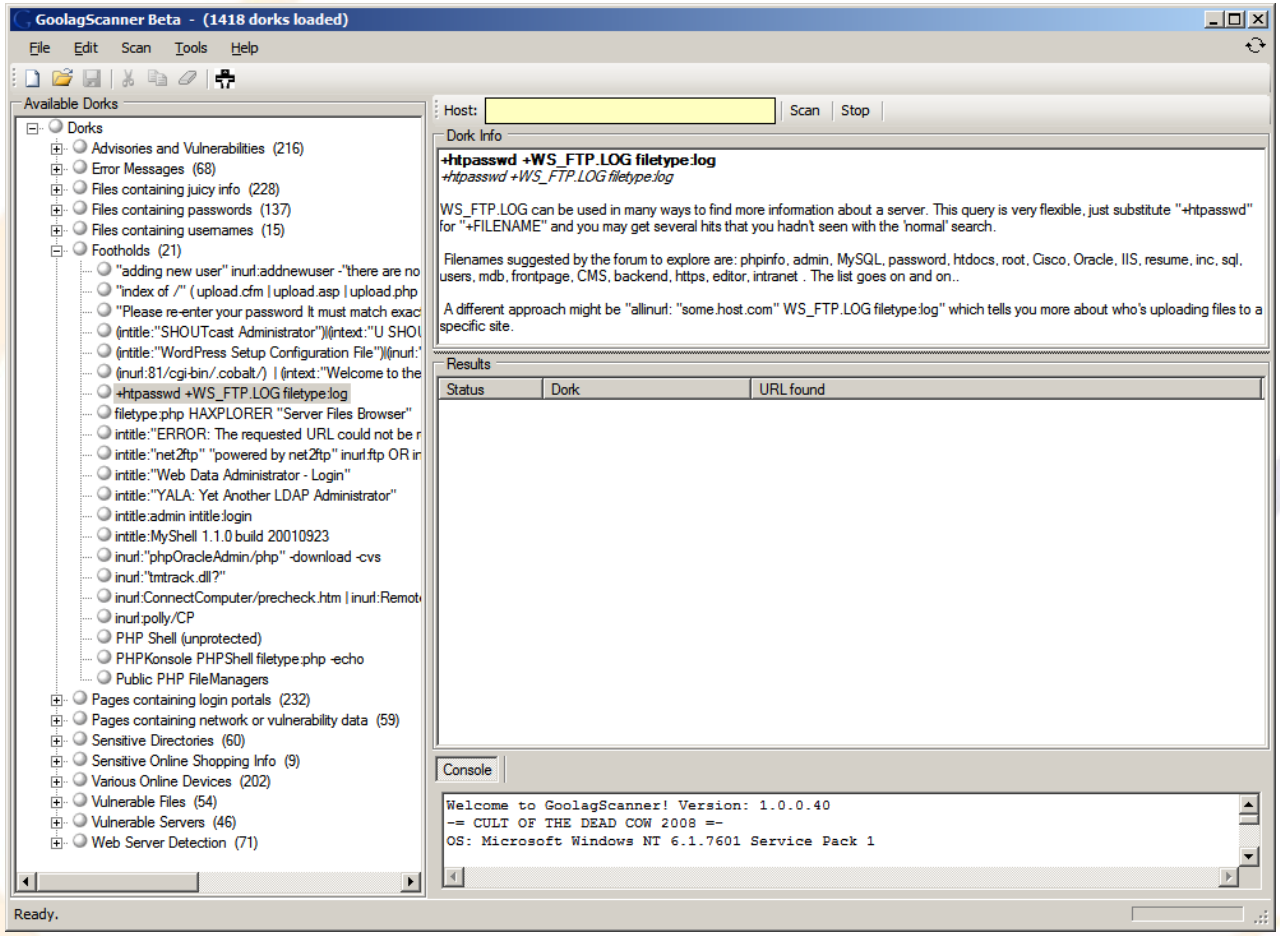
Robots.txt Solution

```
User-agent: Googlebot
Disallow:

User-agent: Yahoo-slurp
Disallow:

User-agent: msnbot
Disallow:

User-agent: Googlebot-Image
Disallow: /

User-agent: psbot
Disallow: /

User-agent: archive.org_bot
Disallow: /

User-agent: *
Disallow: /news/today/
Allow: /news/today/hot/
Disallow: /*.doc$
Disallow: /*.xls$
Disallow: /*.ppt$
Disallow: /*.docx$
```

- Attention: Restrictions exist!
  - → /news/today/ … will not apply to Google, Yahoo & MSN
    - » Or they would have to be added above!
    - » A problem of the specification too!
  - → “Microsoft Office documents” is too unspecific; only individual files (filename!) can be blocked
    - » Here only a few are shown; more exist!
  - → Empty Disallow is seen as illegal by many verifiers
    - » Can be replaced by “Allow: /”
  - → Wildcards are not supported universally
    - » *, $ will not work for all bots
    - » HTTP headers required for them

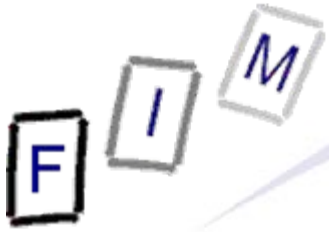
Goolag scanner

- Can be downloaded from the Internet
  - → Contains a very large number of interesting Google scans



Conclusions

- Collecting information from third-party sites is very advantageous to attackers
  - → The target website cannot notice anything suspicious
    - » It is not contacted in any way
- NMap gives a rough overview; but take care of logging
  - → Better used once “inside” or generally from outside
  - → Intense scanning is a hint of an attack
- Both are very “unreliable” as they will usually not give very useful information on a specific target system
  - → More interesting for finding “something” to hack
- “General reconnaissance” tools!



Questions?

Thank you for your attention!

Literature/Links

- NMap
  http://nmap.org/
- Robots Database
  http://www.robotstxt.org/db.html