

Battlefield IDS

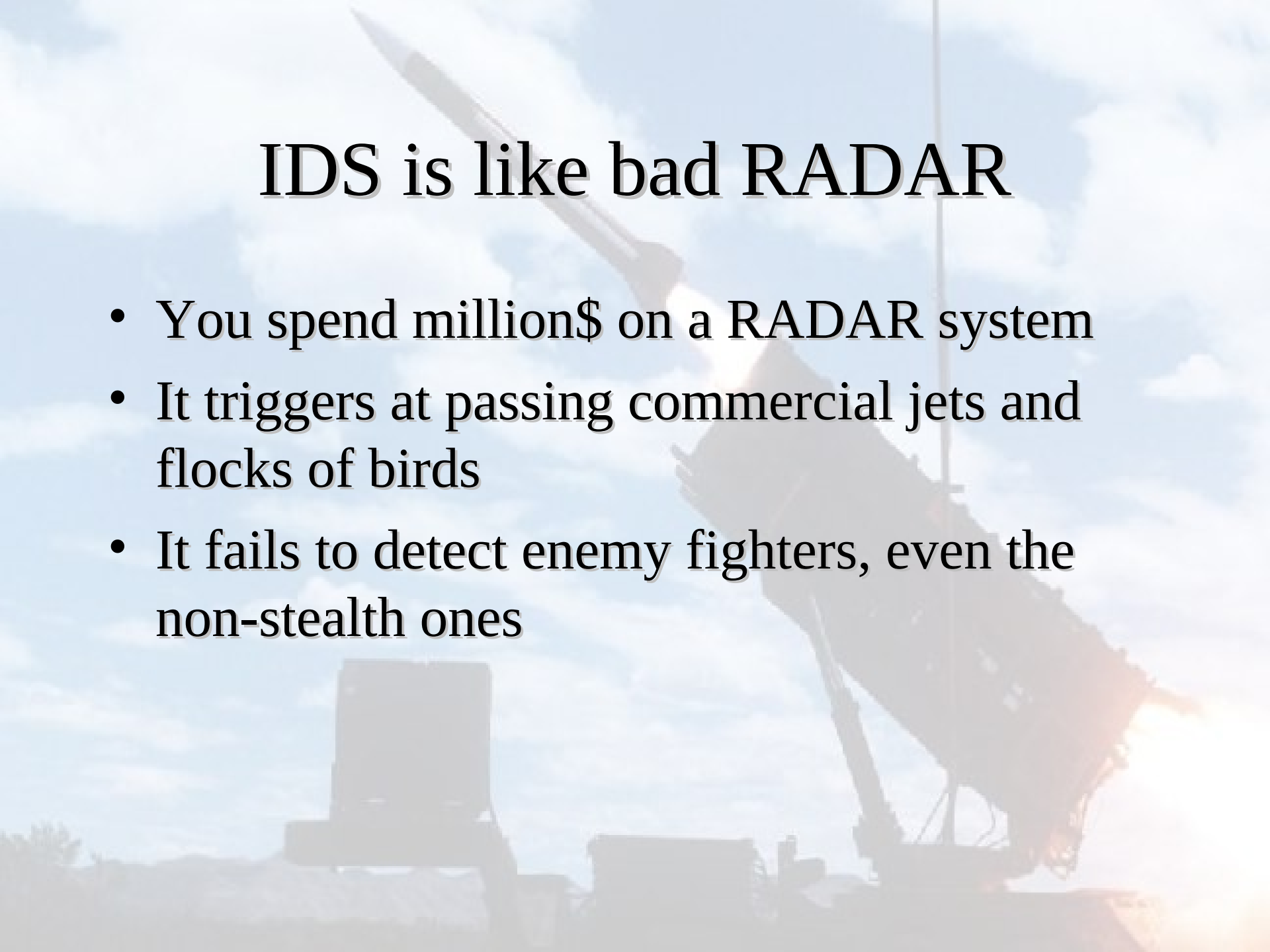
Life in the trenches

IDS is like an old tank

- You spend million\$ on an armored vehicle
- You take it out to the battlefield
- An enemy soldier hits the gas tank using a hand-gun, and it blows up.
- Tanks can be blown up (nukes, howitzers), but should stand up to hand-guns.

IDS is like bad RADAR

- You spend million\$ on a RADAR system
- It triggers at passing commercial jets and flocks of birds
- It fails to detect enemy fighters, even the non-stealth ones



IDS is like Iraqi soldiers

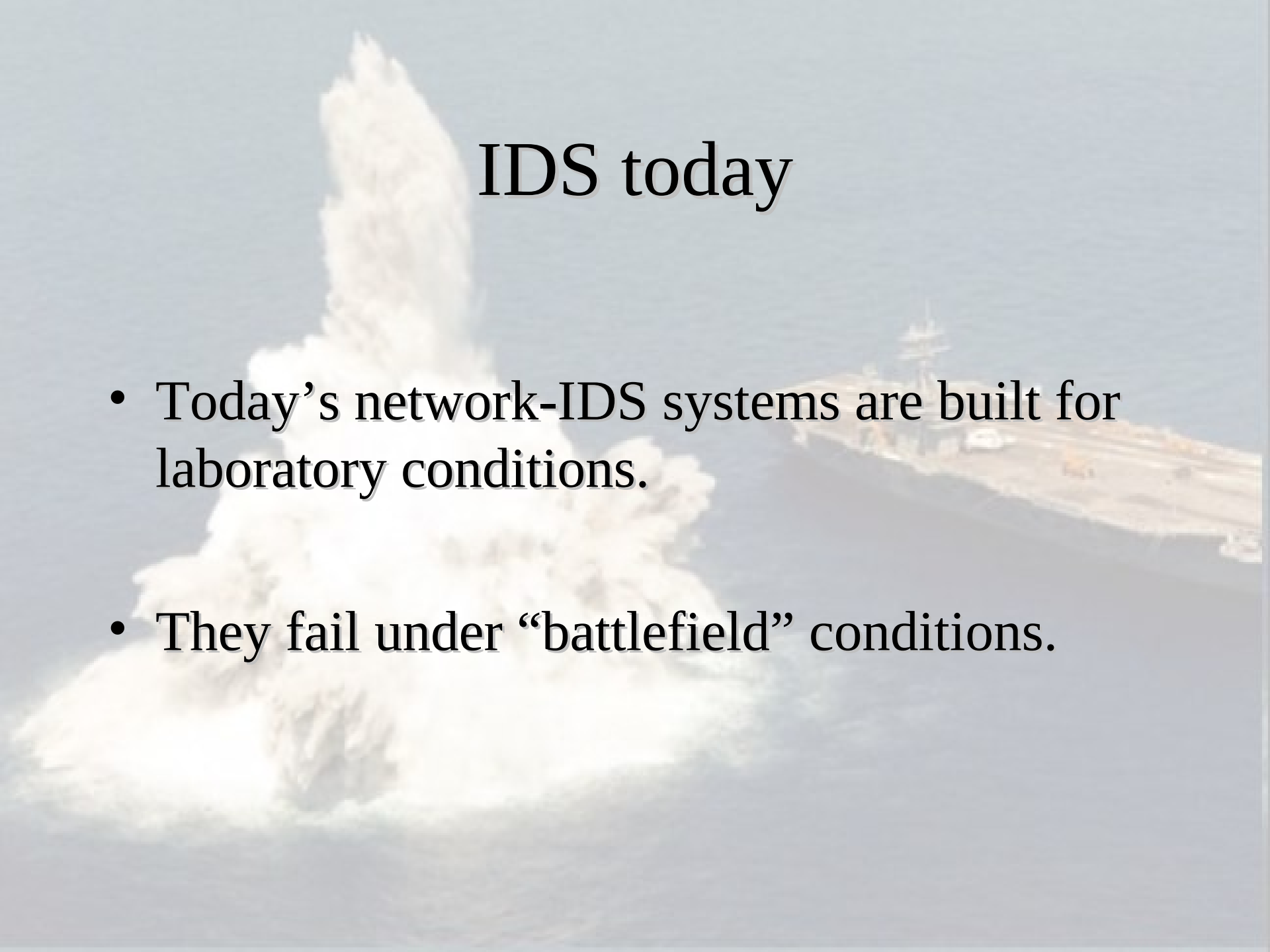
- Iraq had the second largest army¹ in the world (>million soldiers)
- However, their primary tactic during Desert Storm was to throw down their weapons and run away².

¹ March 15, 1991 Wall Street Journal.

² June, 1991 the U.S. estimated that more than 150,000 Iraqi soldiers deserted. Soldiers were very confused and just abandoned Kuwait and fled toward Baghdad.

IDS today

- Today's network-IDS systems are built for laboratory conditions.
- They fail under “battlefield” conditions.

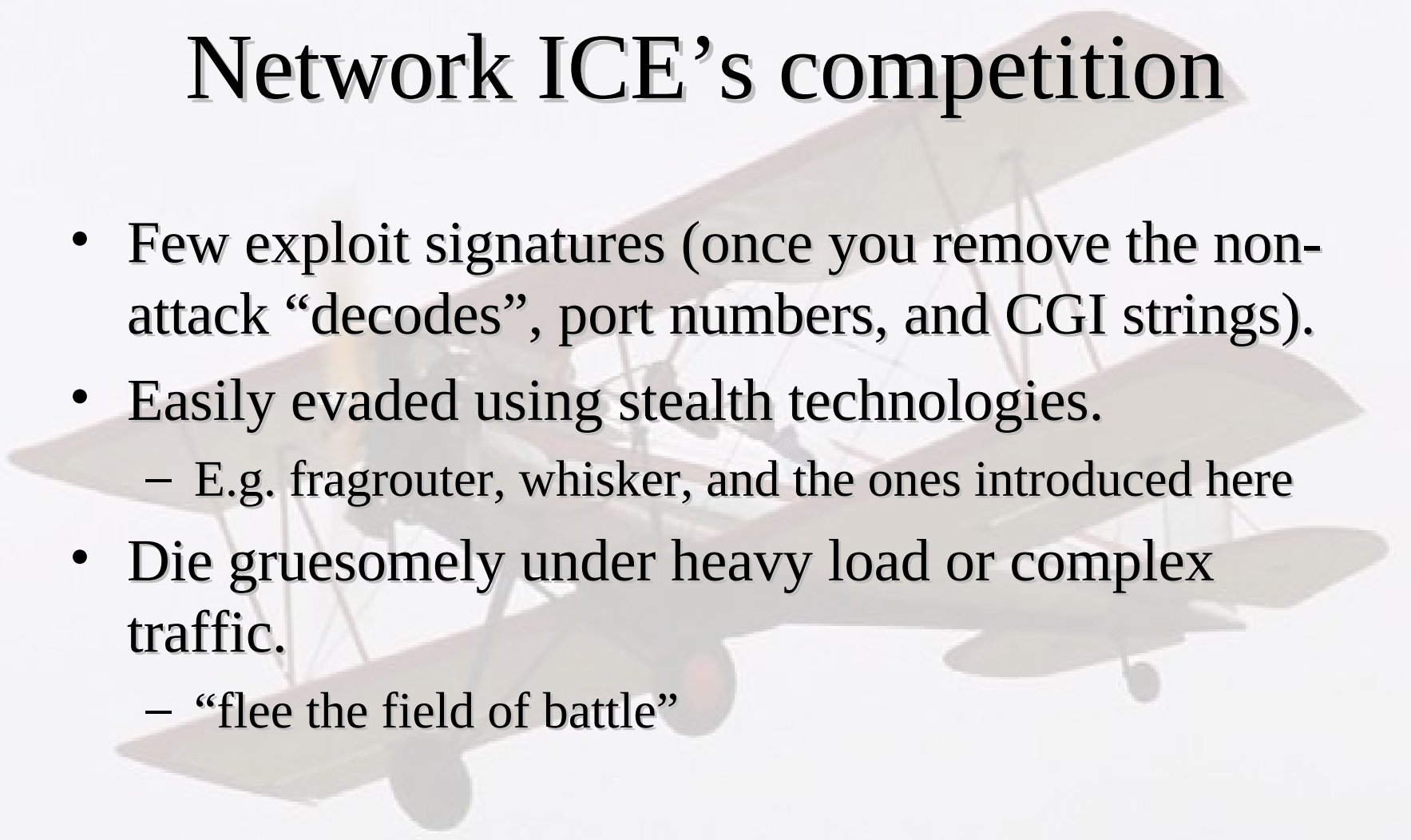


Network ICE

- Battle-tested IDS
- Runs on est. 2-million desktops (BlackICE Defender) in home/corp.
- Monitors high-end websites and ISP backbones (BlackICE Sentry)
 - in high-bandwidth sites that market leaders can't handle.
- I believe more traffic is monitored by BlackICE right this moment than by any other IDS.
 - I believe snort is second.

Blatant marketing plug

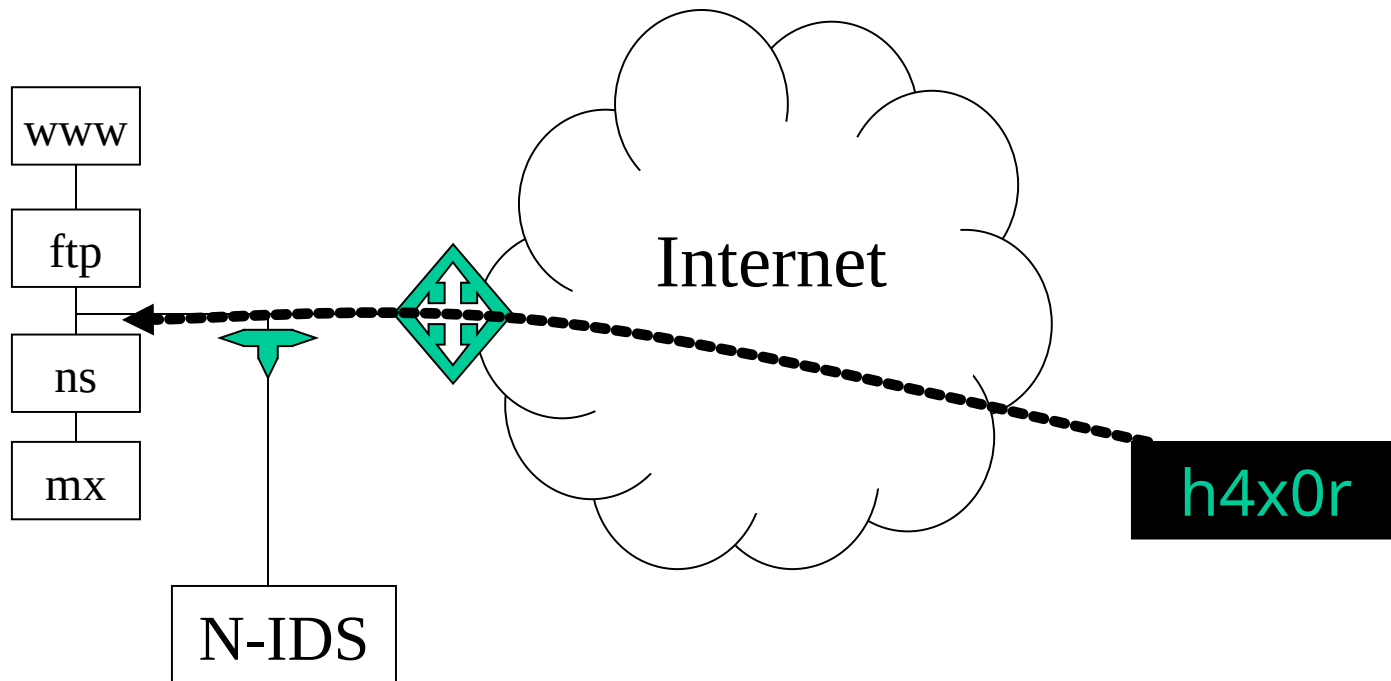
Network ICE's competition



- Few exploit signatures (once you remove the non-attack “decodes”, port numbers, and CGI strings).
- Easily evaded using stealth technologies.
 - E.g. fragrouter, whisker, and the ones introduced here
- Die gruesomely under heavy load or complex traffic.
 - “flee the field of battle”

This page
intentionally left
blank

What is network IDS



What ISN'T network IDS?

- N-IDS cannot detect intrusions, only “signs” of intrusions
 - A detective doesn’t witness the crime; s/he just sees the clues of the crime.
 - Hackers can smudge the clues
 - “erase their fingerprints”
 - N-IDS == sleuth, N-IDS != psychic
- N-IDS doesn’t “protect”, just “detects”
 - Frequent misconception about this presentation is that I’m helping hackers break into sites; but N-IDS don’t protect in the first place.

Classic Evasion: fragrouter

- Simple utility downloadable from the net
- Causes signatures to span packets rather than being fully contained in a single packet
- Defeats packet greppers
- Any/all attacks can be fragrouted

GET /cgi-bin/phf HTTP/1.0

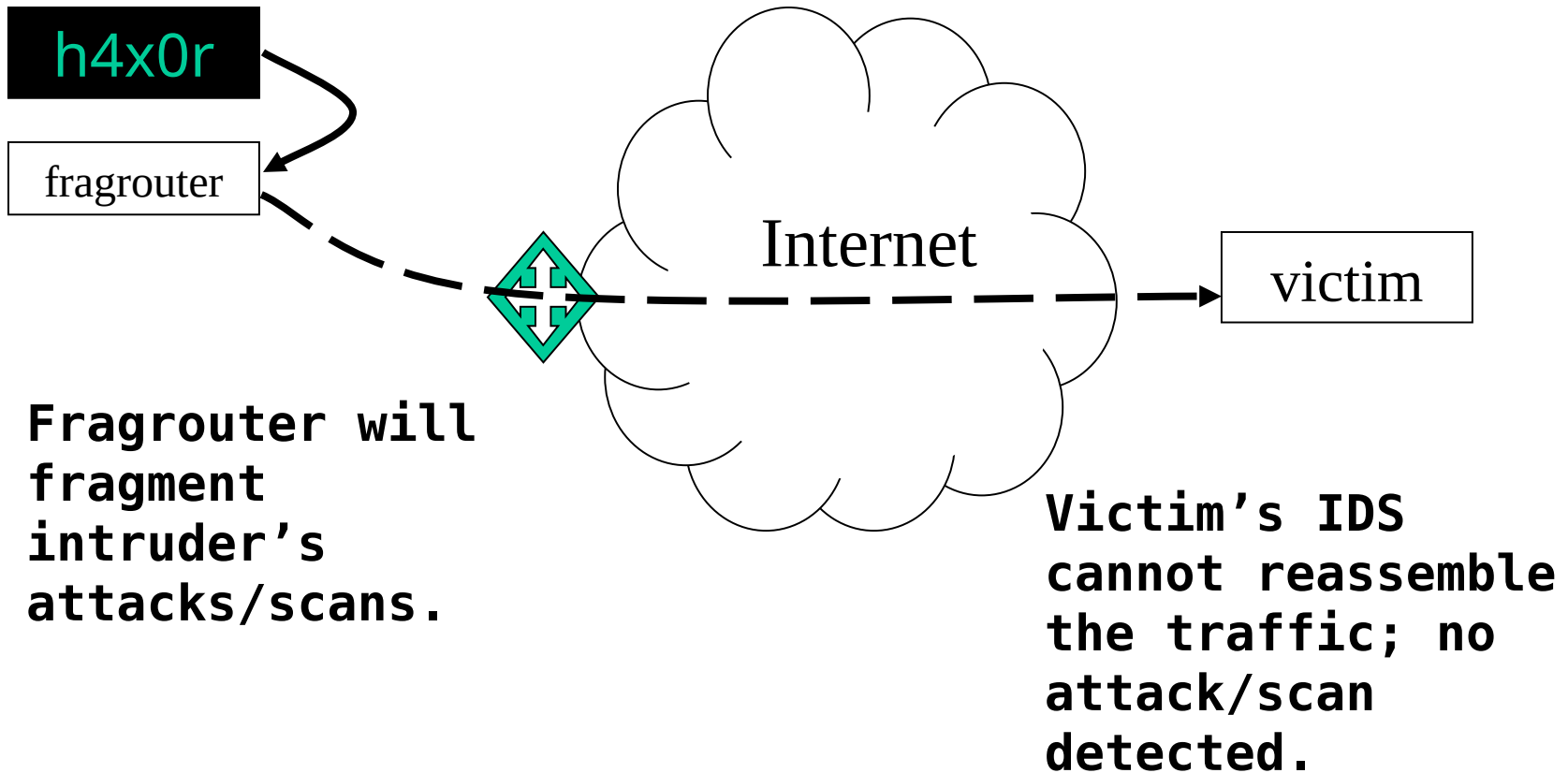
GET /cgi-bi

n/ph

f HTTP/1.0

<http://www.anzen.com/research/nidsbench/fragrouter.html>

fragrouter (cont)



Classic Evasion: nmap

- “stealth” SYN/half-scans
 - Not really stealth as far as N-IDS is concerned.
- SYN/FIN scanning using IP fragments
 - Bypasses some firewalls
 - ...and many IDS systems
 - ...even one vendor’s IDS that claims to reassemble packets

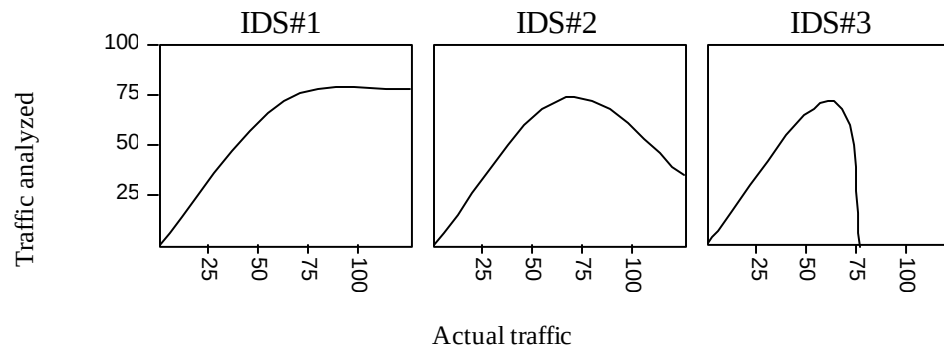
<http://www.insecure.org/nmap>

Classic Evasion: flood

- Packets/second is bigger problem than bits/second
- Vendor comment:
“...but 30,000pps is more than enough for normal networks...”
- But this isn't enough for networks under DDoS siege of 148,800pps
- IDS vendors design for the normal case; hackers create exceptional cases
 - Today's IDSs are built with the same assumptions as normal applications.

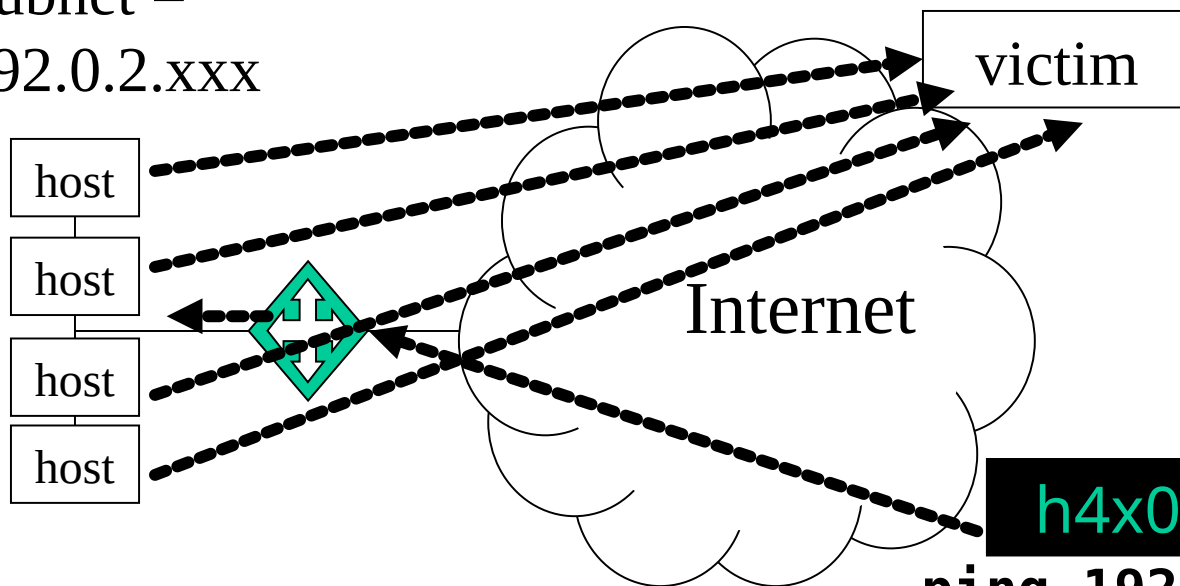
Failure modes

- They usually fail hard
 - Once they reach their limits, they usually degrade to processing no packets.



Smurf/fraggle

Subnet =
192.0.2.xxx

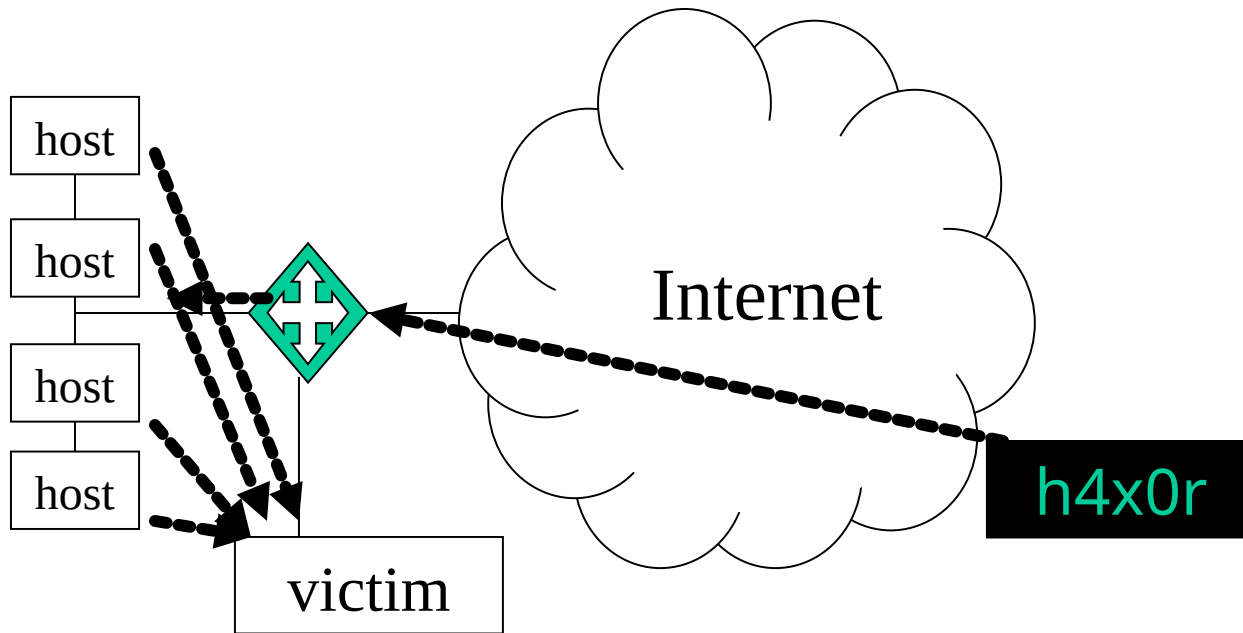


Router uses MAC
address
FF:FF:FF:FF:FF:FF for
IP address 192.0.2.255

ping 192.0.2.255
Or UDP port 7
Or other UDP ports
Spoof victim

<http://www.quadrunner.com/~chuegen/smurf.txt>

Smurf/fraggle (cont)



Classic Evasion: jamming

- From a dialup, run utilities like Targa and nmap decoy scans against the victim all day long.
- Use extensive spoofing.
- Database fills up with these events and keels over.
 - Most vendors use Access .mdb files.
 - Right in the documentation they recommend thumping the database.
 - “DB High-water Mark Reached”
 - “Flee the field of battle”

Classic Evasion: RFP's whisker

- Changes “signature” but not “meaning”.
- Multiple techniques can be combined.

```
GET /cgi-bin/phf
```

```
HEAD /cgi-bin/phf
```

```
GET /%63%67%69%2d%62%69%6e/phf
```

```
GET //cgi-bin/phf
```

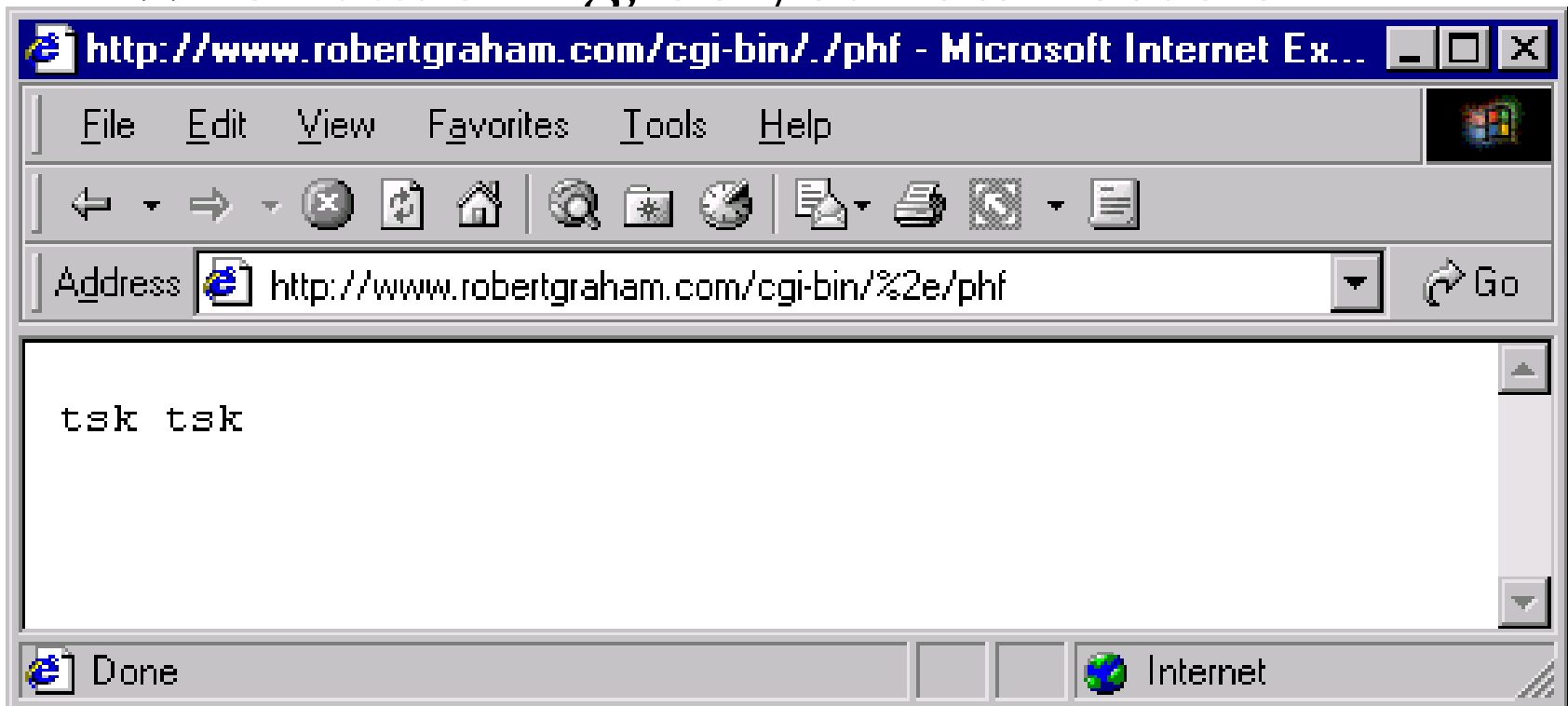
```
GET /cgi-bin/foobar/./phf
```

```
GET /cgi-bin/./phf
```

```
GET%00/cgi-bin/phf
```

whisker (cont)

- Whisker only scans, not attacks.
- When attacking, do your own evasion



Classic Evasion: buffer overflow

- NOP codes
 - Serve as “filler” within buffer overflow exploits
 - Example: 0x90 for x86
- Most CPUs have several NOP codes
- Can also fill with normal ops
 - Increment/decrement registers you’re going to overwrite anyway

Other x86 NOPs

Some 2 byte x86 nops:

87DB	XCHG	EBX, EBX
89DB	MOV	EBX, EBX
89C9	MOV	ECX, ECX
89D2	MOV	EDX, EDX
89C0	MOV	EAX, EAX
8D1F	LEA	EBX, [BX]
8D34	LEA	ESI, [ESI]
8D3D	LEA	EDI, [EDI]

Some 3 byte nops

8D6D00	LEA	EBP, [EBP+00]
8D2424	LEA	ESP, [ESP+00]

6 byte nop

8D 05 00000000	LEA	EAX, [EAX+00]
----------------	-----	---------------

There are longer ones. Some depend on the CPU mode that you are in.

buffer overflow NOP fill

- Randomize signature

```
/*old way*/  
for (i=0; i<BUFSIZE; i++)  
    buf[i] = 0x90;  
  
/*evasion fill*/  
for (i=0; i<BUFSIZE; i+=2) {  
    int x = rand()%NOP_COUNT;  
    buf[i] = nops[x]>>8;  
    buf[i+1] = nops[x];  
}
```

buffer overflow evasion

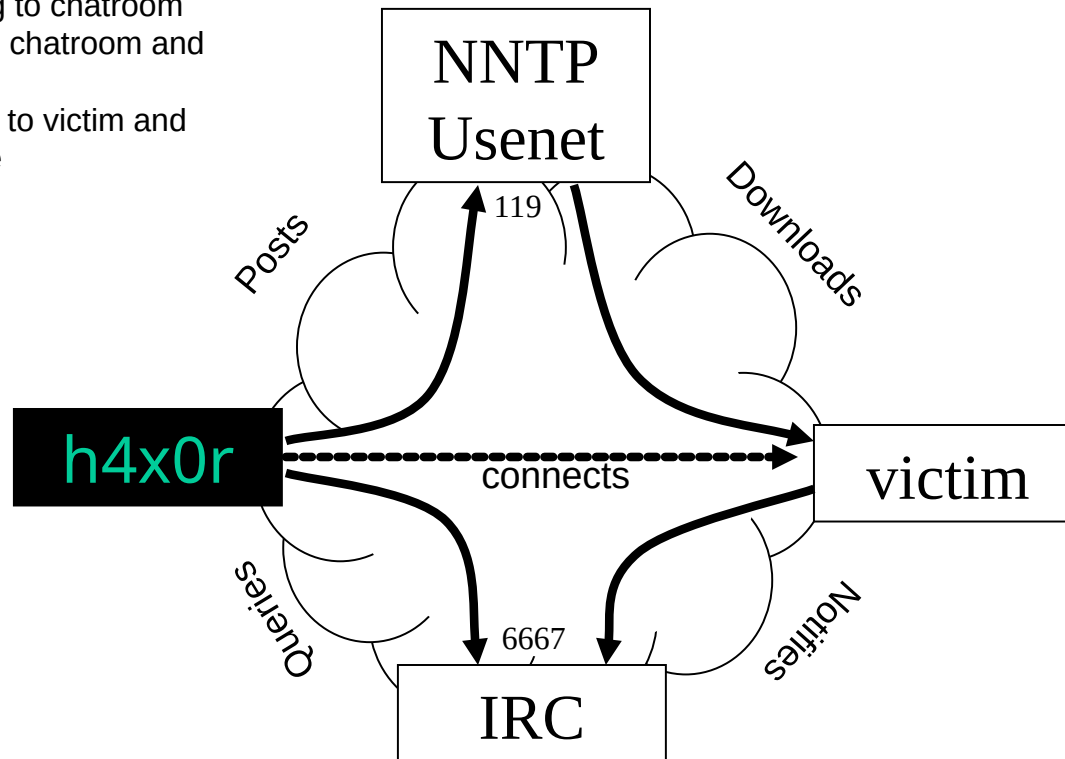
- Only packet-grep is affected by this
- Protocol decoding doesn't look at patterns
 - Field-size within protocol headers
 - Suspicious content like binary when text expected

Classic Evasion: change Trojan ports

- Back Orifice = 31337, sub7 = 1243, 27374
 - But they can run on any port
- State of the net: Sub7
 - Heavy scans for 27374 and 1243, because Sub7 includes built-in scanner...
 - ...but most successful compromises are for non-default ports and non-default passwords.

Trojan infection

1. Post trojan
2. Victim downloads/runs
3. Trojan posts msg to chatroom
4. Hacker logs onto chatroom and finds victim
5. Hacker connects to victim and controls machine



server:

startup method[s]

☐ registry -Run ? ☒ WIN.INI
☐ registry -RunServices ☐ less known method
key name: ? ☐ _not_ known method

notification options

victim name:
☐ enable ICQ notify to UIN:
☐ enable IRC notify. ? notify to:
irc server: port:
☐ enable e-mail notify. ? notify to:
 server: user:

installation

☐ automatically start server on port:
☐ use random port ?
☐ server password: reenter:
☐ protect server port and password
☐ enable IRC BOT
server name: ☒ use default name
☐ specify a filename:
☐ melt server after installation
☐ enable fake error message:
☐ bind server with EXE file: ?

protect server

☐ protect the server so it can't be edited/changed ? password: reenter:

☒ closeEditServer after saving or updating settings *note: if you have problems opening the server [click here](#)

Ueberevasion: Theory

- Theory
 - OSI 7-layer model
 - Presentation Layer is responsible for formatting data on the wire
 - Changing format changes raw pattern, evades pattern grep

7. Application
6. Presentation
4. TCP
3. IP
2. Ethernet
1. Cabling/bits

UeberEvasion: RPC record fragging

- RECORD MARKING
 - Needed to overcome TCP's byte-stream abstraction
 - Additional layer of fragmentation within RPC

7. automount, statd,
tooltalk, admind,
portmap, mountd,
lockd, nfs, pcnfs, nis,
cmsd, amd, sadmind,
null proc scan

6. RPC/XDR

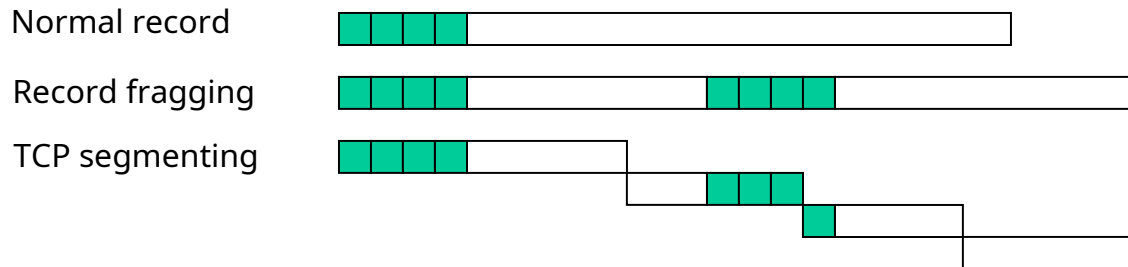
4. TCP

3. IP

2. Ethernet

RPC record fragging

- Boundaries are unrelated
 - Record marking and TCP segment boundaries do not need to match up/overlap
- Record marking $\neq$ XDR
 - No restriction on 4-byte alignment for size of record frags



mysend()

```
/** RPC anti-IDS send function.  
 * Replace call to send() with this function. */  
static void mysend(int fd, const char *msg, int sizeof_msg)  
{  
    int i;  
    int is_at_end = 0;  
    MyBuf lower = {0};  
    MyBuf upper = {0};  
  
    /* Send each byte through our randomizer one at a time.*/  
    for (i=0; i<sizeof_msg; i++) {  
        if (i == sizeof_msg - 1)  
            is_at_end = 1;  
        mysend_upper(fd, msg[i], is_at_end, &lower, &upper);  
    }  
}
```

mysend_upper() record frag

```
static int mysend_upper(int fd, char c, int is_at_end, MyBuf *myupper, MyBuf *mylower) {
    int r = rand() & 0xFF;
    myupper->buf[myupper->length] = c;
    myupper->length++;
    if (r + myupper->rvalue > randomness || is_at_end || myupper->length >= MYBUF_MAX) {
        int i;
        /* First, construct the record header */
        myupper->record_marker[0] = (unsigned char)(is_at_end ? 0x80 : 0x00);
        myupper->record_marker[1] = (unsigned char)(myupper->length >> 16);
        myupper->record_marker[2] = (unsigned char)(myupper->length >> 8);
        myupper->record_marker[3] = (unsigned char)(myupper->length >> 0);
        /* Now send the bytes one at a time to the lower layer,
         * which will do its own buffering and random transmission.*/
        for (i=0; i<4; i++)
            mysend_lower(fd, myupper->record_marker[i], 0, mylower);
        for (i=0; i<myupper->length; i++)
            mysend_lower(fd, myupper->buf[i], is_at_end && i==(myupper->length-1), mylower);
        myupper->length = 0;
        myupper->rvalue = 0;
    } else
        myupper->rvalue += randomness/10;
    return 0;
}
```

mysend_lower() TCP frag

```
static int mysend_lower(int fd, char c, int is_at_end, MyBuf *my)
{
    int r = rand() & 0xFF;
    my->buf[my->length] = c;
    my->length++;
    if (r + my->rvalue > randomness
        || is_at_end || my->length >= MYBUF_MAX) {
        usleep(rand()*100);
        send(fd, (char*)my->buf, my->length, 0);
        my->length = 0;
        my->rvalue = 0;
    } else
        my->rvalue += randomness/10;
    return 0;
}
```

MyBuf structure

```
int randomness = 200;  
#define MYBUF_MAX 100  
typedef struct MyBuf {  
    unsigned char record_marker[4];  
    unsigned char buf[MYBUF_MAX];  
    int length;  
    int rvalue;  
} MyBuf;
```

Normal PortDump

- Single packet
 - Easy to detect

```
80 00 00 28 2E 48 78 EC 00 00 00 00 00 00 00 02
00 01 86 A0 00 00 00 02 00 00 00 04 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
```

Decode:

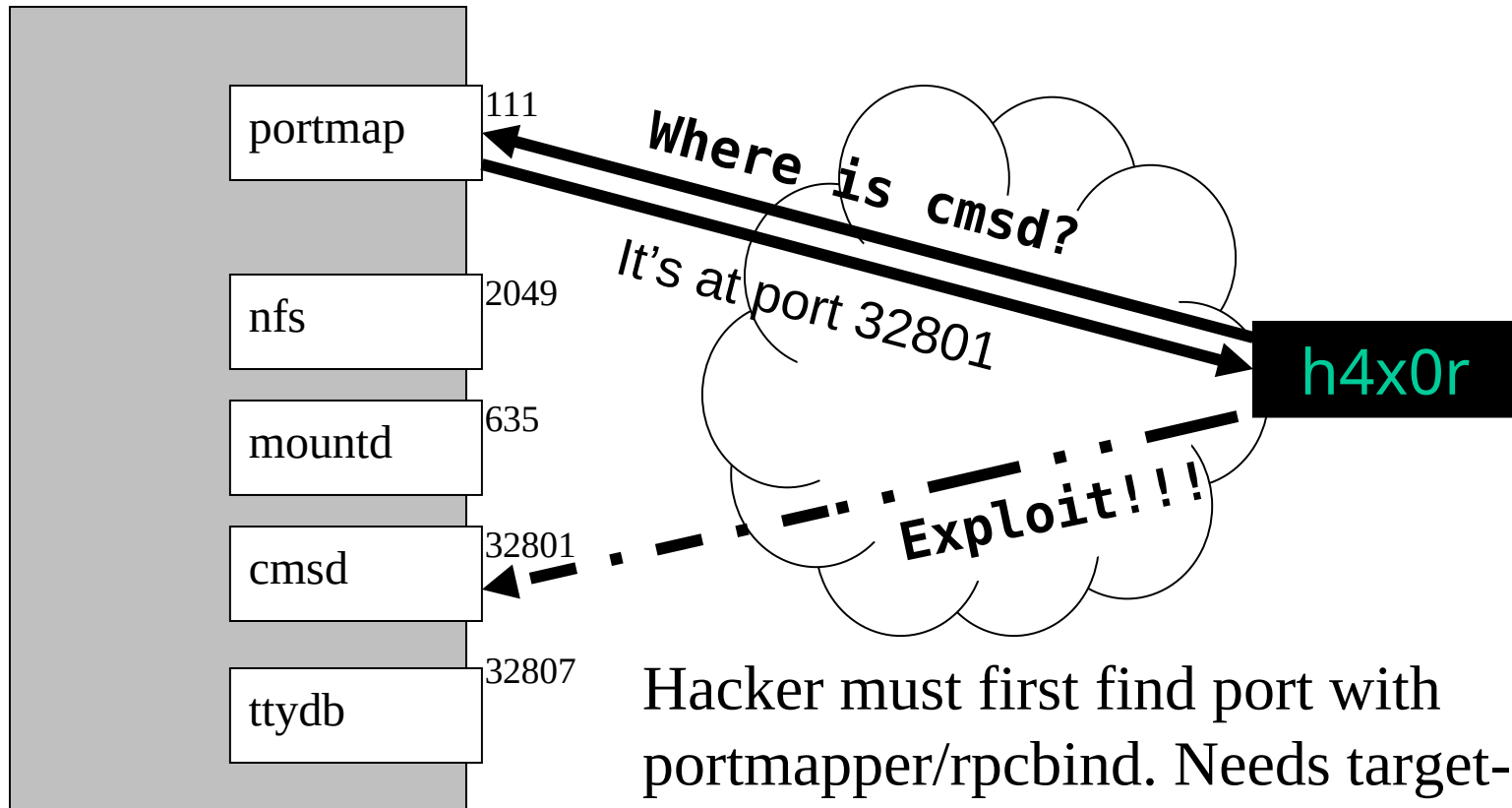
```
E len=0x28      xid=2E4878EC msgtype=call rpcver=2
Prog=portmap    progvers=2    proc=dump(4) auth=null(0)
Authlen=0       verf=null(0)  verflen=0
```

Evasion PortDump

00 00
00 05 2e 48 78
ec
00 00
00 00 01
00 00 00 00 0c 00 00 00 00 00 02 00 01
86 a0 00 00 00
00 02
00 02 00 00 00 06 00 00
00 04 00 00
00 00 00 04
00 00 00 00
00 00 00 04 00 00 00 00 00 00 00 01
00 00 00
00 02 00 00 80 00 00 03 00
00
00

- multiple packet
 - Hard to detect
 - Evades all IDSs

How RPC works



Hacker must first find port with portmapper/rpcbind. Needs target-based profiling to know which ports are RPC.

NGC smellsLike()

UeberEvasion: FTP option insertion

- FTP uses Telnet for control connection
 - It isn't really Telnet
 - ...but does support options.
- 0xFF is the “option” code, followed by one or two bytes.
 - USE\xFF\xF1R rob
 - PA\xFF\xF1SS foo

FTP Options

- Like RPC
 - Evasion at the OSI Presentation layer
- Command-channel exploits:
 - Bounce scans, tar –compress, buffer overflows, CWD ~root, USER/PASS, SITE EXEC, printf() format bugs, shell characters

7. FTP, Telnet
6. Telnet
4. TCP
3. IP
2. Ethernet

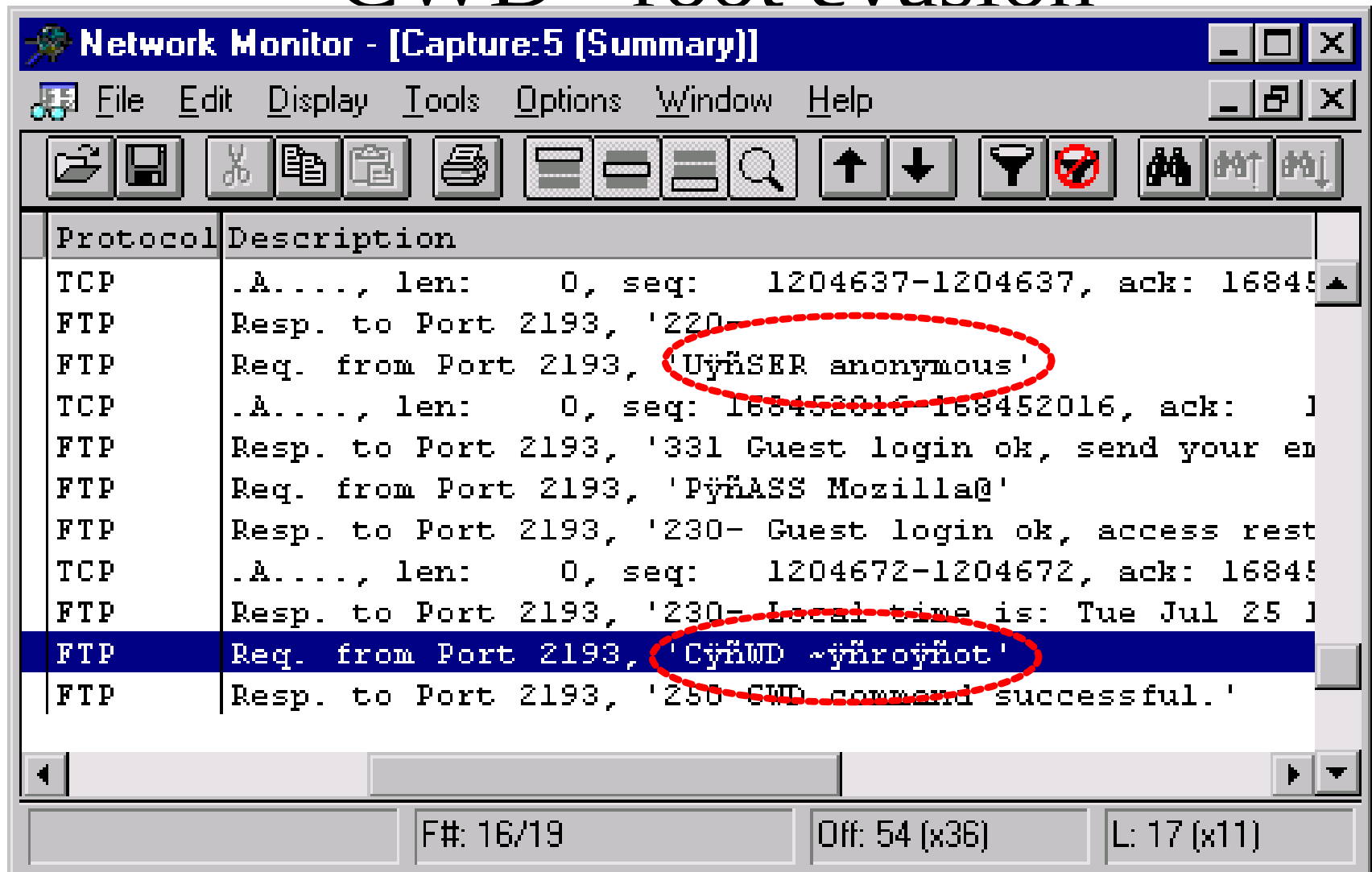
FTP (cont.)

- N-IDS needs target profiling
 - UNIX FTP support option negotiation
 - 3-character DO/DONT/WILL/WONT negotiation
 - Elicits response from server; useful for desynchronization.
 - WinNT FTP
 - No option negotiation
 - Same imbedded NUL problem as whisker
 - Other FTP servers on Windows
 - Don't support options

FTP (cont)

- Many simplistic servers don't support options
 - Eg. ServU
 - Therefore, exploit scripts need to profile target
 - U\xFF\xF1SER anonymous

CWD ~root evasion

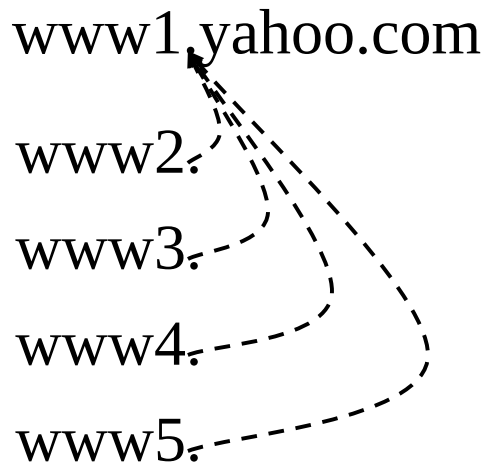


Protocol	Description
TCP	.A...., len: 0, seq: 1204637-1204637, ack: 16849
FTP	Resp. to Port 2193, '220'
FTP	Req. from Port 2193, 'UyñSER anonymous'
TCP	.A...., len: 0, seq: 168452016-168452016, ack: 1
FTP	Resp. to Port 2193, '331 Guest login ok, send your en
FTP	Req. from Port 2193, 'PýñASS Mozilla@'
FTP	Resp. to Port 2193, '230- Guest login ok, access rest
TCP	.A...., len: 0, seq: 1204672-1204672, ack: 16849
FTP	Resp. to Port 2193, '230- Local time is: Tue Jul 25 1
FTP	Req. from Port 2193, 'CýñWD ~ýñroýñot'
FTP	Resp. to Port 2193, '250 CWD command successful.'

Evasion: DNS

- Case sensitivity
- Compression
 - Splits names up, hides names the IDS might look for
 - Can crashes IDSs and sniffers using recursion

DNS compression



A DNS packet can contain many names. Essentially LZ-like compression with pointers back to repeated strings.

TXT CHAOS VERSION.BIND

Network Monitor - [T:\tracefiles\exploits\dns-evade-version-bind.cap (Summary)]

File Edit Display Tools Options Window Help

Frame Time Src MAC Addr Dst MAC Addr Protocol Description

Frame	Time	Src MAC Addr	Dst MAC Addr	Protocol	Description
7	2.519	GEMINI	ACCTON9D8E32	DNS	0xA:Std Qry for Version.Bind
8	2.520	00E018E00F96	GEMINI	DNS	0xA:Std Qry Resp. for VERSIO

00000000 00 00 E8 9D 8E 32 00 40 05 A4 79 32 08 00 45 00 ..F*Ä2.0.ÿ2..E.

00000010 00 3C 50 DD 00 00 80 11 D5 52 0A 0A 00 09 0A 0A .<P!...Ç.+R.....

00000020 00 65 0B E1 00 35 00 28 FC 7C 00 0A 01 00 00 01 .e.B.5.(n|.....

00000030 00 00 00 00 00 00 07 56 65 72 73 69 6F 6E C0 1AVersion+.

00000040 00 10 00 03 04 42 69 6E 64 00Bind.

Internet Domain Name System Packet F#: 7/19 Off: 42 (x2A) L: 26 (x1A)

Summary: protocol evasion

- Conceptually part of OSI Presentation Layer
 - For HTTP, RPC, FTP, and DNS
- You might see them appear in nmap soon
 - FTP bounce scan evasion
 - RPC null-proc evasion

This page
intentionally left
blank

Most popular ways into a system

- DNS BIND
- FTP
- RPC
- HTTP
- Trojans

Easiest ways of evading IDSs

- DNS BIND
- FTP
- RPC
- HTTP
- Trojans
- Frag
- Flood
- Jam

Conclusion:

- Protocol evasion is still new
 - I've only touched the surface; the field is wide open for further research
 - In creating this document, I discovered yet another FTP issue that can evade even our product
 - Lots of other protocols are vulnerable to presentation-layer and application layer evasion.

A faded background image showing several soldiers in a desert environment. Some soldiers are in the foreground, wearing helmets and carrying equipment, while others are further back. The scene appears to be a training exercise or a combat situation.

For more info:

<http://www.robertgraham.com/slides/defcon8>
<mailto:defcon8@robertgraham.com>

N-IDS FAQ:

<http://www.robertgraham.com/pubs>

BlackICE Defender for Linux (in a month):

<http://netice.com/advice/support/os/linux>