the hacker's choice

Professional Application Mapping

van Hauser, THC

vh@thc.org

http://www.thc.org

Contents

About THC

History

- Founded on 1st October 1995 by joining Drunken Traders Inc. and LORE BBS
- First we came up with a cool acronym (THC) and then thought about what it could mean.
- We finally agreed on "The Hacker's Choice"
- Hey, we were kids back then ☺
- We were and still are a release group. Who wants to join has to release something pretty cool under the THC label.

About THC

Today

- No one of us is breaking into systems, or committing other computer crimes.
- Wide scope of interest:
  - Network Security/Hacking >> parasite, hydra, flood, probe, **gg**
  - Unix Security/Hacking >> unix-hacking-toolkit
  - Windows Security/Hacking >> ipf, happybrowser, cupass
  - Application Security/Hacking >> **amap**, **vmap**, ra-bbs-hack
  - Credit Card generation/verifying tools >> thc-cred, thc-shagg
  - Wardialing >> thc-scan
  - Wardriving >> wardrive, thc-rut
  - Phreaking >> pbxhack, gd, login hacker
  - Cryptography/Anonymity/Authentication >> passid, fuzzyfingerprints, anon unix
  - Trojans and Backdoors >> ra-bbs, rwwwshell
  - Exploits >> realserv, lpset, thc-sql etc.
  - Ethical articles >> hackers go corporate, human2hacker
  - … and in old times also anarchy and virus stuff … examine our magazines!

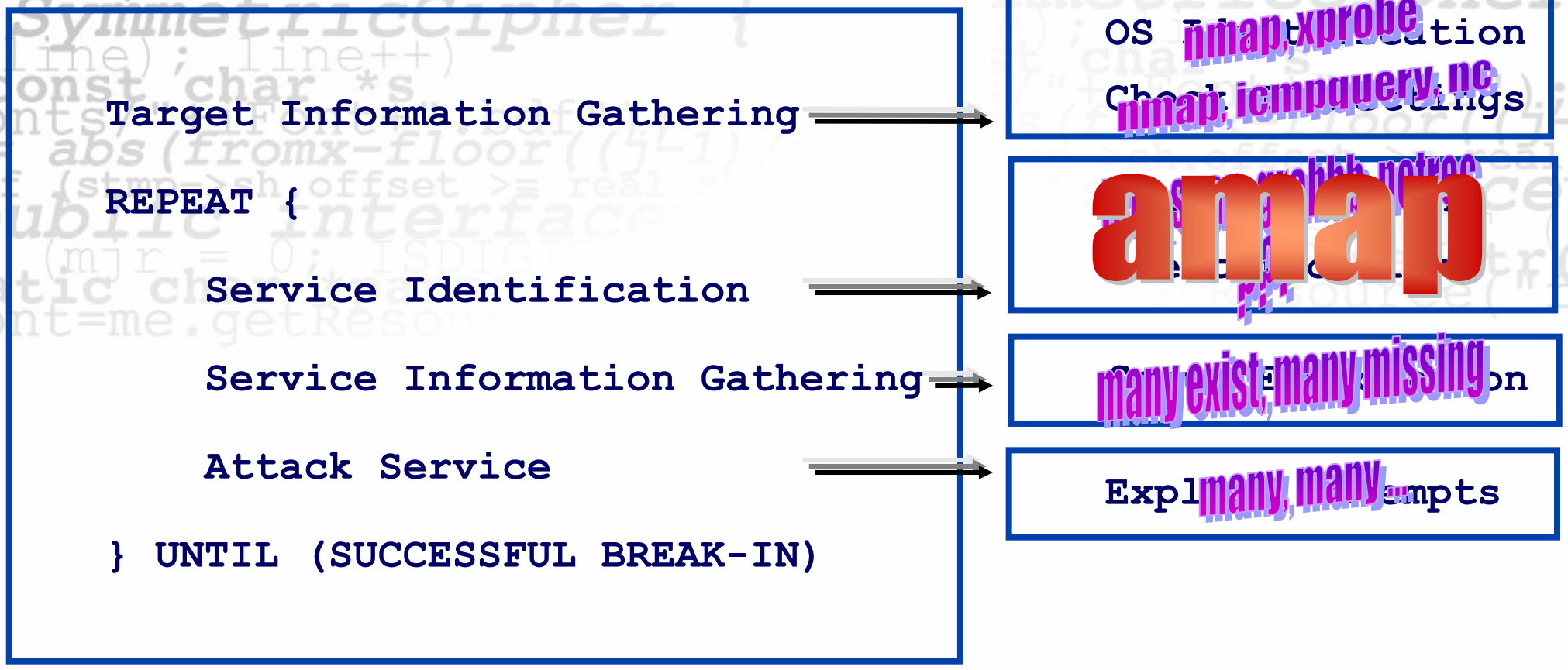
About THC

Our Web Page

- Has got all our tools (28!), articles (32!) and exploits (7) online.
- A great forum to ask and respond to questions
- And: a cool online security quiz changing every two weeks, with high-scores and prices

Visit us at http://www.thc.org

The common way a pentester works



The problems of application identification

- There are many, many, many protocols
- Banner grabbing can only identify plain protocols, e.g. FTP, HTTP, SMTP
- Several protocols only send a response when the client sends the correct initiation.
- All such protocols need of course DIFFERENT protocol initiation strings ☹
- Therefore, identifying an SAP R/3 or Oracle installation on an unusual port is unfeasible …
- … until now ☺

How does amap work

- Amap includes two important files:
  - A trigger file: appdefs.trig
  - A response file: appdefs.resp
- These two ascii text files contain the information to identify protocols
- Every trigger in the appdefs.trig file is sent to every port to be identified (TCP/UDP and harmful/harmless specifications apply)
- All responses received are matched against patters in the appdefs.resp file
- (oh yes, there's also appdefs.rpc, which is for RPC identification)

Massive application mapping with amap

Amap allows you to:

- Perform parallel protocol identifications
  - Probe ports in parallel
  - Probe hosts in parallel
- Currently has got a database of
  - 25 triggers (client protocol initiation strings)
  - 250 responses (server protocol initiation/answer strings)
  - 450 RPC IDs
- Can identify services which have an SSL front-end
- Can identify exact RPC service type and version
- Can use nmap port scan ouput files
- Can output the results in machine readable files
- Contains a tool to brute force valid client initiation triggers (which can also be used to find DOS and overflow problems)

How to use amap

```
Syntax: amap [-USRHbdruv]1 [-o <file> [-m]] [-D <file>] [-t sec] [-T cons]
        [-p PROTO ] [-i <file>] [TARGET PORT [PORT] [PORT] ...]
Options:
    -1          Only send triggers to a port until 1st identification. Speeeeed!
    -i FILE     Nmap machine readable outputfile to read ports from
    -S          Do NOT look behind an SSL port
    -R          Do NOT identify RPC service
    -H          Do NOT send application triggers marked as potentially harmful
    -u          Do NOT dump unrecognised responses (better for scripting)
    -U          Ports specified on commandline are UDP (default is TCP)
    -v          verbose mode, use twice for debug (not recommended :-)
    -o FILE     Write output to file FILE
    -m          Make output to file (-o) machine-readable (colon-separated list)
    -d          Dump hex traffic (only if a response is received)
    -b          Print ascii banner of responses
    -T CONS     Amount of parallel connections to make (default %d, max %d)
    -p PROTO    Only test for applicational protocol PROTO (i.e.: ftp)
    -t SEC      Response timeout, wait longer on slow connections (default %d)
    -D FILE     Read from Definitions FILE[.trig|.resp|.rpc] instead of default
    -h          Print this shit
    TARGET PORT   The target address and port(s) to scan (additional to -i)
```

How does amap work

```
laptop:/tmp # amap -v -i camp.nmap
Using nmap file camp.nmap ... done
Using trigger file /usr/local/bin/appdefs.trig ... loaded 22 triggers
Using response file /usr/local/bin/appdefs.resp ... loaded 246 responses
Using trigger file /usr/local/bin/appdefs.rpc ... loaded 450 triggers

amap v4.2 (www.thc.org) started at 2003-08-08 16:51:26 - APPLICATION MAP mode

Total amount of tasks to perform in plain connect mode: 208
Protocol on 81.161.148.217:139/tcp (by trigger http) matches netbios-session
Protocol on 81.161.149.206:37/tcp (by trigger http) matches time
Protocol on 81.161.149.206:13/tcp (by trigger http) matches daytime-unix
Protocol on 81.161.149.206:22/tcp (by trigger http) matches ssh-openssh
Protocol on 81.161.149.206:113/tcp (by trigger http) matches auth
Protocol on 81.161.149.206:515/tcp (by trigger http) matches lpd
Protocol on 81.161.149.206:21/tcp (by trigger ftp) matches ftp
Protocol on 81.161.148.217:1025/tcp (by trigger ms-ds) matches netbios-session
Protocol on 81.161.148.217:135/tcp (by trigger ms-ds) matches netbios-session
Protocol on 81.161.149.206:1024/tcp (by trigger rpc) matches rpc
Protocol on 81.161.149.206:2049/tcp (by trigger rpc) matches rpc
Total amount of tasks to perform in RPC connect mode: 900
Protocol on 81.161.149.206:2049/tcp matches rpc-nfs-v3
Protocol on 81.161.149.206:1024/tcp matches rpc-nlockmgr-v4

Unidentified ports: 81.161.149.206:9/tcp 81.161.149.206:111/tcp (total 2).

amap v4.2 finnished at 2003-08-08 16:51:50
laptop:/tmp #
```

How does amap work

The format of the "appdefs.trig" file:

NAME:[COMMON_PORT]:[IP_PROTOCOL]:0|1:TRIGGER_STRING

- NAME: The name of the protocol
- COMMON PORT: default port for protocol (unused currently)
- IP PROTOCOL: TCP or UDP or empty (for both)
- '0' or '1': 1 - Trigger can crash applications, 0 - harmless trigger
- TRIGGER STRING: "TEXT\r\n" ascii string or 0x012345678 hex string

Examples of the triggers:

FTP:21:TCP:0:"USER AMAP\r\n"

- The trigger for FTP sends the string "USER AMAP" followed by a carriage return and linefeed to all TCP ports and is not considered harmful

EVIL:::1:0x00000000000000

- The trigger for EVIL sends 7 null bytes to any TCP or UDP port, and is considered harmful

How does amap work

The format of the "appdefs.resp" file:

NAME:[TRIGGER]:[IP_PROTOCOL]:[MIN,MAX LENGTH]:REGEX

- NAME: The name of the protocol
- TRIGGER: Identification applies only if the response was triggered by the defined trigger (appdefs.trig)
- IP PROTOCOL: Identification applies only if the protocol is UDP/TCP
- MIN, MAX: Identification applies only if size matches min/max definition
- REGEX: Identification applies only if response content is matched by regex definition

Examples of the responses:

FTP::TCP::^220.*\n331

- any response which starts with "220" and later on has got the output "331" after a newline is identified as FTP

SSL:SSL:TCP::^0x1603

- Any response which starts with the hexadecimal bytes 16 and 03 is identified as SSL if it was triggered by the SSL trigger

How does amap work

STATISTICS

- In normal mode, 1 port needs ca. 1 second for identification
- In fast (-1) mode, 1 port needs ca. 1/10th second for identification

Add up to 5 seconds for program start-up, waiting for replies and shutting down.

How to use amap

Options you should enable when running amap:

- -b – to see the banners received
- -1 – if you want speeeeed
- -mo FILE – to generate a machine readable output

Use amap with "-i nmap_outputfile" or "127.0.0.1 23"

If you get unknown responses, which look like this:

```
Unrecognized response from 81.161.148.208:8900/tcp received.
Send this output and the name of the application to amap-dev@thc.org:
0000:  5e2f 312e 3020 5673 6666 2052 4541 4459    [ ^/1.0 Vsff READY ]
```

, identify the application(s) and send the information to:

amap-dev@thc.org

How to use amapcrap

```
Syntax: amapcrap [-S] [-u] [-m 0ab] [-n connects] [-N delay] [-w delay] [-e]
                 [-v] TARGET PORT

Options:
  -S            use SSL after TCP connect (not usable with -u)
  -u            use UDP protocol (default: TCP) (not usable with -c)
  -n connects   maximum number of connects (default: unlimited)
  -N delay      delay between connects in ms (default: 0)
  -w delay      delay before closing the port (default: 250)
  -e            do NOT stop when a response was made by the server
  -v            verbose mode
  -m 0ab        send as random crap:0-nullbytes, a-letters+spaces, b-binary
  -M min,max    minimum and maximum length of random crap
  TARGET PORT   target (ip or dns) and port to send random crap
```

How to use amapcrap

Options you should enable when running amapcrap:

- -m b – to send just binary strings

Let it run until it reports a success and put the lines it recommends to the corresponding files. Then amap is able to identify these applications.

If you identify new applications, send the information to amap-dev@thc.org.

How does amapcrap work

```
laptop:/prg # amapcrap -m b 81.161.149.250 139
# Starting AmapCrap on 81.161.149.250 port 139
# Writing a "+" for every 10 connect attempts
#

# Put this line into appdefs.trig:
PROTOCOL_NAME::tcp:0:0x916e61714251c1d6d281016b0e9018a5b285829d8ff274527ffedc586
5628ff6d0f168134329ea15aaeb81b87c995e2f1fe1ccaed34001533fde

# Put this line into appdefs.resp:
PROTOCOL_NAME::tcp::0x830000018f

laptop:/prg #
```

Automated post processing

```
$ cat attack-script.sh
#!/bin/sh
test -z "$1" && { echo "Syntax: $0 TARGET"; exit 1; }

nmap -oM nmap.out -p 1-65535 -sSU $1
amap -i nmap.out -mo amap.out > /dev/null 2>&1

# MYSQL attack
for i in `grep ":mysql:" amap.out`; do
    TARGET=`echo "$i" | awk -F: '{print $1}'`
      PORT=`echo "$i" | awk -F: '{print $2}'`
    mysql_sploit -p $PORT $TARGET
done

# Oracle attack...
# FTP attack...
```

End – conclusion

- Amap makes protocol identifications very easy
- This enhances the quality and success of penetration tests
- Identification of some protocols is hard
- The more binary protocols are out there, the more triggers exist, and have to be sent to applications, the longer it takes
- A few protocols can't be identified by their output

Join our amap mailing list, were we regulary publish beta version:

amap-subscribe@thc.org

End – the future

- More, more, more protocol identifications
- Will hopefully be added to Nessus soon (like Hydra ☺)
- We did not come up with many missing features ☺
  So: give us your input!

Where to get it?

V4.2 released today at:

http://www.thc.org

Questions?

Grenzgaenger – Crossing the boarder

van Hauser, THC

vh@thc.org

http://www.thc.org

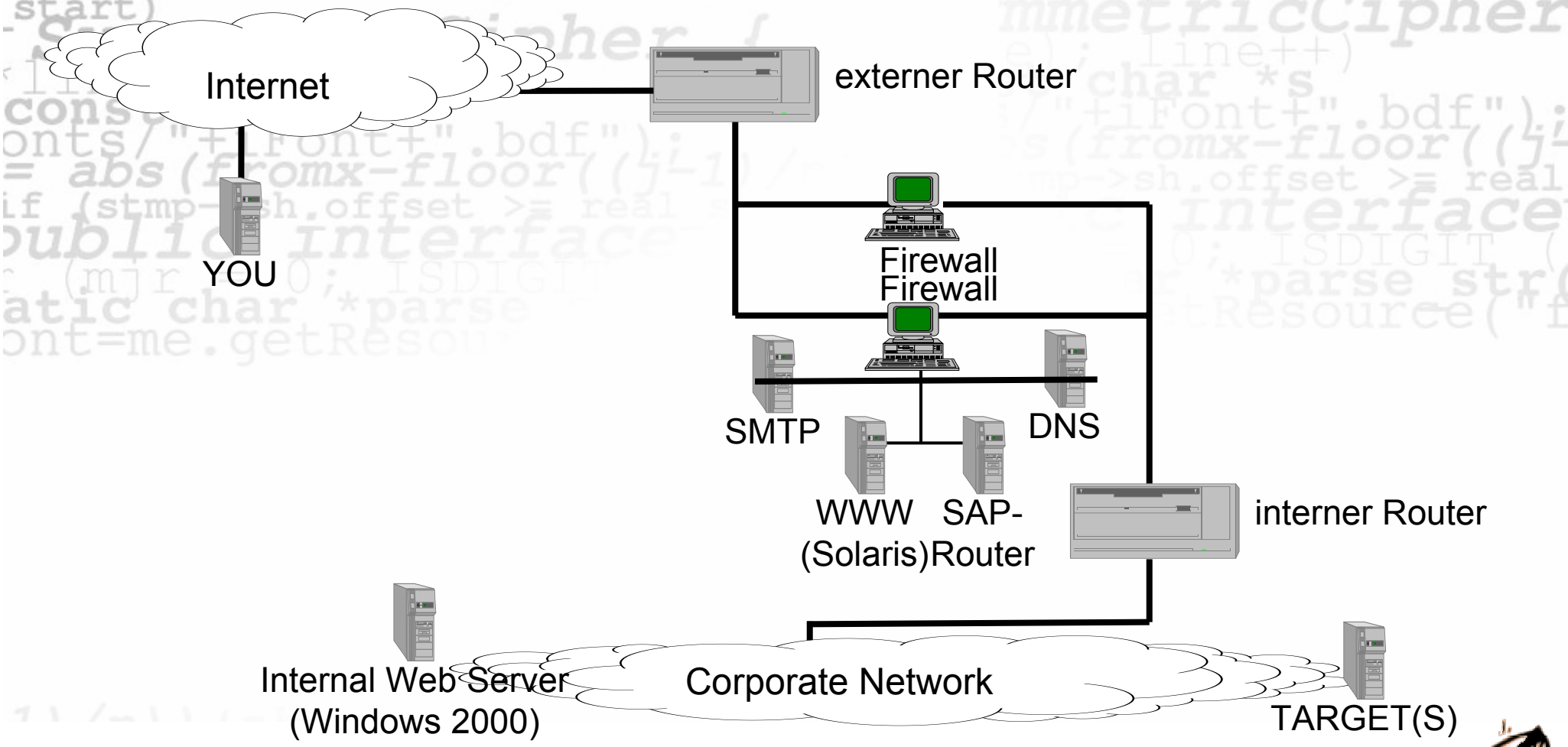
Contents

The Goal

- You 0wn a server in a network
- However, it is protected by a firewall and only allows a few ports in
- You would like to attack the network from the 0wned box, however
  - Your tools of the trade do not support that platform/OS
  - You do not have got an interactive login
  - etc.
- Wouldn't it be great to run tools like nmap, exploits, etc. from any system you 0wn, without the need to have your tools on that boxes?

What tools do exist to help you?

A common corporate network set-up



Solutions

- For executing command on a web server:
  - CMD.asp (and others)
  - shell.pl (and others)
- Reverse Shells
  - rwwwshell.pl
  - netcat
  - 'sploits
- Tunnel single connections
  - httptunnel
- Tunnel multiple connections
  - IPSEC
  - Socks
  - Grenzgaenger

How does Grenzgaenger work

- It uses the preload library feature (LD_PRELOAD) to hook into the interesting library calls:
  - connect()
  - gethostbyname()
  - gethostbyaddr()
  - ... and others ...
- Any usuable connect() etc. call is proxied through as many proxies as you like and performed on the last proxy instance
- This is invisible to the tool, Grenzgaenger is faking this for you ☺
- In essence, Grenzgaenger is a Hacker Socks

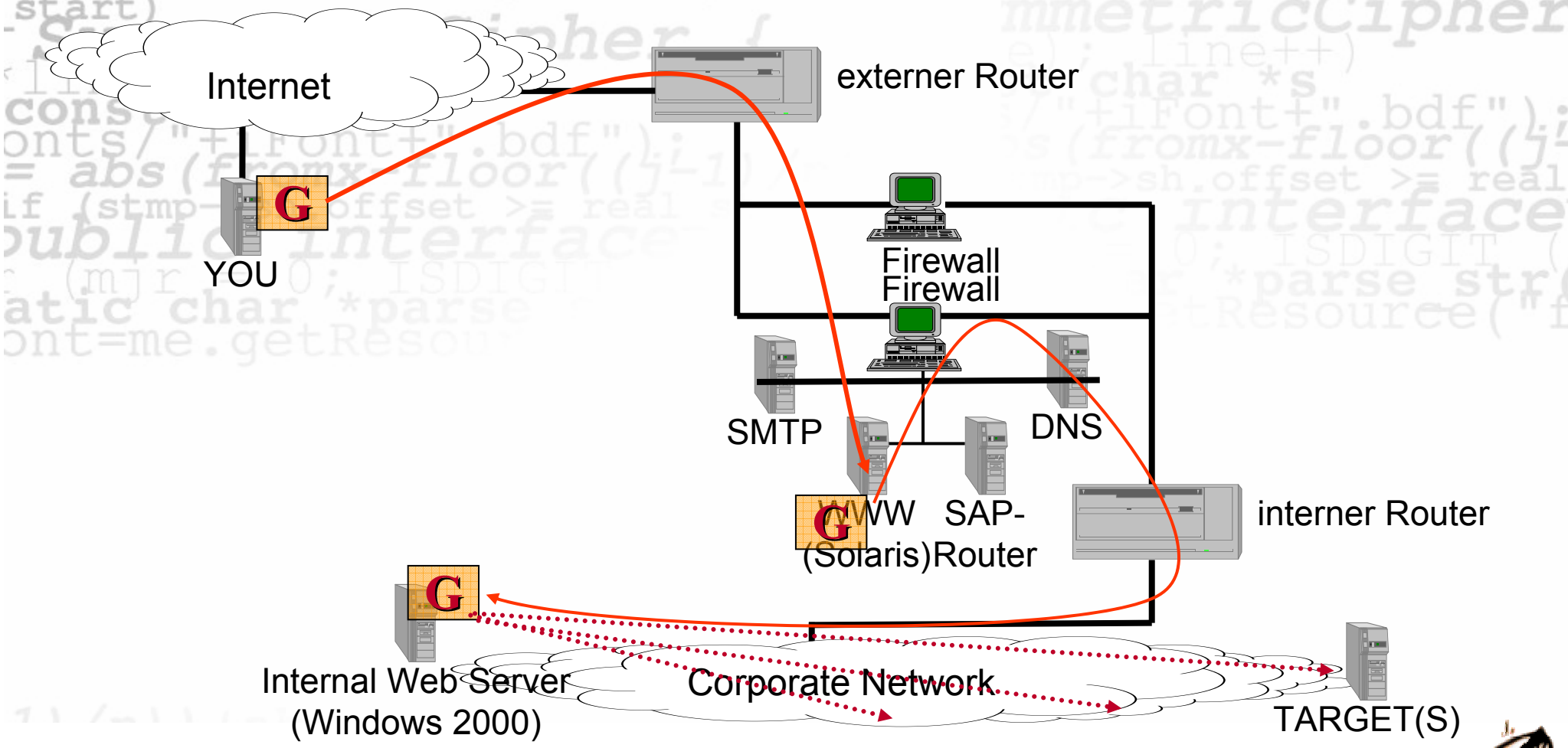
Limitations of Grenzgaenger

- Using LD_PRELOAD and proxying the data does NOT allow us to:
  - Use RAW sockets
  - Use tools which need to sniff replies off the wire (libpcap)
- The current release is in ALPHA state

How to use Grenzgaenger

- Run one Grenzgaenger on your local system
- Run Grenzgaenger instances on any systems you like
- Edit „gg“ to point to the other Grenzgaenger instances:
  - GG_TUNNEL=„10.0.0.1:443:secret>10.0.2.2:443:secret>**10.3.3.3**:443:no“
- Run any command with „gg“ in front of it to send the data to the target network 10.3.3.0:
  - gg telnet 10.3.3.4 23
  - gg nmap –sT –P0 10.3.3.0/24
  - gg nmap –sL 10.0.0.0/8
  - gg rpcinfo –p 10.3.3.10
  - etc.

A common corporate network set-up



It looks like this ...

```
laptop:/prg/grenzgaenger-alpha # gg nmap -sT -n -P0 -p 100-113 81.161.148.208

Starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-08-08 17:07 CEST
gg-intercept: connection to proxy established
Interesting ports on 81.161.148.208:
(The 13 ports scanned but not shown below are in state: closed)
Port       State       Service
111/tcp    open        sunrpc

Nmap run completed -- 1 IP address (1 host up) scanned in 1.414 seconds
laptop:/prg/grenzgaenger-alpha #
```

It looks like this ...

```
laptop:/prg/grenzgaenger-alpha # ssh vh@81.161.148.210
Password:
vh$ cd /tmp
vh$ ./ggd
Info: Admin connect from 81.161.148.222
Info: Admin connection successfully initiated
Info: Connect id 53050 to 81.161.148.208:109/tcp - failed
Info: Connect id 53051 to 81.161.148.208:104/tcp - failed
Info: Connect id 53052 to 81.161.148.208:112/tcp - failed
Info: Connect id 53053 to 81.161.148.208:111/tcp - success
Warning: Request to close connection NOT fulfilled, id 53054 was not found
Info: Executed close command on connect port
Warning: Request to close connection NOT fulfilled, id 0 was not found
Info: Connect id 53055 to 81.161.148.208:113/tcp - failed
Info: Connect id 53056 to 81.161.148.208:102/tcp - failed
Info: Connect id 53057 to 81.161.148.208:106/tcp - failed
Info: Connect id 53058 to 81.161.148.208:105/tcp - failed
Info: Connect id 53059 to 81.161.148.208:108/tcp - failed
Info: Connect id 53060 to 81.161.148.208:107/tcp - failed
Info: Connect id 53061 to 81.161.148.208:103/tcp - failed
Info: Connect id 53062 to 81.161.148.208:100/tcp - failed
Info: Connect id 53063 to 81.161.148.208:101/tcp - failed
Info: Connect id 53064 to 81.161.148.208:110/tcp - failed
vh$ exit
laptop:/prg/grenzgaenger-alpha #
```

End – the future

- Secure ID generation and usage
- Data encryption
- Support reverse/cmd line tunnel connections
- Proxy support
- "Local stuff" (exec commands etc.)
- Master mode

Expect the next version within 2-3 weeks …

Where to get it?

First public alpha available at:

http://www.thc.org

Questions?