Running code in browser poses security risks

- Compromise host
  - Write to file system
  - Interfere with other processes in browser environment
- Steal information
  - Read file system
  - Read information associated with other browser processes (e.g., other windows)
  - Fool the user
  - Reveal information through traffic analysis

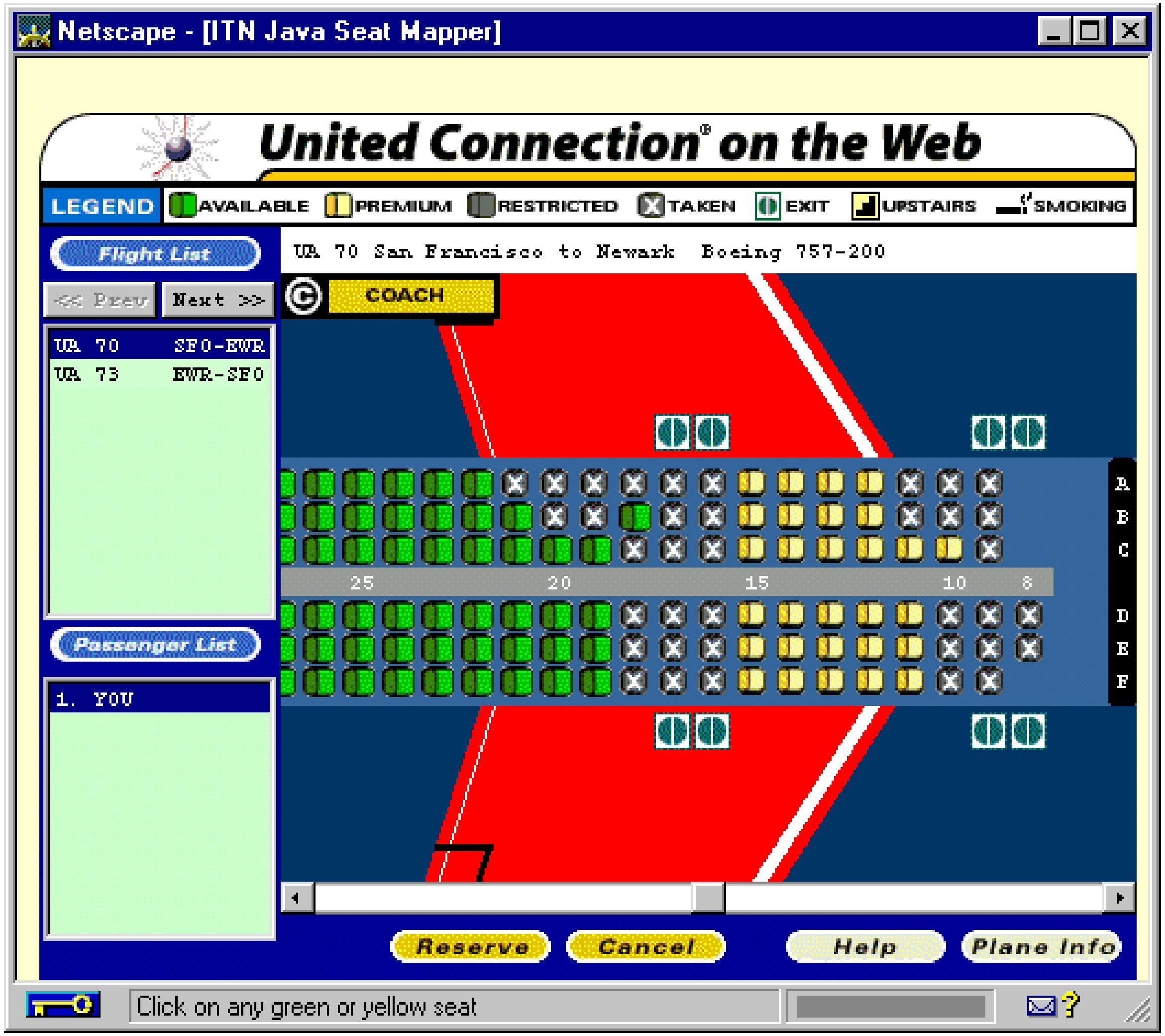
Browser sandbox

- Idea
  - Code executed in browser has only restricted access to OS, network, and browser data structures
- Isolation
  - Similar to address spaces or SFI, conceptually
  - Browser is a “weak” OS
  - Same-origin principle
    - Browser “process” consists of related pages and the site they come from

Java

- General programming language
- Web pages may contain Java code
  - Java executed by Java Virtual Machine
  - Special security measures associated with Java code from remote URLs
- Javascript, other security models are based on Java security model

Java Applet



Netscape - [ITN Java Seat Mapper]

United Connection® on the Web

UA 70 San Francisco to Newark Boeing 757-200

UA 70 SFO-EWR
UA 73 EWR-SFO

1. YOU

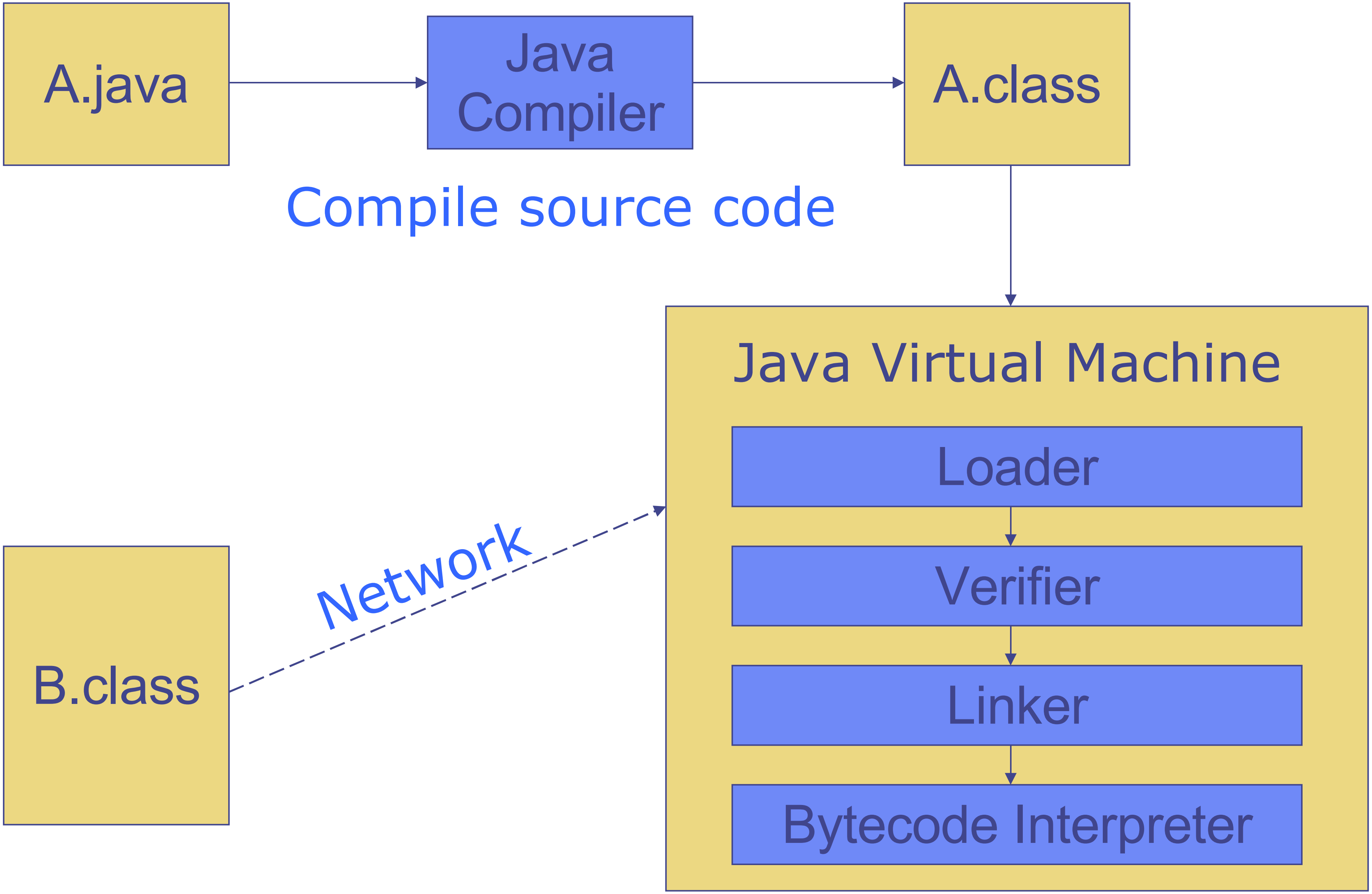
Click on any green or yellow seat

- Local window
- Download
  - Seat map
  - Airline data
- Local data
  - User profile
  - Credit card
- Transmission
  - Select seat
  - Encrypted msg

Mobile code security mechanisms

- Examine code before executing
  - Java bytecode verifier performs critical tests
- Interpret code and trap risky operations
  - Java bytecode interpreter does run-time tests
  - Security manager applies local access policy
- Security manager policy based on
  - Site that suppplied the code
  - Code signing – who signed it?

Java Virtual Machine Architecture



Class loader

- Runtime system loads classes as needed
  - When class is referenced, loader searches for file of compiled bytecode instructions
- Default loading mechanism can be replaced
  - Define alternate ClassLoader object
    - Extend the abstract ClassLoader class and implementation
  - Can obtain bytecode from network
    - VM restricts applet communication to site that supplied applet

Verifier

- Bytecode may not come from standard compiler
  - Evil hacker may write dangerous bytecode
- Verifier checks correctness of bytecode
  - Every instruction must have a valid operation code
  - Every branch instruction must branch to the start of some other instruction, not middle of instruction
  - Every method must have a structurally correct signature
  - Every instruction obeys the Java type discipline

Last condition is fairly complicated

Type Safety of JVM

- Load-time type checking
- Run-time type checking
  - All casts are checked to make sure type safe
  - All array references are checked to be within bounds
  - References are tested to be not null before dereference
- Additional features
  - Automatic garbage collection
  - NO pointer arithmetic

If program accesses memory, the memory is allocated to the program and declared with correct type

How do we know verifier is correct?

- Many early attacks based on verifier errors
- Formal studies prove correctness
  - Abadi and Stata
  - Freund and Mitchell
    - Found error in initialize-before-use analysis

JVM uses stack machine

- Java

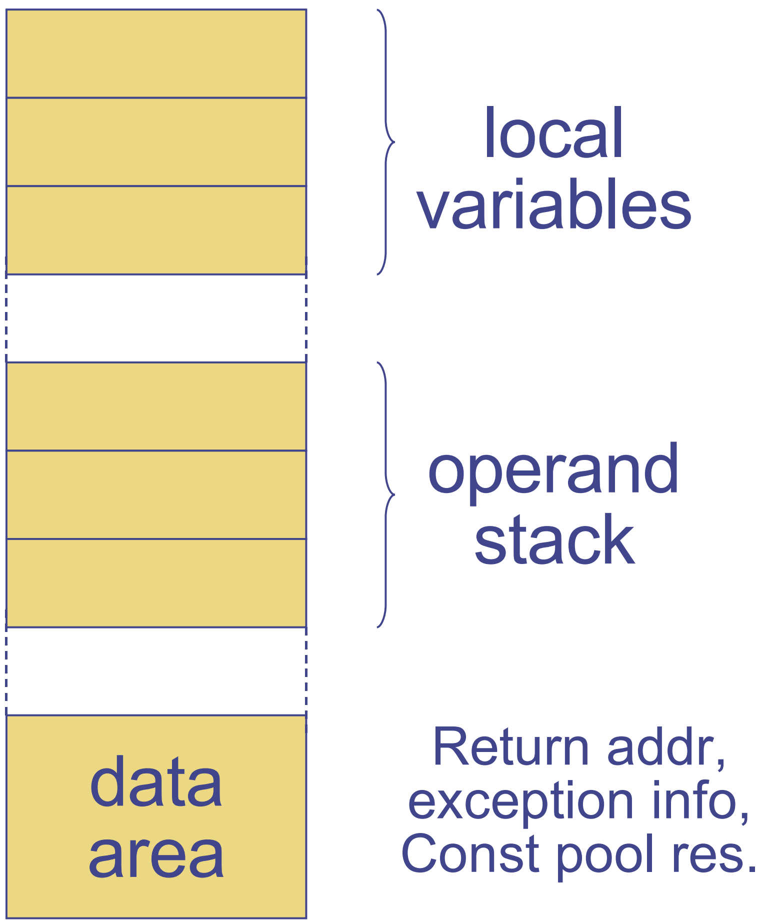
```
Class A extends Object {
        int i
        void f(int val) { i = val + 1;}
}
```

- Bytecode

```
Method void f(int)
        aload 0     ; object ref this
        iload 1     ; int val
        iconst 1
        iadd        ; add val +1
        putfield #4 <Field int i>
        return
```

refers to const pool

JVM Activation Record



local variables

operand stack

data area

Return addr, exception info, Const pool res.

Java Object Initialization

```
Point p = new Point(3);
p.print();

1:  new Point
2:  dup
3:  iconst 3
4:  invokespecial <method Point(int)>
5:  invokevirtual <method print()>
```

- No easy pattern to match.
- Multiple refs to same uninitialized object.

Bug in Sun’s JDK 1.1.4

- Example:

variables 1 and 2 contain references to two different objects,
verifier thinks they are aliases

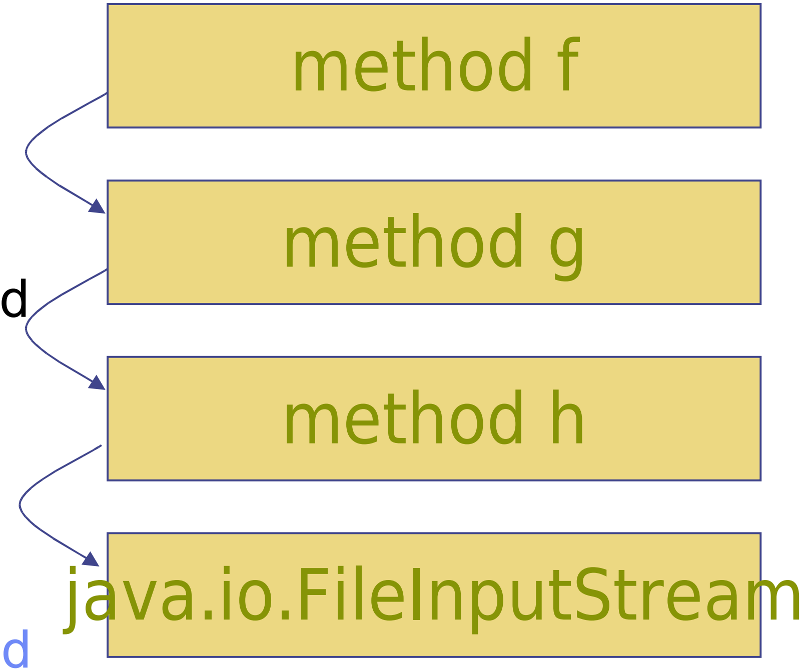
```
1:   jsr 10
2:   store 1
3:   jsr 10
4:   store 2
5:   load 2
6:   init P
7:   load 1
8:   use P
9:   halt
10:  store 0
11:  new P
12:  ret 0
```

Security Manager

- Java library functions call security manager
- Security manager object answers at run time
  - Decide if calling code is allowed to do operation
  - Examine protection domain of calling class
    - Signer: organization that signed code before loading
    - Location: URL where the Java classes came from
  - Uses the system policy to decide access permission

Stack Inspection

- Permission depends on
  - Permission of calling method
  - Permission of all methods above it on stack
    - Up to method that is trusted and asserts this trust



method f

method g

method h

java.io.FileInputStream

Many details omitted

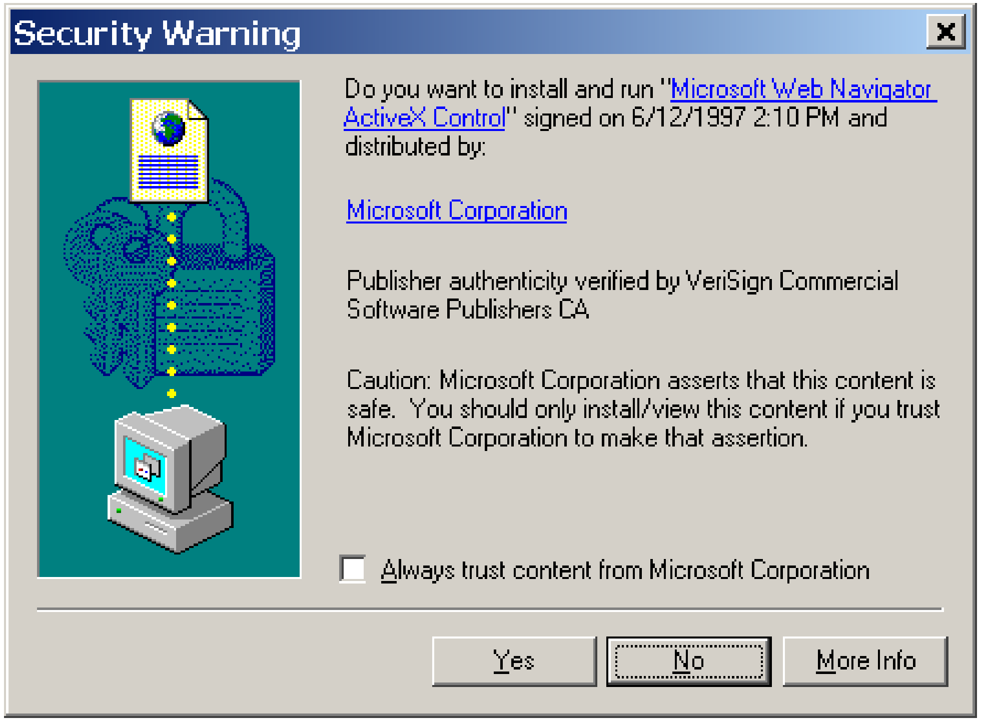
Stories: Netscape font / passwd bug; Shockwave plug-in

ActiveX

- ActiveX controls reside on client's machine, activated by HTML object tag on the page
  - ActiveX controls are not interpreted by browser
  - Compiled binaries executed by client OS
  - Controls can be downloaded and installed
- Security model relies on three components
  - Digital signatures to verify source of binary
  - IE policy can reject controls from network zones
  - Controls marked by author as *safe for initialization*, *safe for scripting* which affects the way control used

Once accepted, installed and started, no control over execution

Installing Controls



Security Warning

Do you want to install and run "Microsoft Web Navigator ActiveX Control" signed on 6/12/1997 2:10 PM and distributed by:

Microsoft Corporation

Publisher authenticity verified by VeriSign Commercial Software Publishers CA

Caution: Microsoft Corporation asserts that this content is safe. You should only install/view this content if you trust Microsoft Corporation to make that assertion.

□ Always trust content from Microsoft Corporation

If you install and run, no further control over the code.

In principle, browser/OS could apply sandboxing, other techniques for containing risks in native code. But don't count on it.

Risks associated with controls

- MSDN Warning
  - An ActiveX control can be an extremely insecure way to provide a feature
- Why?
  - A COM object, control can do any user action
    - read and write Windows registry
    - access the local file system
  - Other web pages can attack a control
    - Once installed, control can be accessed by any page
    - Page only needs to know class identifier (CLSID)
- Recommendation: use other means if possible

http://msdn.microsoft.com/library/default.asp?url=/code/list/ie.asp

IE Browser Helper Objects (Extensions)

- COM components loaded when IE starts up
- Run in same memory context as the browser
- Perform any action on IE windows and modules
  - Detect browser events
    - GoBack, GoForward, and DocumentComplete
  - Access browser menu, toolbar and make changes
  - Create windows to display additional information
  - Install hooks to monitor messages and actions
- Summary: No protection from extensions

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebgen/html/bho.asp

Dynamic content

- Servers often generate client-specific content
  - E.g., your shopping cart, your portal home page, ...
- Simplest method: CGI programs
  - Client connects to server
  - Server spawns CGI program in a new process
  - Script generates contents of web page
- Problem: slow
  - Interpreters (perl, python, php) slow to start up
  - Even creating processes is somewhat slow

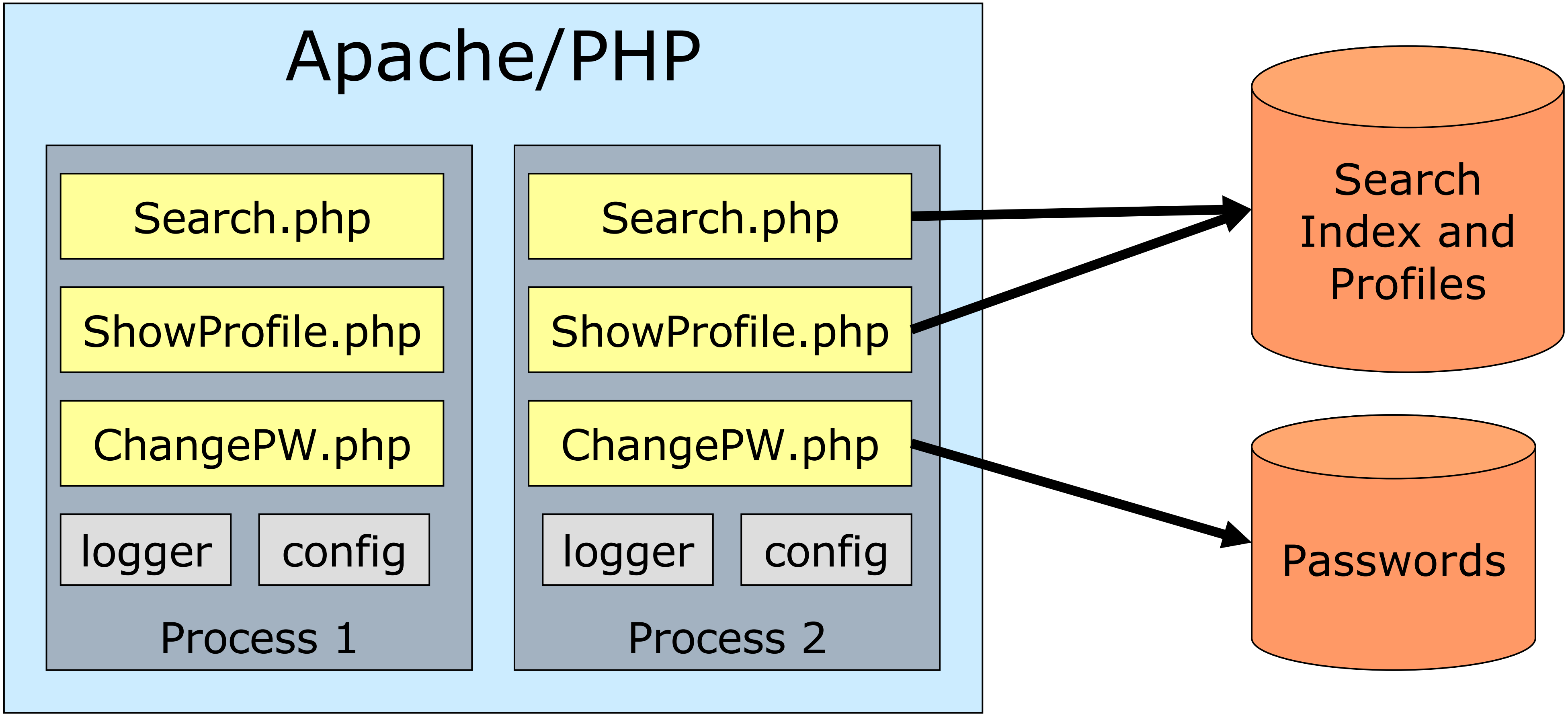
Solution: Embeded interpreter

- Embed script interpreter into web server
  - Eliminates need to spawn a process per connection
  - Eliminates need to keep re-parsing same script
- Structure server as pool of workers
  - Pre-spawn many identical server processes
  - Any free server can handle any connection
- Problem: Isolation

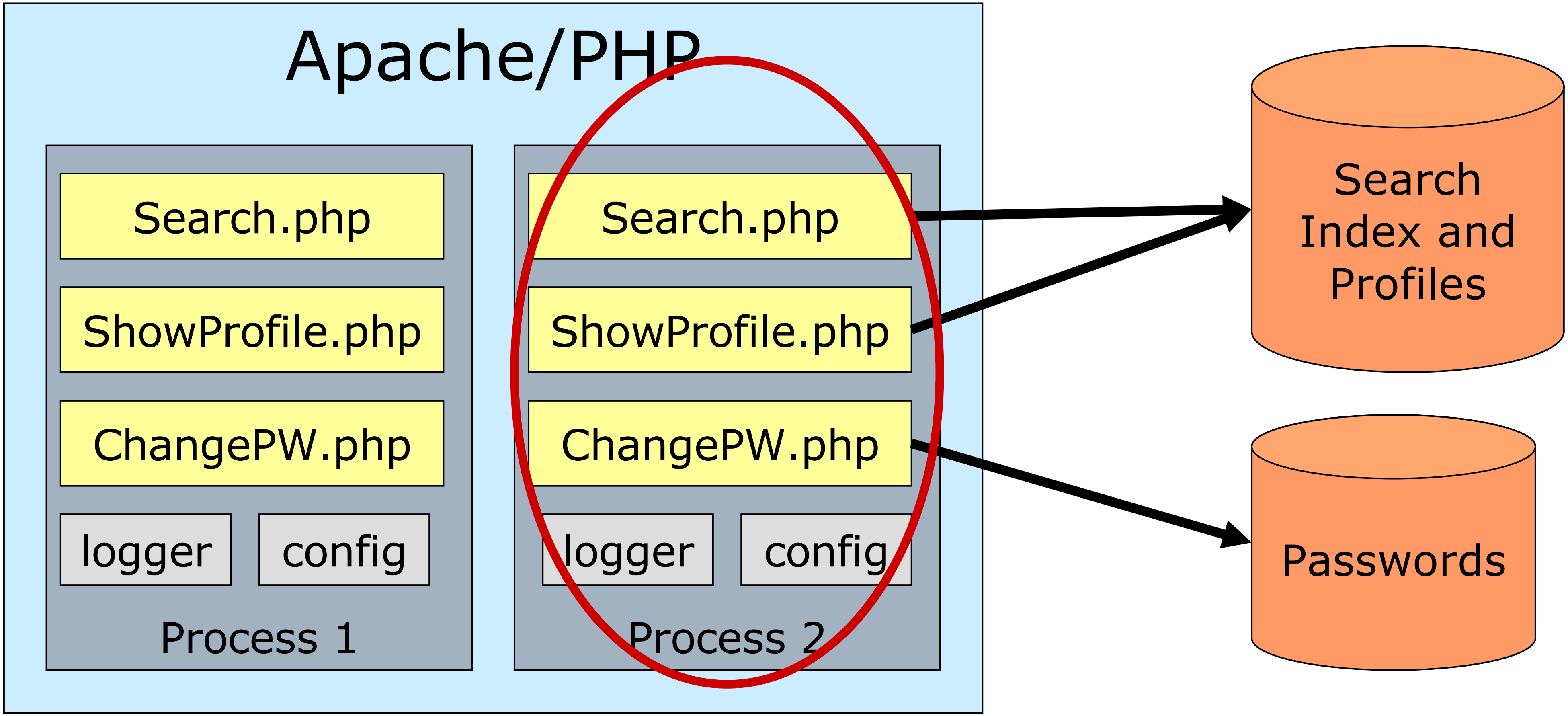
Example: Apache/PHP

- History of buffer overruns in Apache & PHP
- Bugs allow escape from chroot-like PHP feature
- Users often introduce bugs in PHP scripts
  - E.g., SQL injection (download list of users)
  - E.g., forget to check for “../../” in path
- Performance often requires other C code
  - Which introduced more overruns, etc.

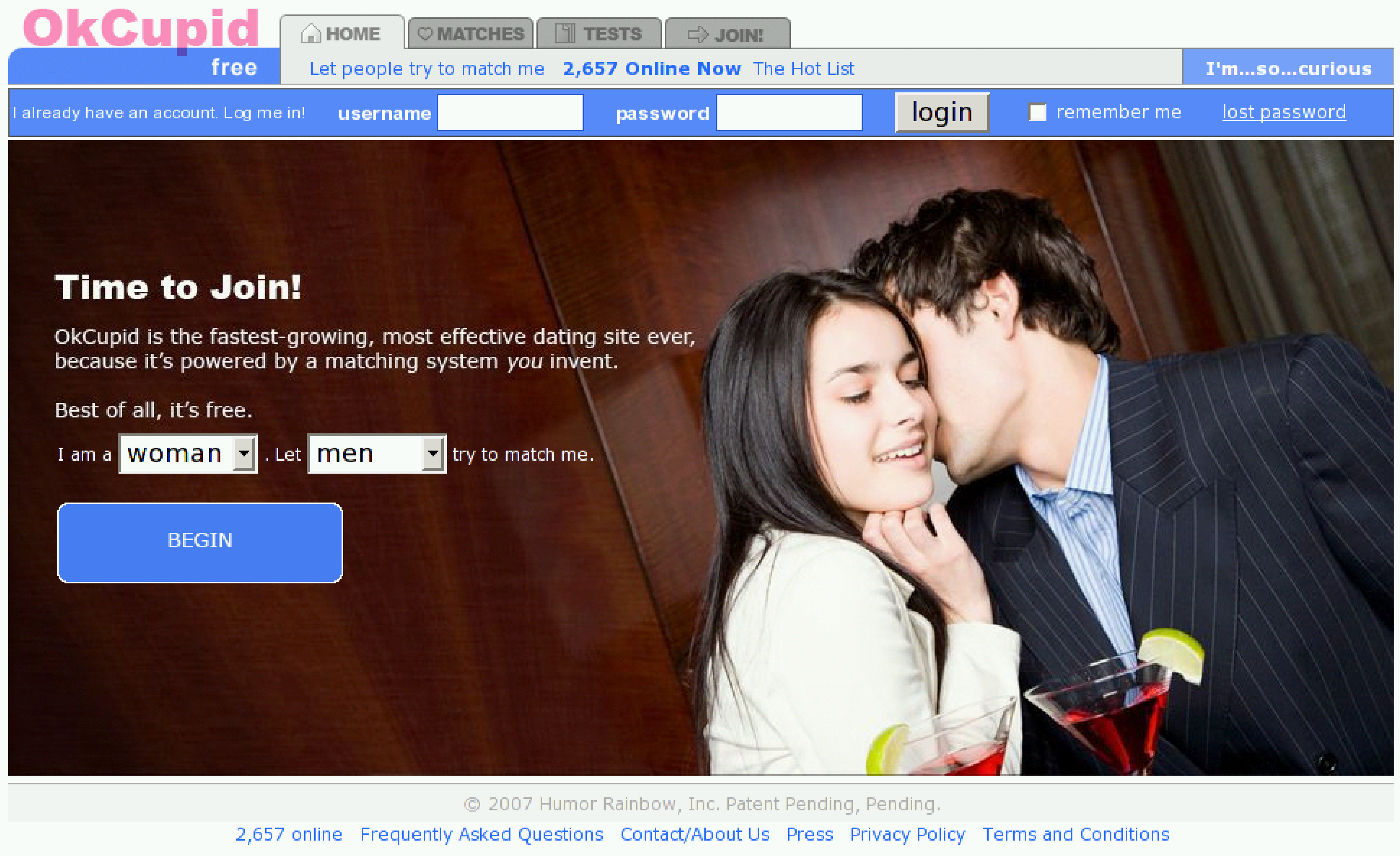
Apache/PHP Isolation



Apache/PHP Isolation



OKWS web server [Krohn]



OkCupid free

HOME MATCHES TESTS JOIN!

Let people try to match me **2,657 Online Now** The Hot List **I'm...so...curious**

I already have an account. Log me in! **username** **password** login □ remember me lost password

Time to Join!

OkCupid is the fastest-growing, most effective dating site ever, because it's powered by a matching system *you* invent.

Best of all, it's free.

I am a woman . Let men try to match me.

BEGIN

© 2007 Humor Rainbow, Inc. Patent Pending, Pending.

2,657 online Frequently Asked Questions Contact/About Us Press Privacy Policy Terms and Conditions

- Attempt to achieve performance and security
- As secure as possible given Unix underneath
- Used for production web site okcupid.com

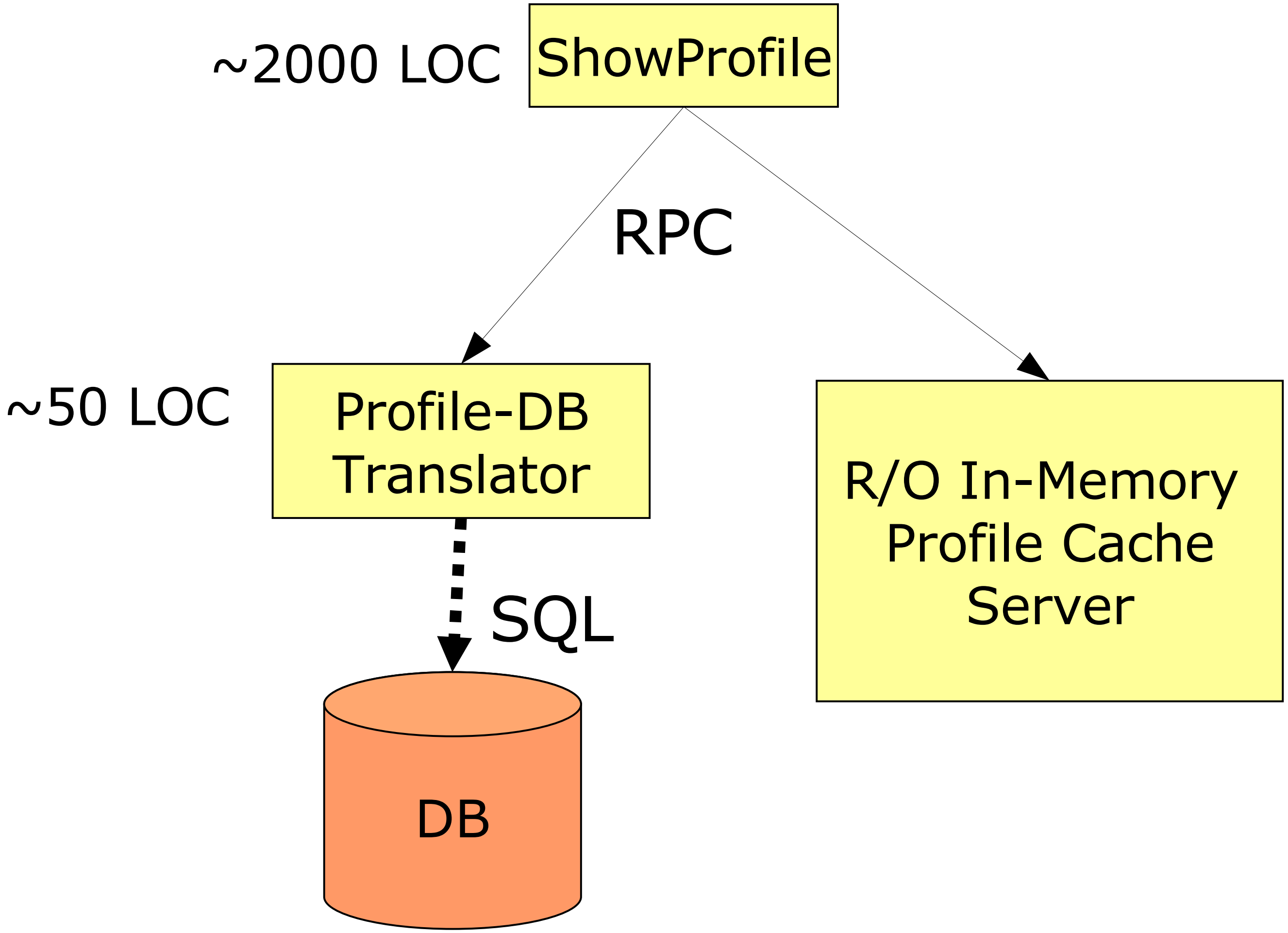
OKWS Design

- A Web site consists of many Web *services*.
  - e.g., Search, ShowProfile, ChangePW
  - A and B are distinct services if they access different pools of data.
- **One-to-one mapping between Web Services and Unix processes.**

OKWS Isolation Strategy

- Process pool fixed at startup (~10).
- Each obeys least-privilege principle.
- Isolates processes:
  - From SQL database access
  - From each other
  - From the OS (filesystem in particular)
  - From DBs they need not access.

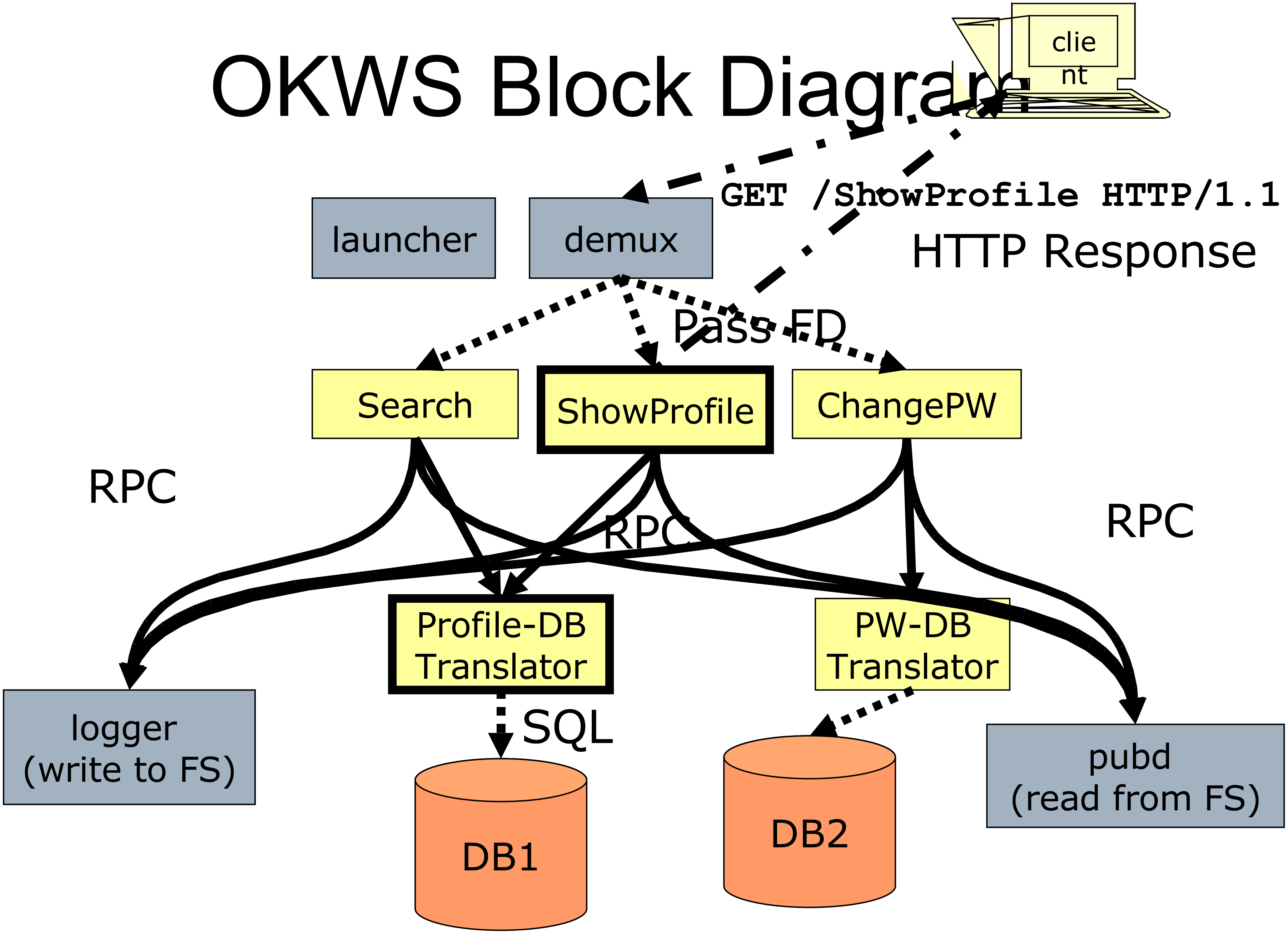
How To Build a Web Service



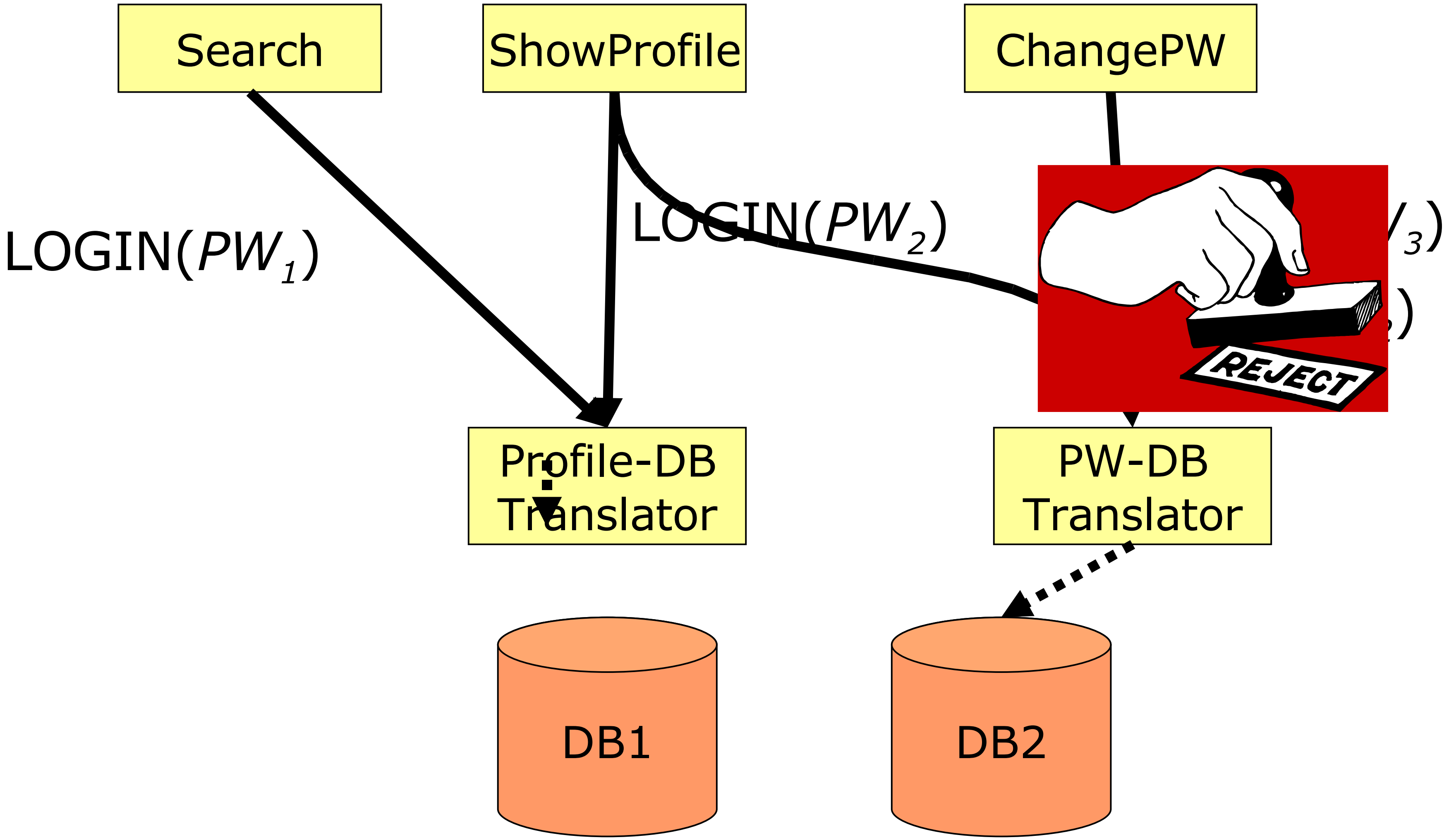
Structured DB Interface

- SQL Alone
  - Allow **ShowProfile** to **`SELECT`** from the **`PROFILES`** table.
  - Allows **ShowProfile** to "**`SELECT * FROM PROFILES`**"`;`
- SQL + RPC-to-SQL Translator
  - Allow **ShowProfile** to read a profile from the database for a given user ID.

OKWS Block Diagram



Isolating DBs



Search

ShowProfile

ChangePW

LOGIN(PW_1)

LOGIN(PW_2)

/$_3$)

)

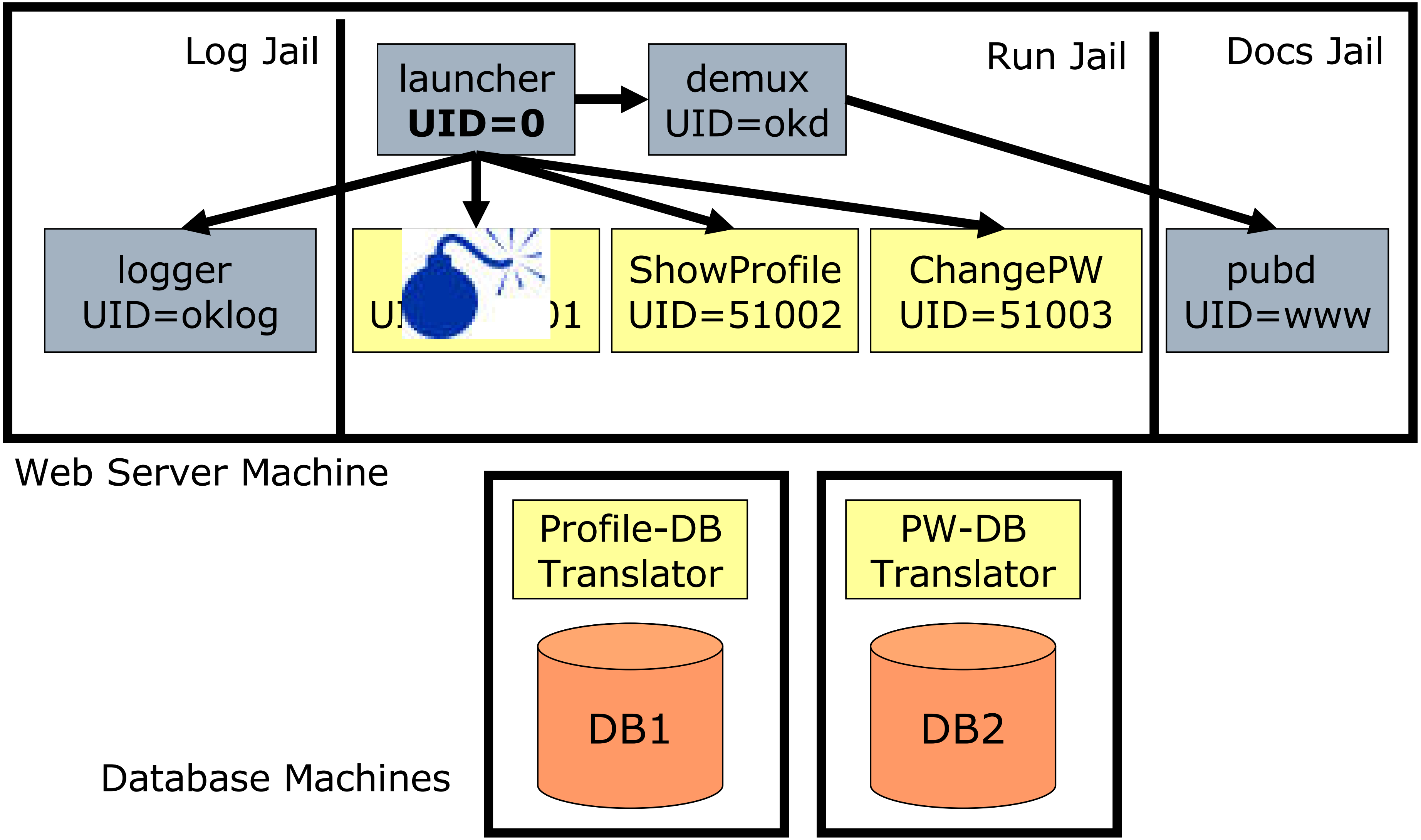
Profile-DB Translator

PW-DB Translator

DB1

DB2

OKWS Process Isolation



If Service A is Compromised…

- cannot access its own DB outside the RPC interface provided.
- cannot access *setuid* executables.
- cannot access logs, config files, source files, privileged ports.
- cannot send service B signals
- cannot trace service B's system calls
- cannot access B's database

OKWS limitations

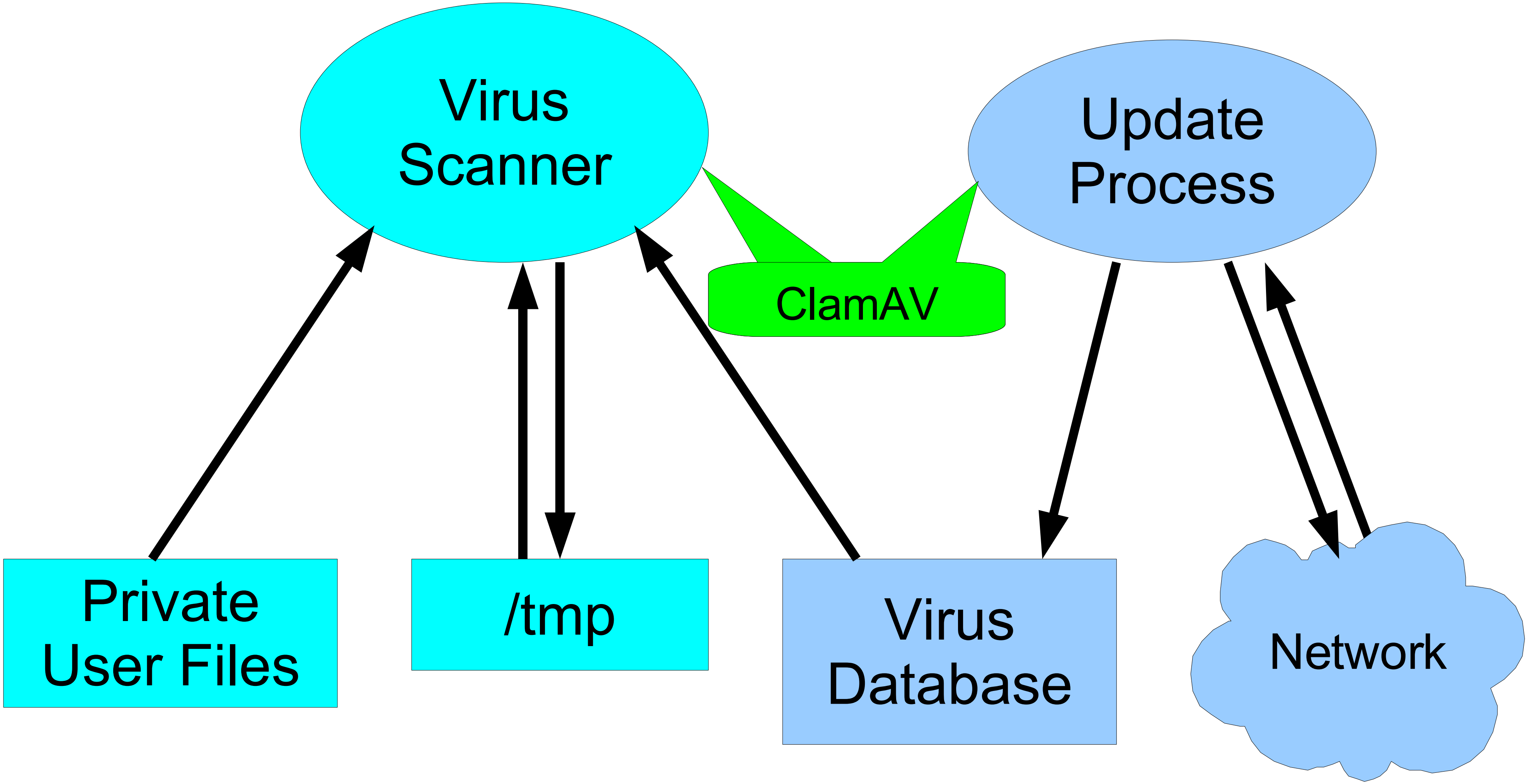
- No isolation within a service
  - Implemented by Unix process
  - E.g., buffer overrun would allow one user to see another user's data
- Many bugs lead to data disclosure
- How to provide better isolation?
  - Maybe launch one process per connection
  - But very expensive, need different DB interface
- To do it right, might need a new OS

HiStar [Zeldovich et al.]

- Resurrect MAC ideas for very different domain
- OS that makes all information flow explicit
- Idea: Damage from bug can only spread where information can flow
  - If A can't communicate with B
  - Then A can't subvert B's proper operation
  - And A can't learn B's private information
- Force cross-user information flows to go through small, well-understood code

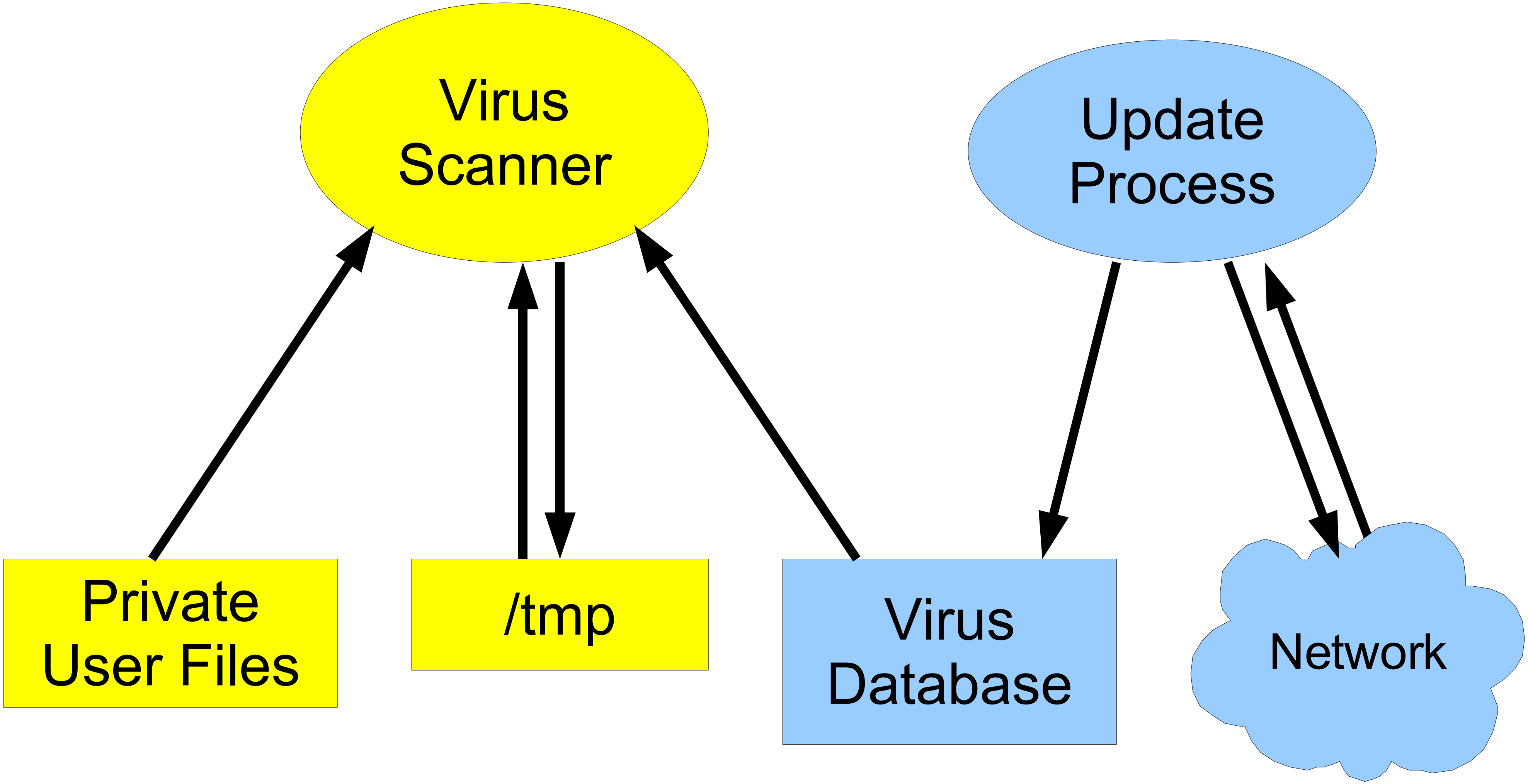
Review: Covert channels on Unix

- E.g., how to prevent virus scanner leaking file?



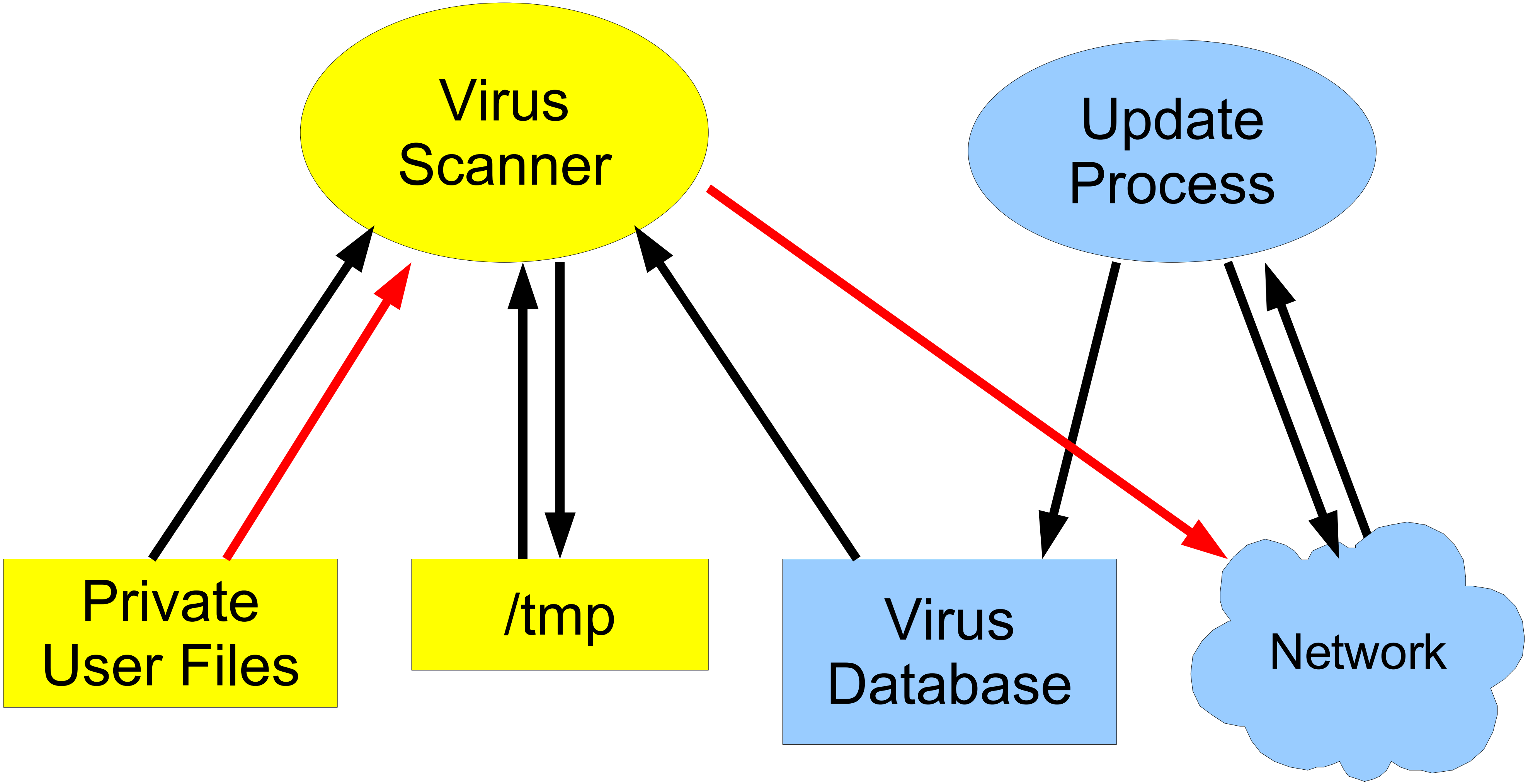
- Goal: private files cannot go onto the network

Information Flow Control



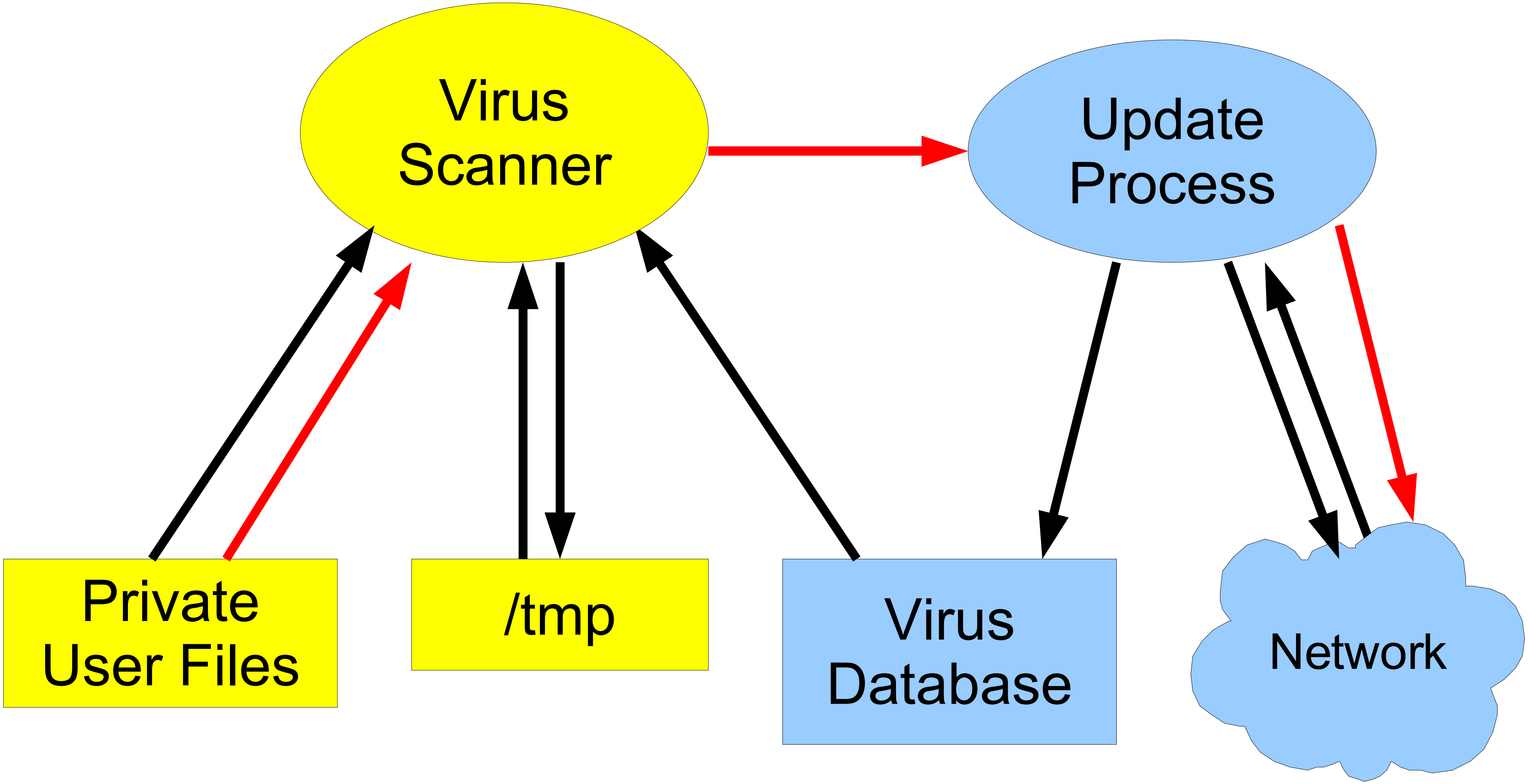
- Goal: private files cannot go onto the network

Buggy scanner leaks private data



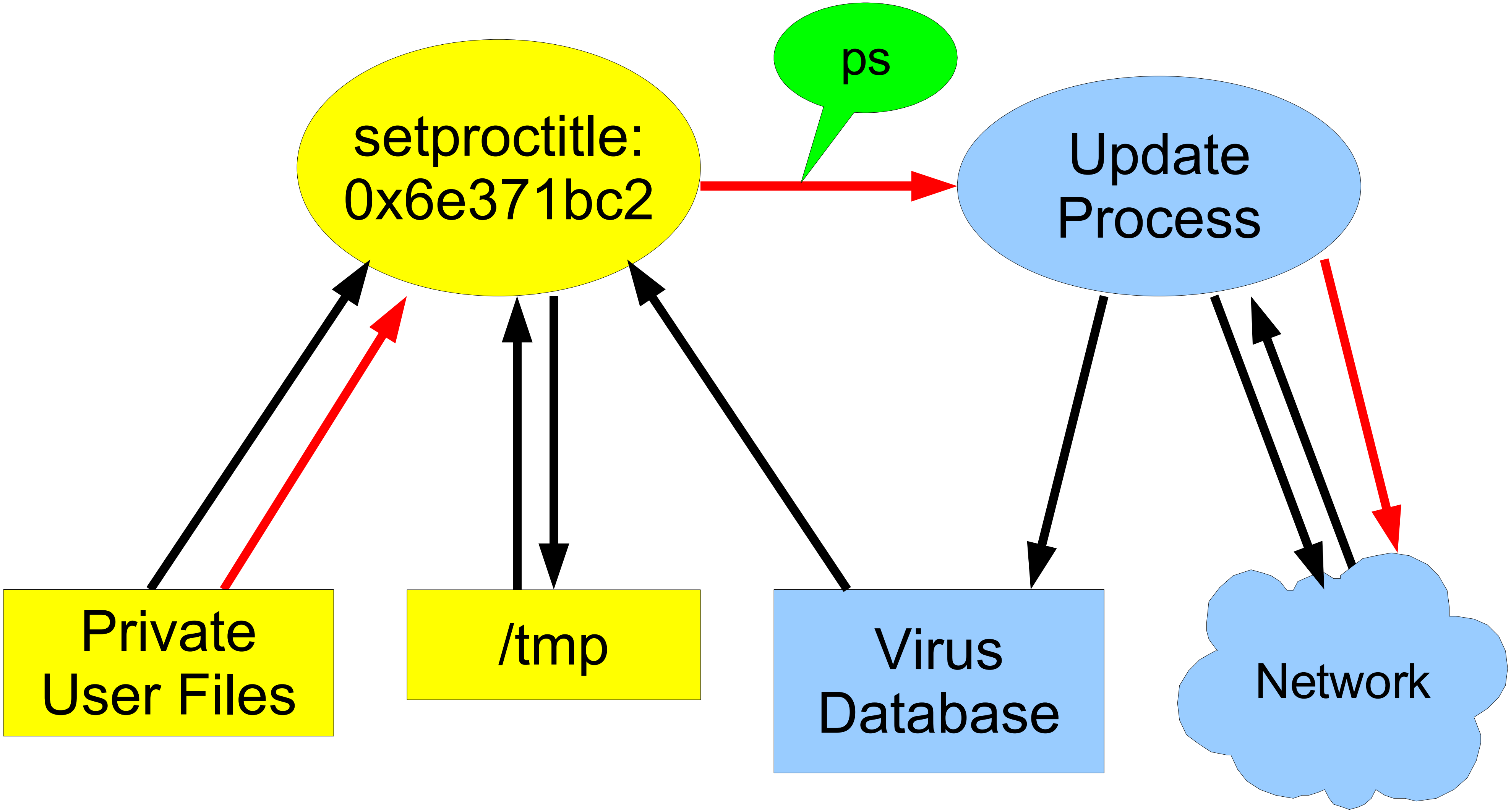
- Must restrict sockets to protect private data

Buggy scanner leaks private data



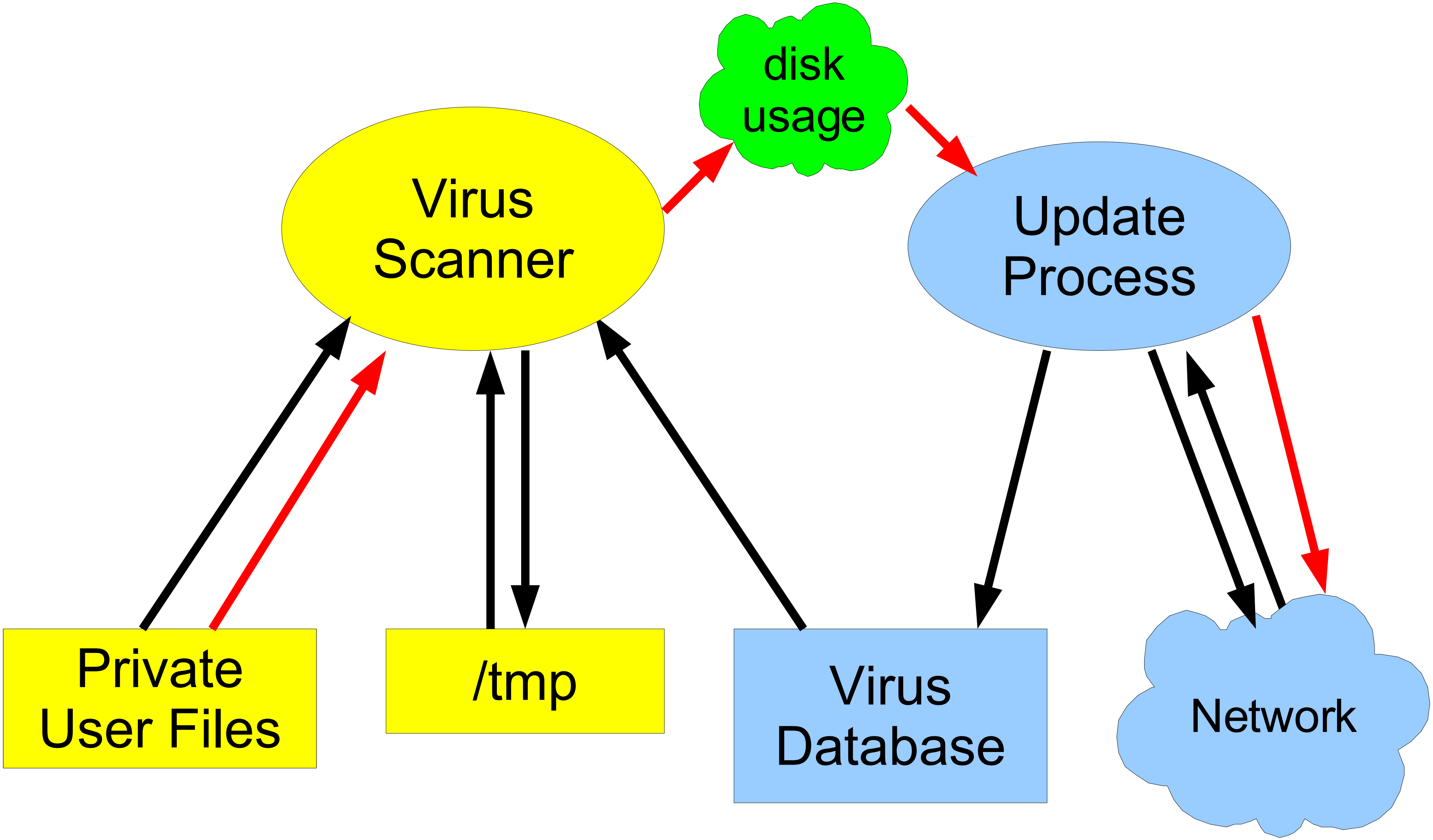
- Must restrict scanner's ability to use IPC

Buggy scanner leaks private data



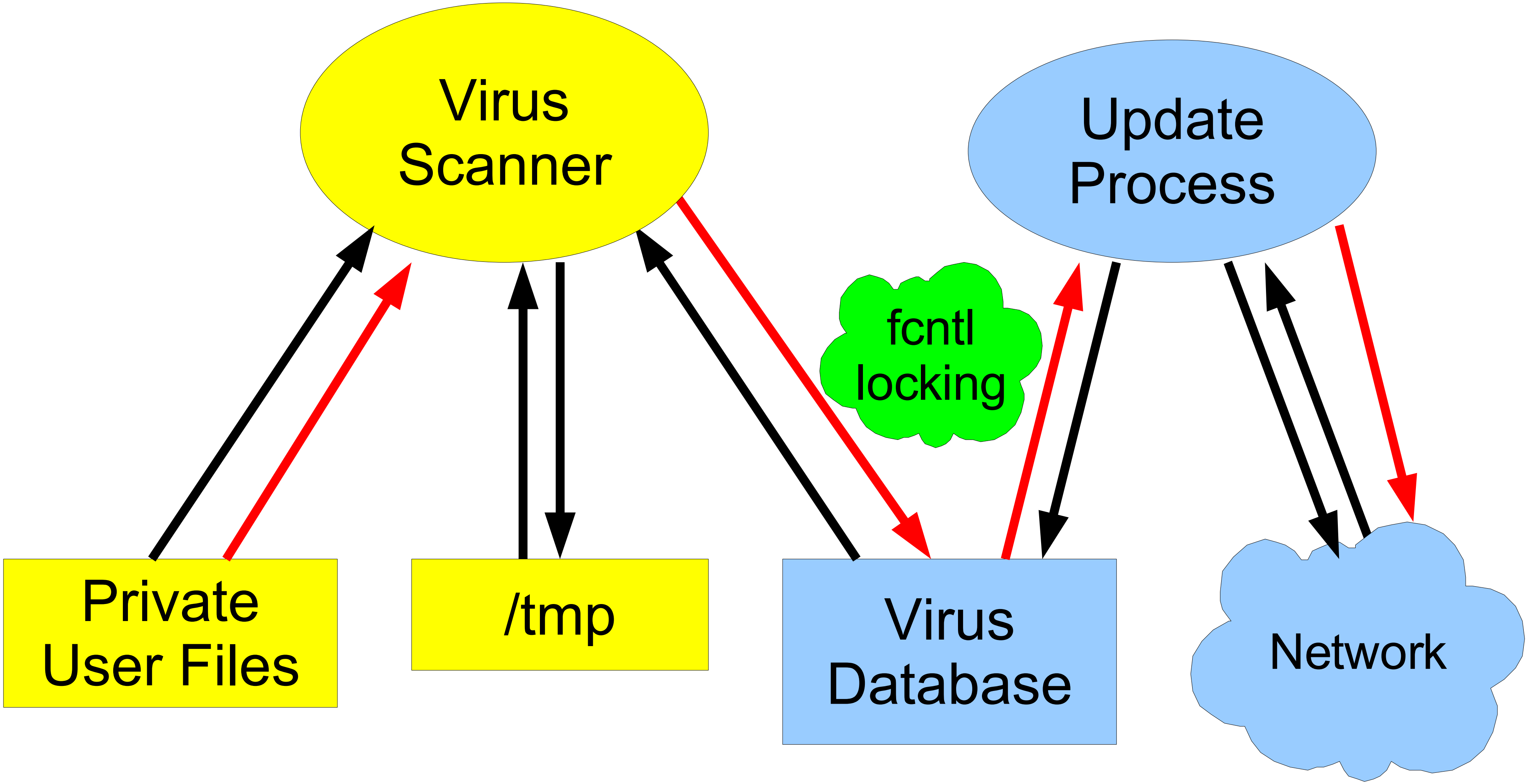
- Must restrict access to /proc, ...

Buggy scanner leaks private data



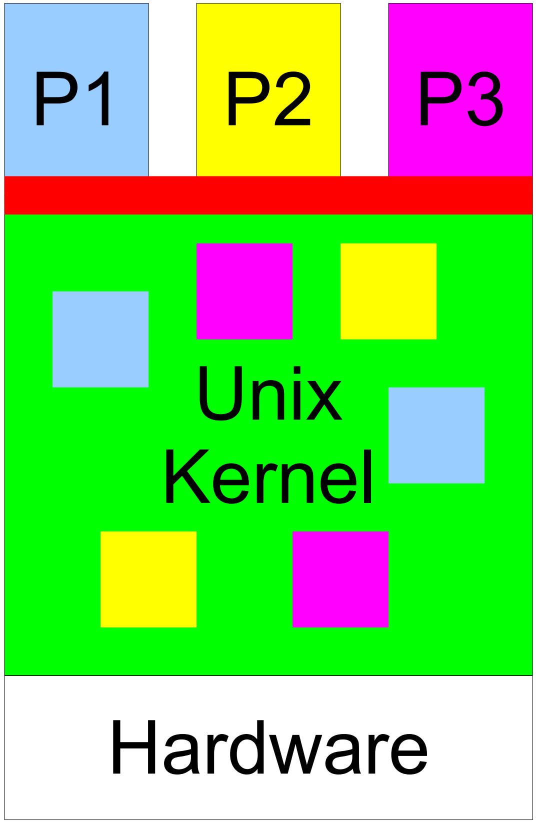
- Must restrict FS'es that virus scanner can write

Buggy scanner leaks private data



- List goes on – is there any hope?

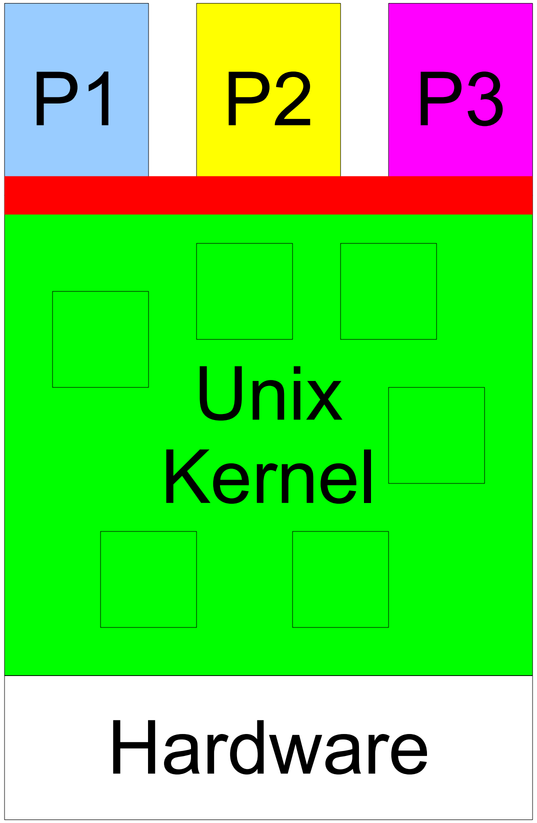
What's going on?



Unix

- Kernel not designed to enforce these policies
- Retrofitting difficult
  - Need to track potentially any memory observed or modified by a system call!
  - Hard to even enumerate

What's going on?

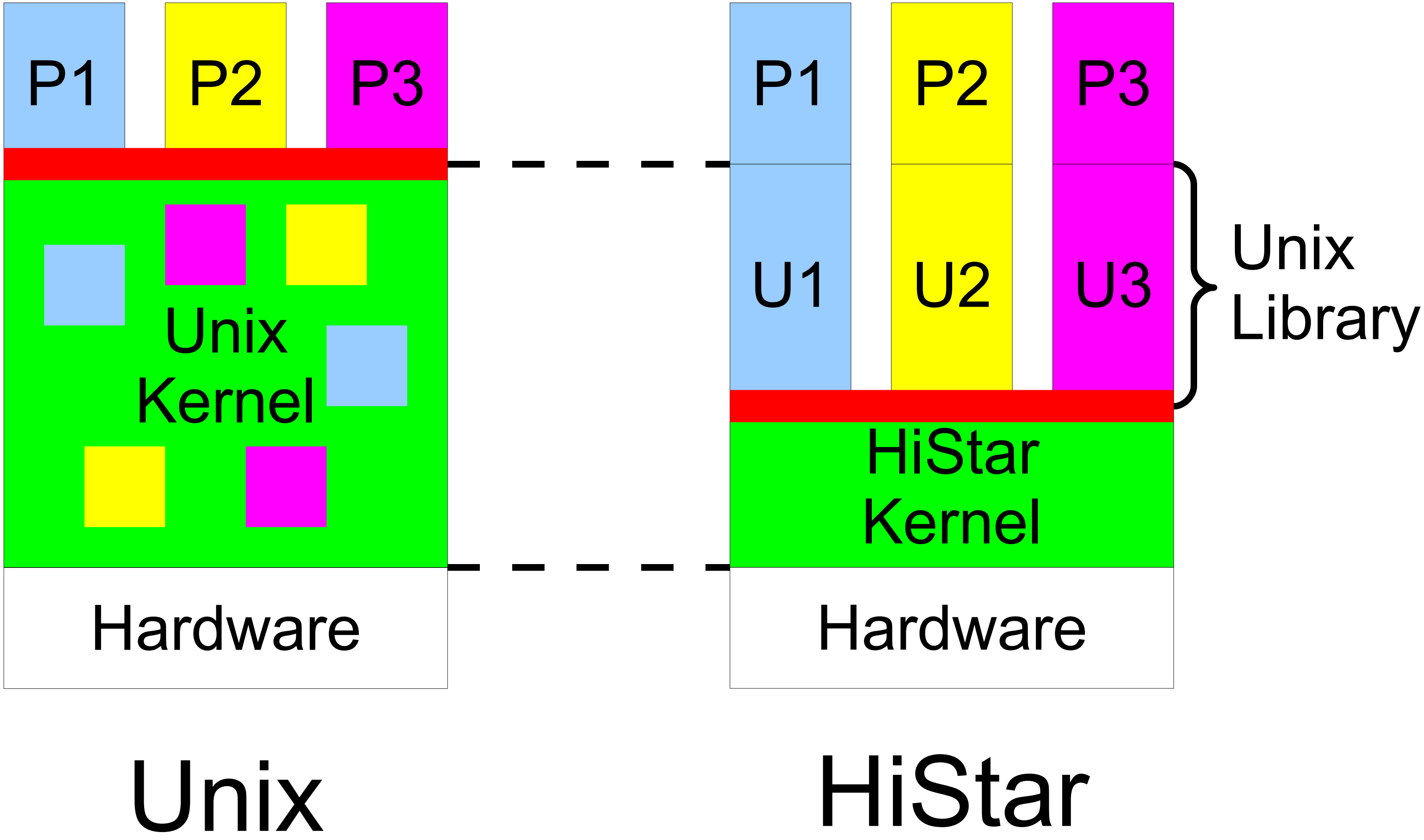


Unix

- Kernel not designed to enforce these policies
- Retrofitting difficult
  - Need to track potentially any memory observed or modified by a system call!
  - Hard to even enumerate

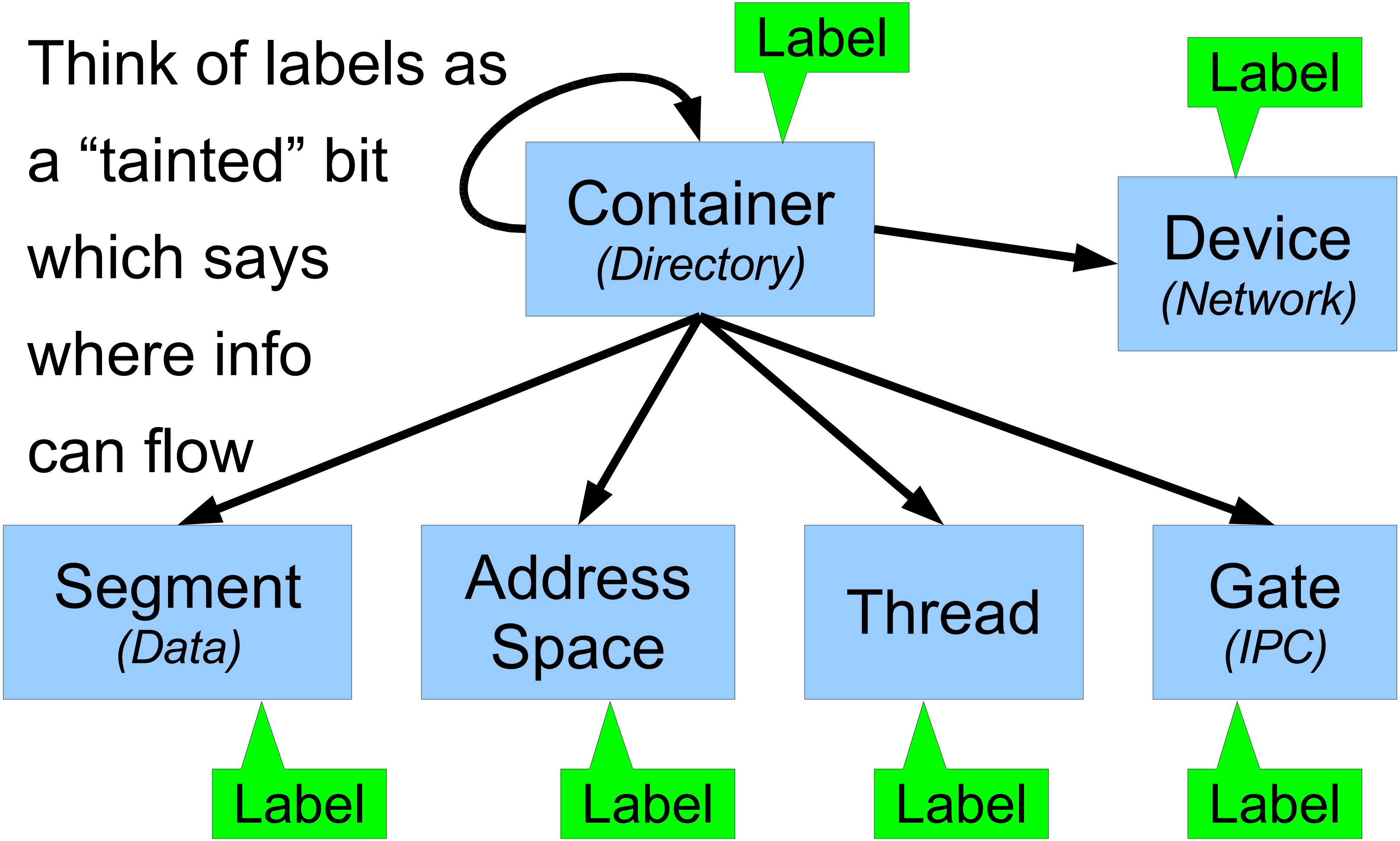
HiStar Solution

- Make all state explicit, track all communication

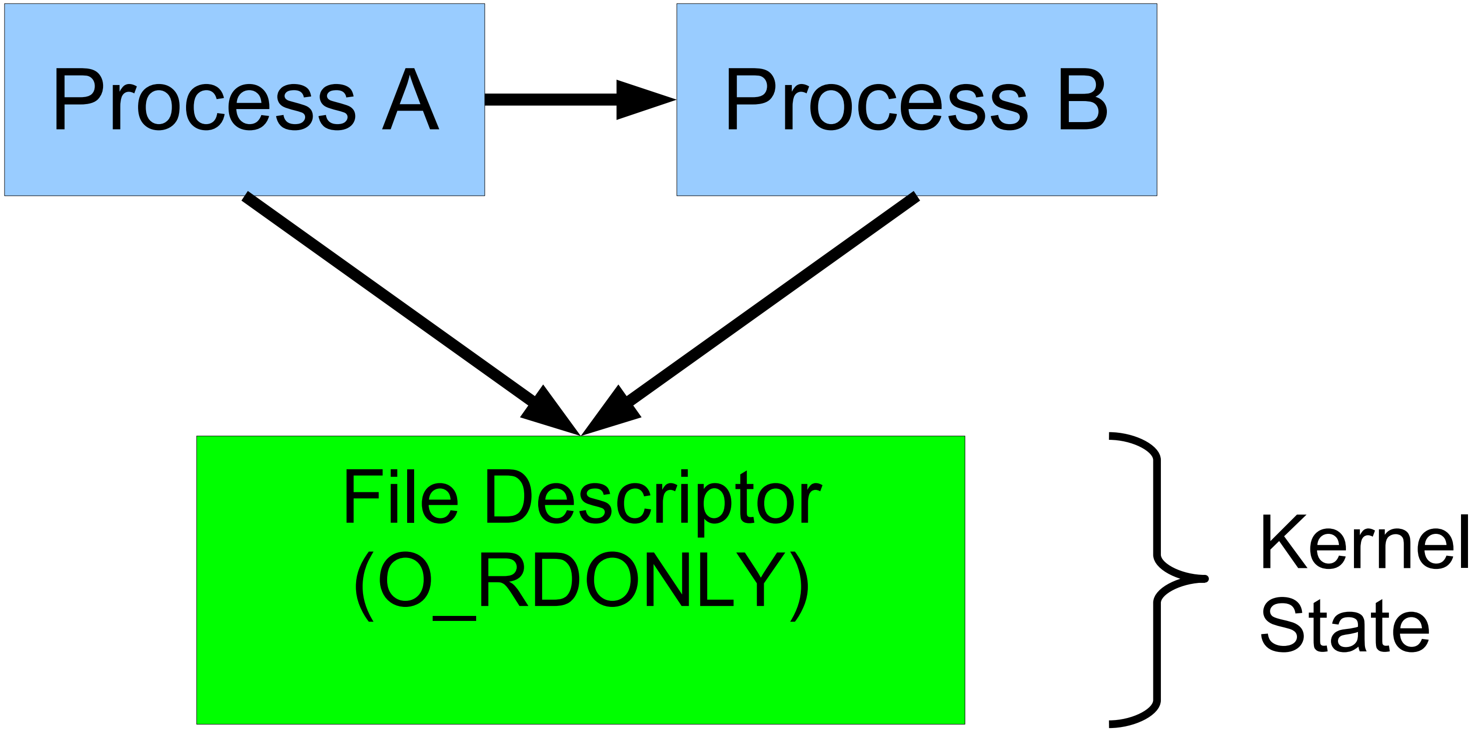


Kernel has only low-level objects

Think of labels as a “tainted” bit which says where info can flow

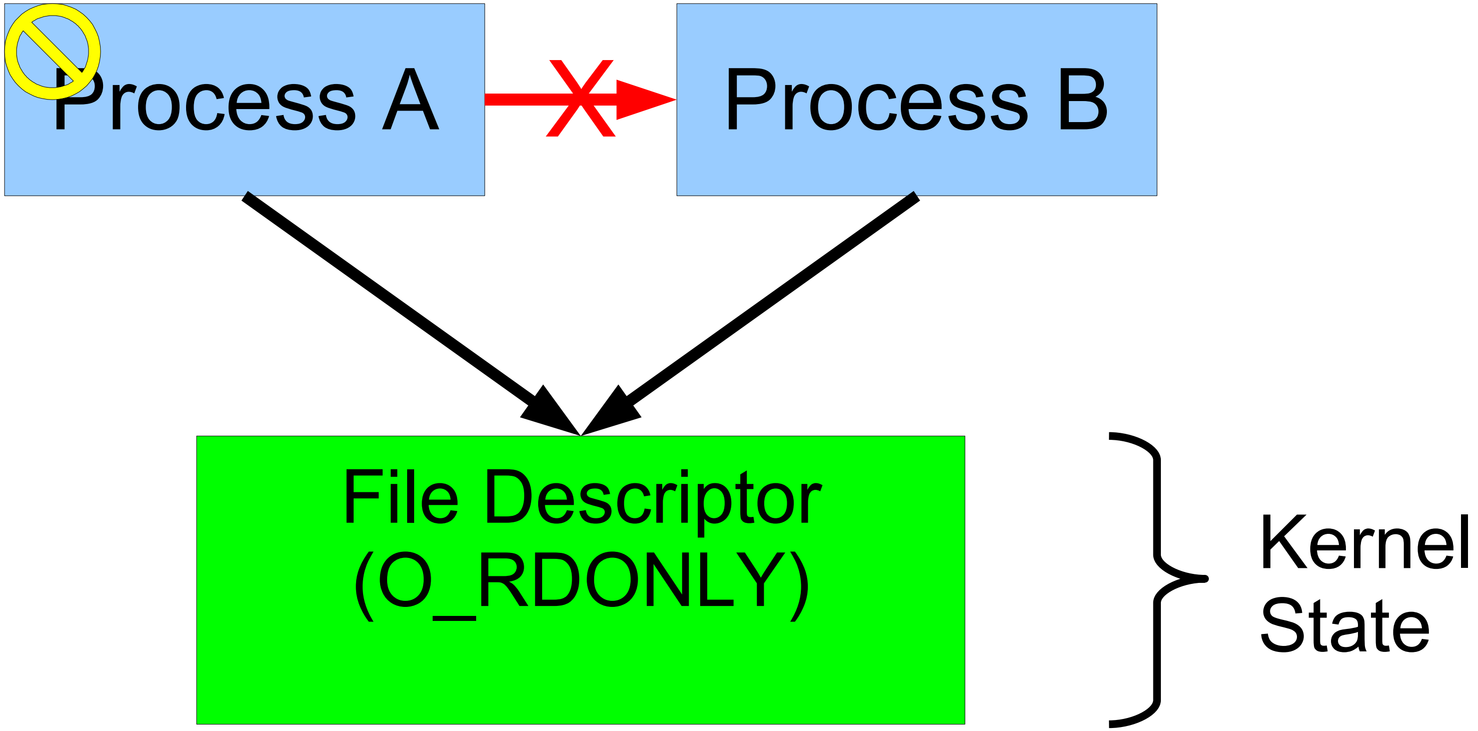


Unix File Descriptors

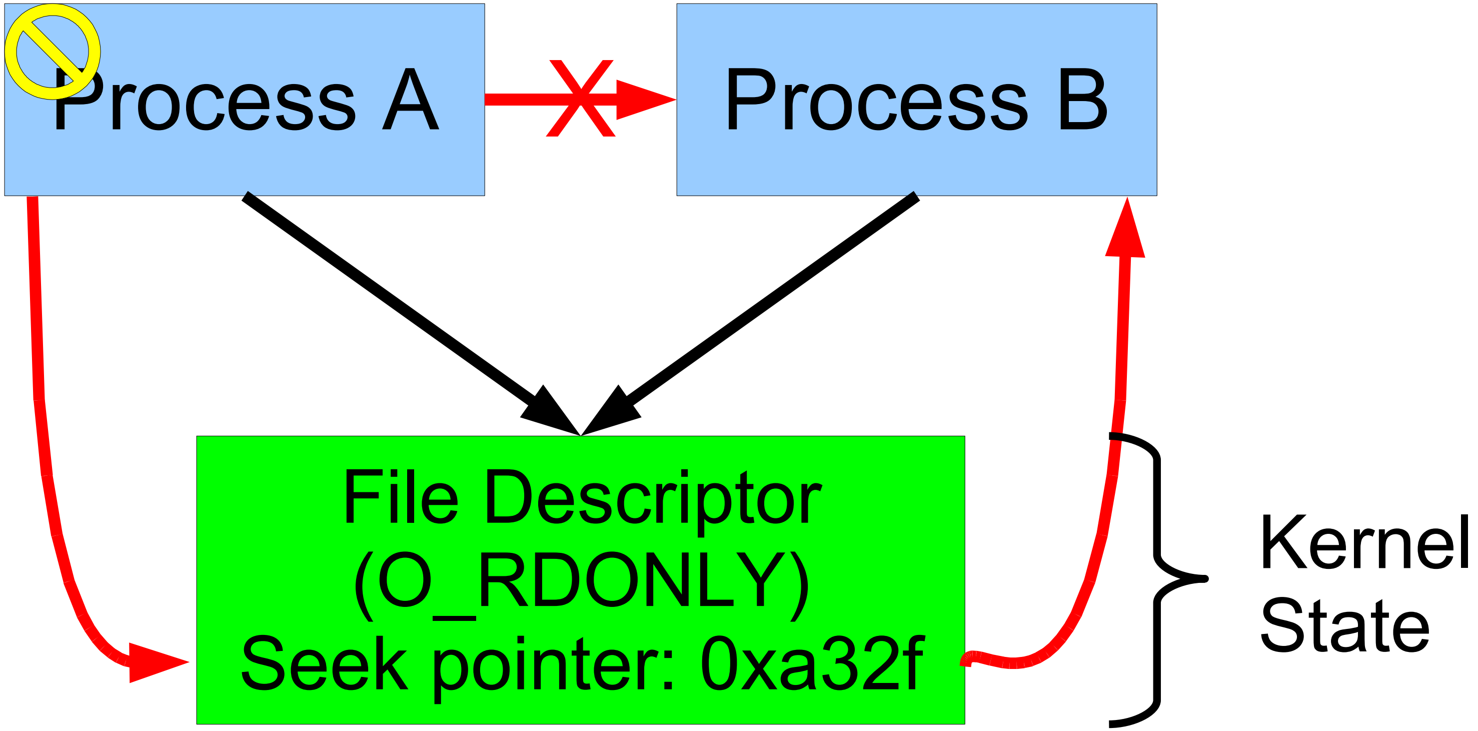


Unix File Descriptors

- Tainted process only talks to other tainted procs

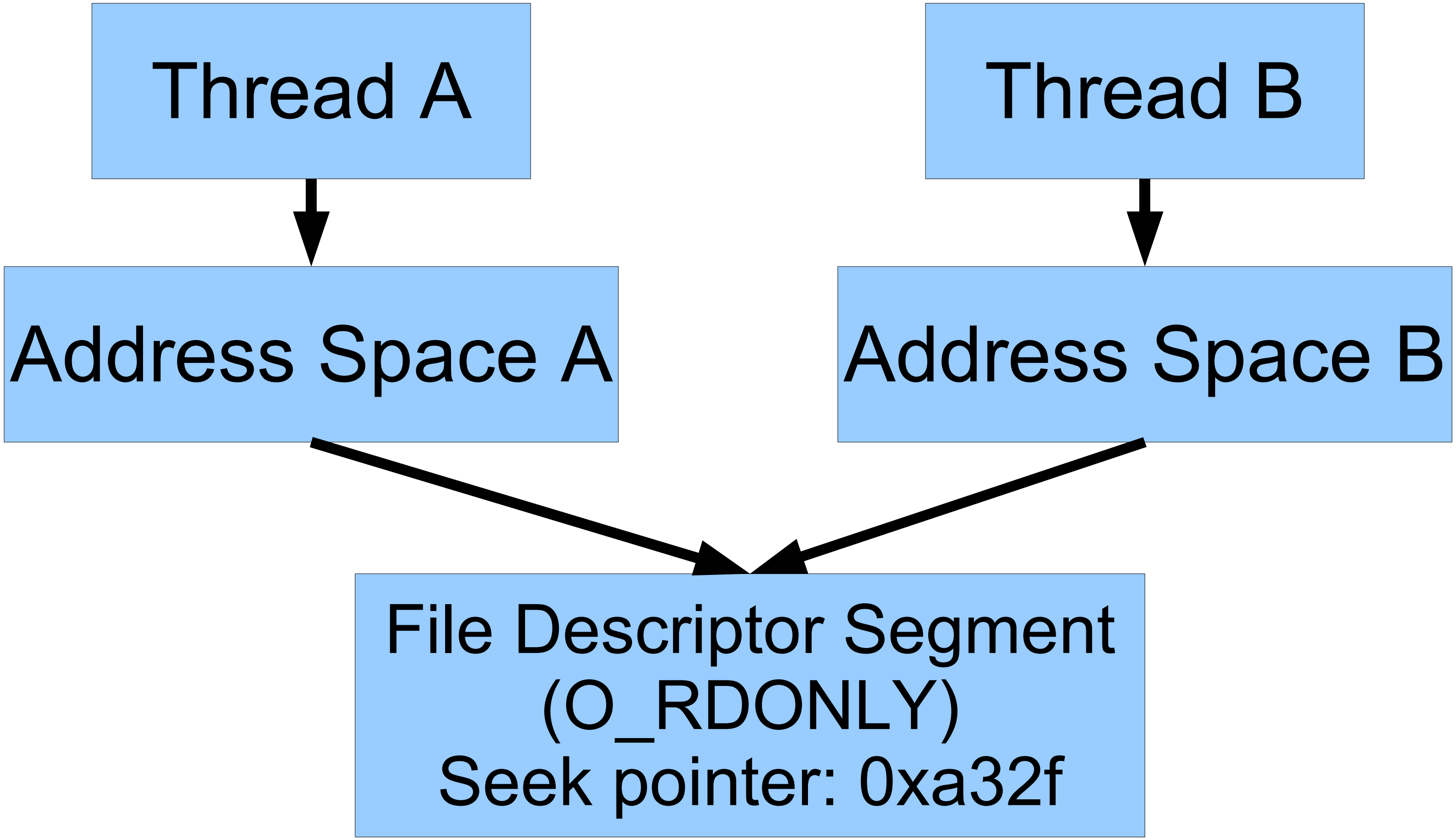


Unix File Descriptors



- Lots of shared state in kernel, easy to miss

HiStar File Descriptors



Thread A

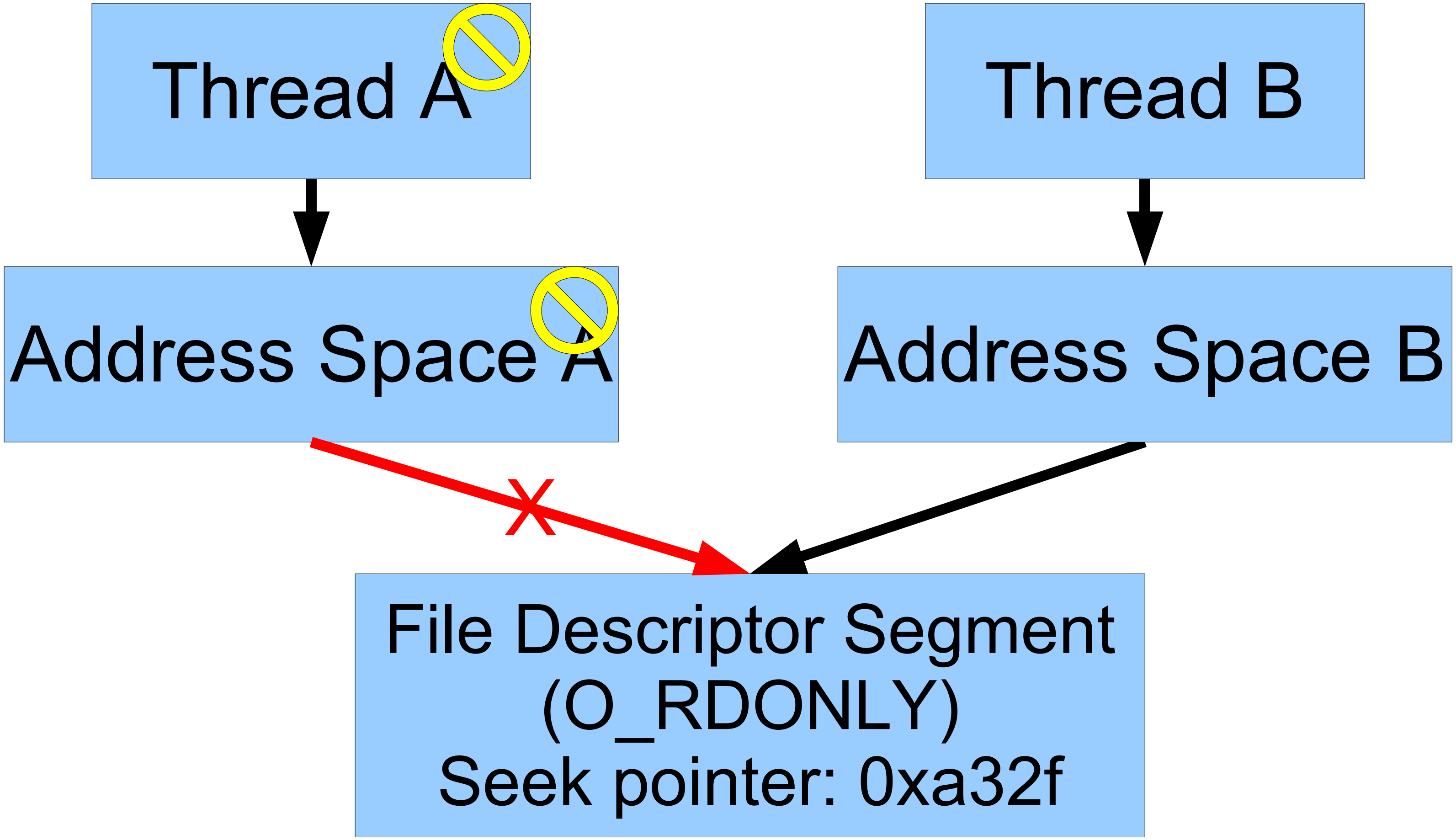
Thread B

Address Space A

Address Space B

File Descriptor Segment
(O_RDONLY)
Seek pointer: 0xa32f

HiStar File Descriptors



Thread A

Thread B

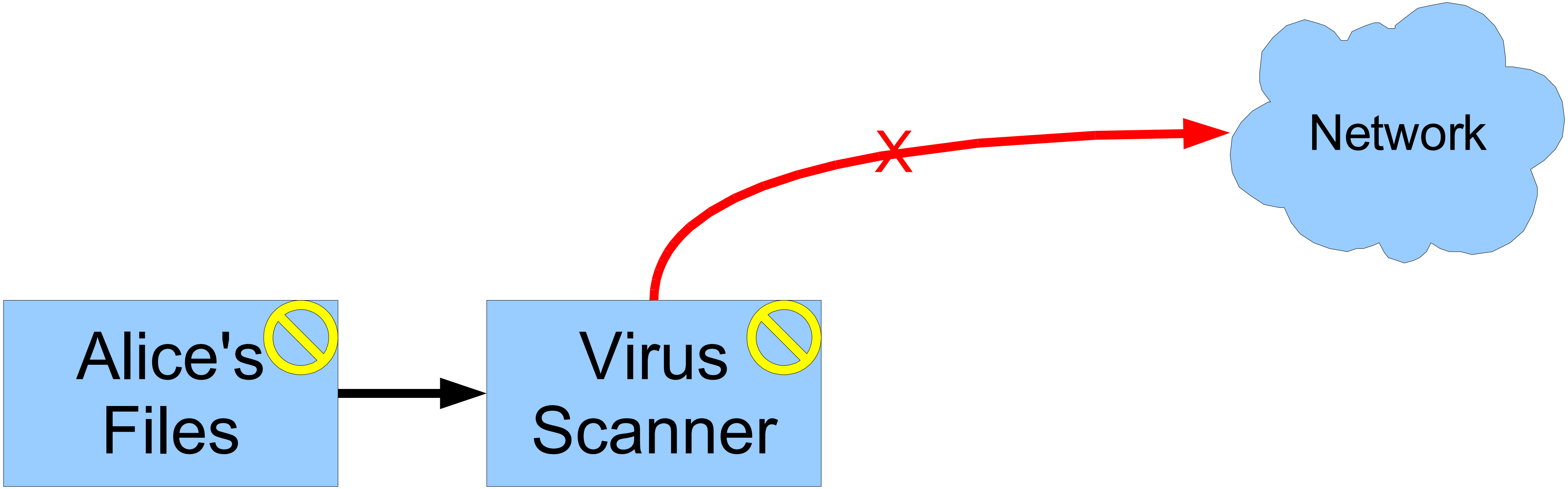
Address Space A

Address Space B

File Descriptor Segment
(O_RDONLY)
Seek pointer: 0xa32f

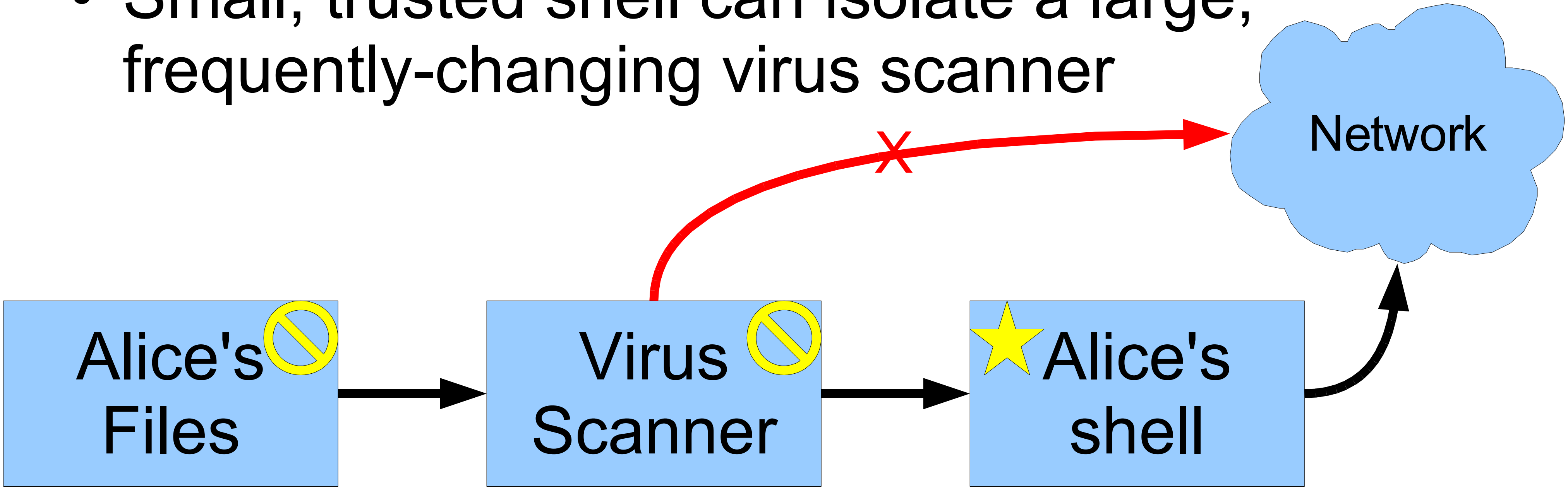
- All shared state is now explicitly labeled
- Just need segment read/write checks

How do we get anything out?

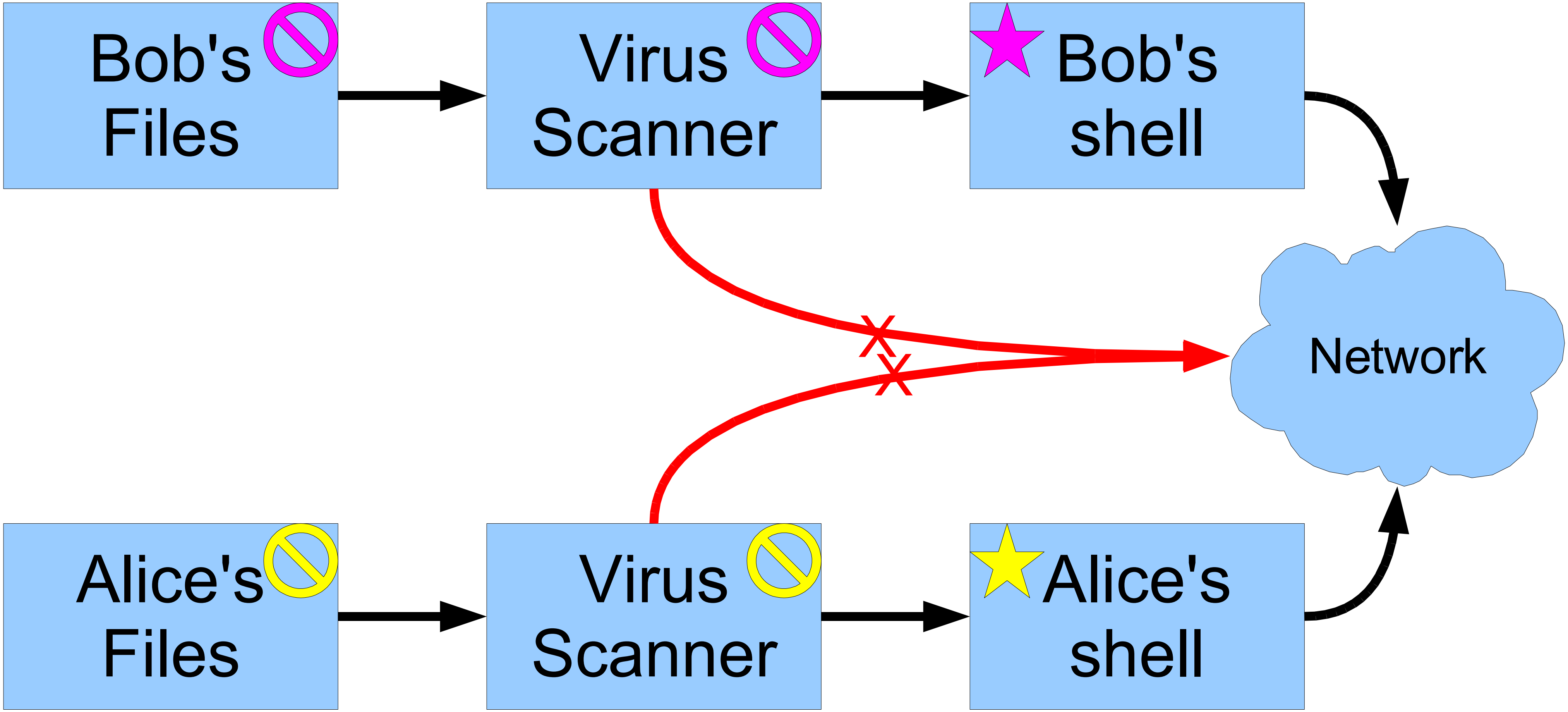


“Owner” privilege

- Yellow objects can only interact with other yellow objects, or objects with yellow star
- Small, trusted shell can isolate a large, frequently-changing virus scanner



Multiple categories of taint

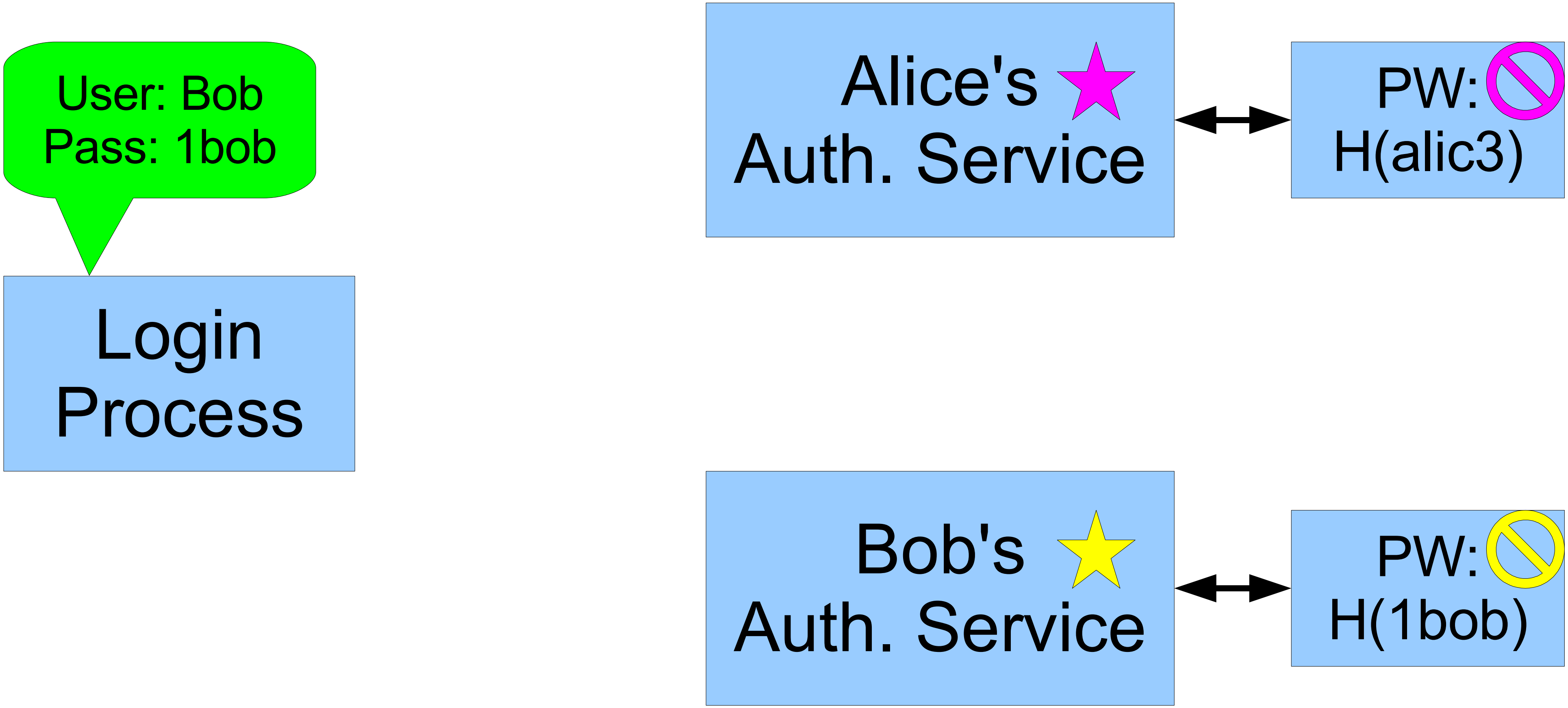


- Owner privilege and information flow control are the only access control mechanism
- Anyone can allocate a new category, gets star

HiStar benefits

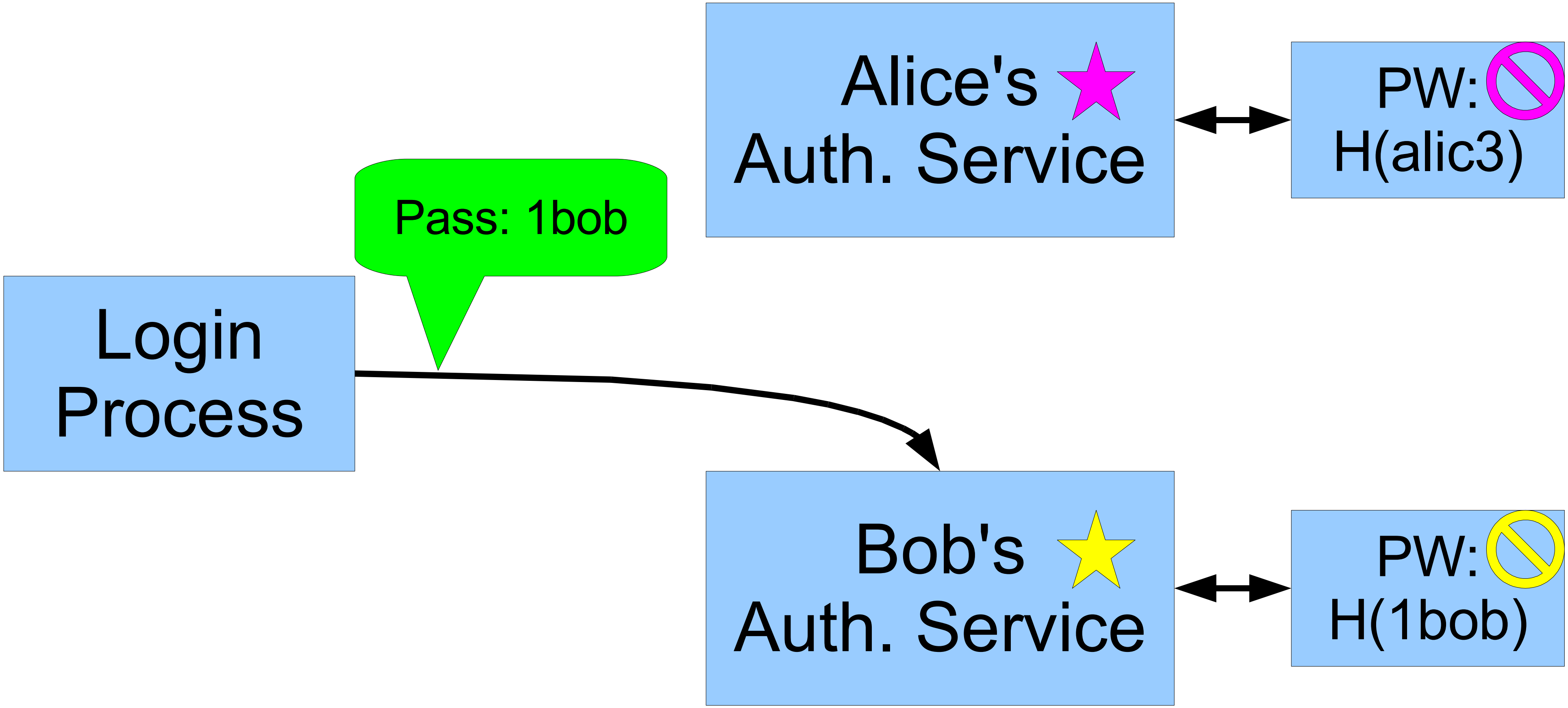
- Can factor applications into many mutually distrustful pieces
- Much of the code can be mostly untrusted
- No need for fully trusted code
  - Even login doesn't need superuser privs
- Flexible enough for web applications
  - Can allocate huge number of categories (e.g., could use one per user account on okcupid.com)
  - Can re-use OS login mechanism for web server

Login on HiStar



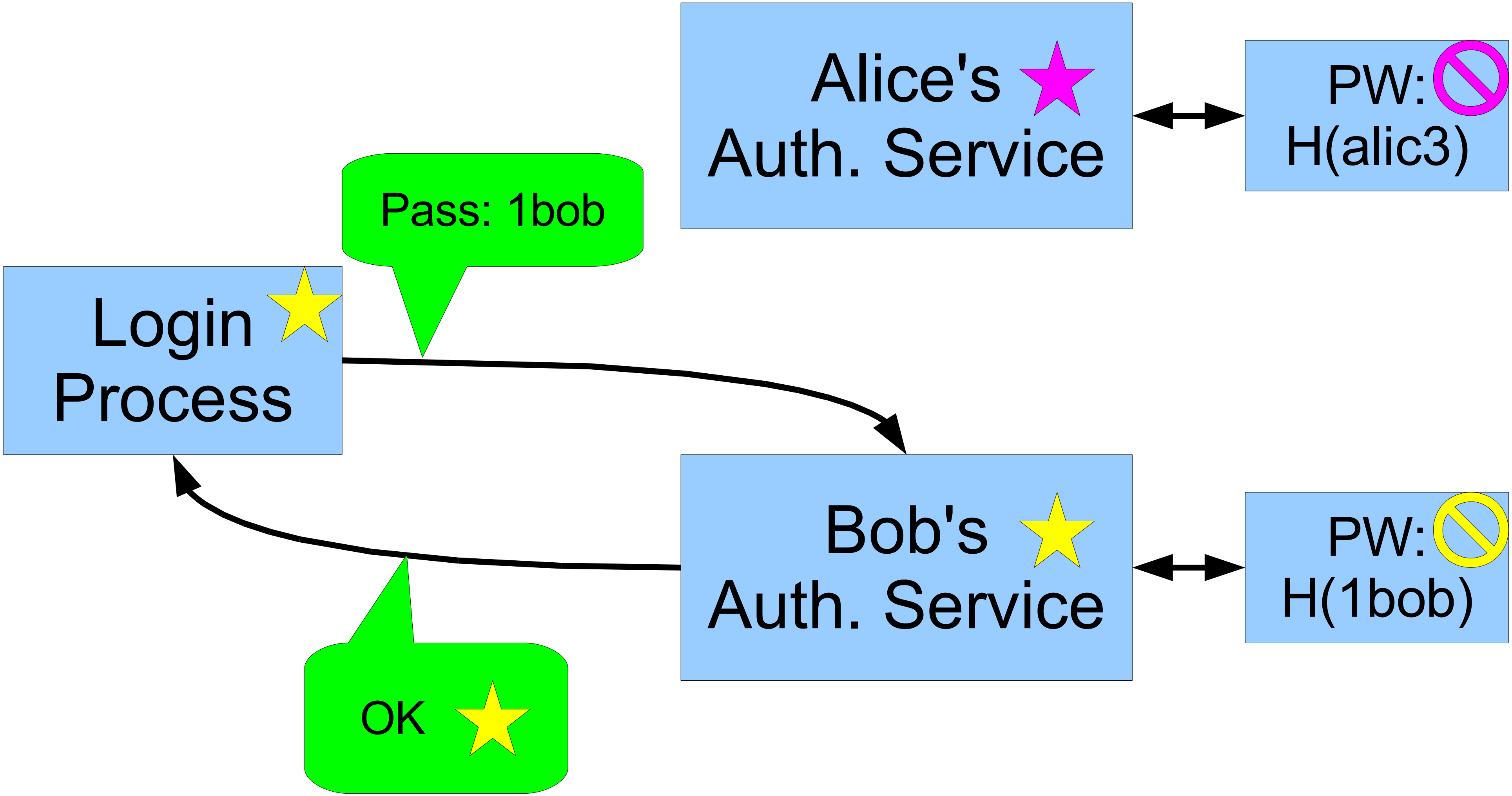
- Each user can provide their own auth. service

Login on HiStar



- Each user can provide their own auth. service

Login on HiStar



Alice's ★
Auth. Service

PW: ⃠
H(alic3)

Pass: 1bob

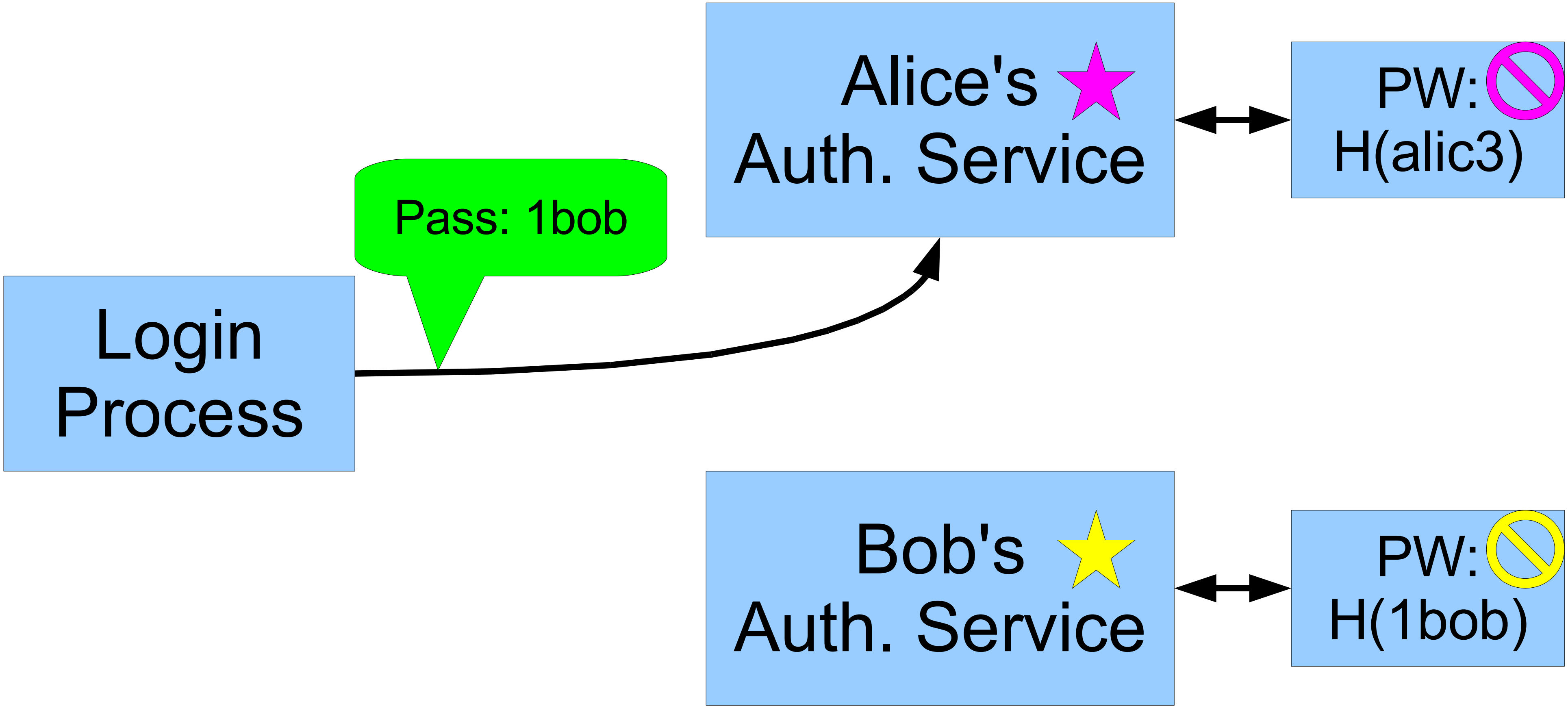
Login ★
Process

Bob's ★
Auth. Service

PW: ⃠
H(1bob)

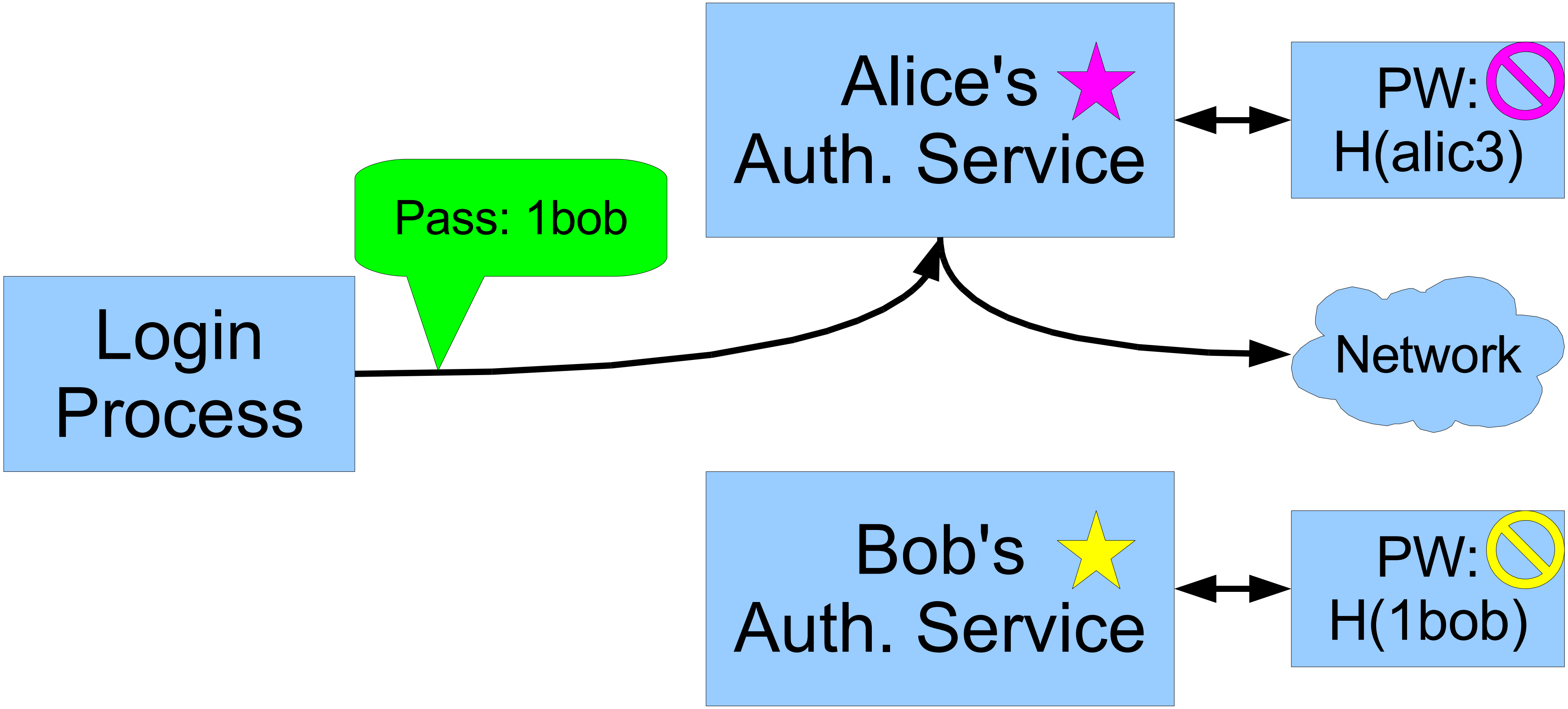
OK ★

Password disclosure



- What if Bob mistypes his username as “alice”?

Password disclosure



- What if Bob mistypes his username as “alice”?

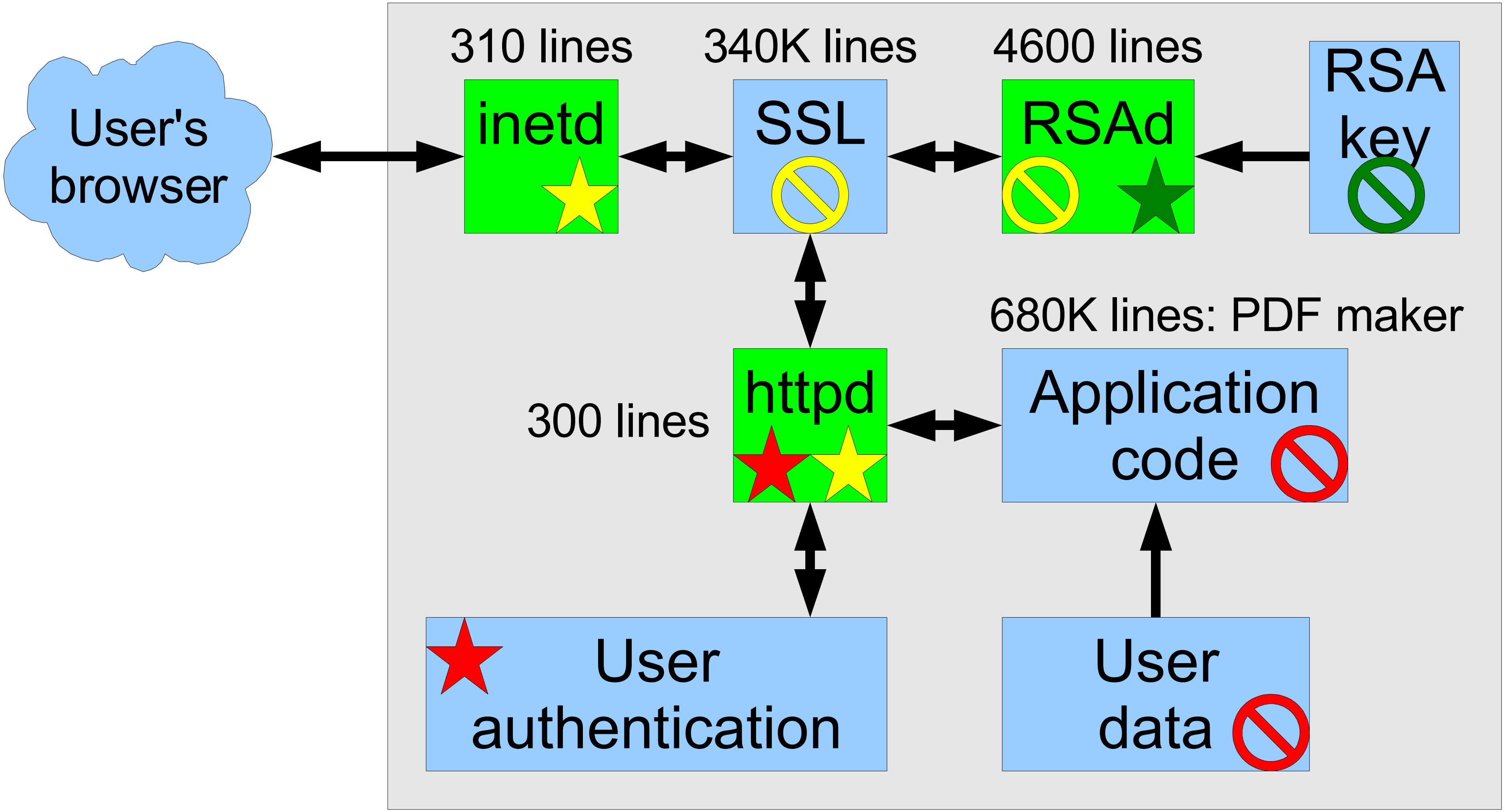
Avoiding password disclosure

- It's all about information flow
  - HiStar enforces:
  - “Password cannot go out onto the network”

- Real login uses ephemoral taint category to protect passwords

HiStar SSL Web Server

- Unlike OKWS, isolate application code per user



Reducing trusted code

- HiStar lets developers reduce trusted code
  - No code with every user's privilege during login
  - No trusted code needed to initiate authentication
  - 110-line trusted wrapper for complex virus scanner
  - Web server isolates different users' app. code

- Small kernel: <20,000 lines of code

Advertising

- Publishers get ads through Ad networks
  - E.g., AdBrite
- AdBrite gives you Javascript to generate ads

```
function print_ads () {
  for ( each ad ) {
    document.write ( text of ad );
  }
}
```

- Publisher gets paid per click on an Ad

Incentives for fraud

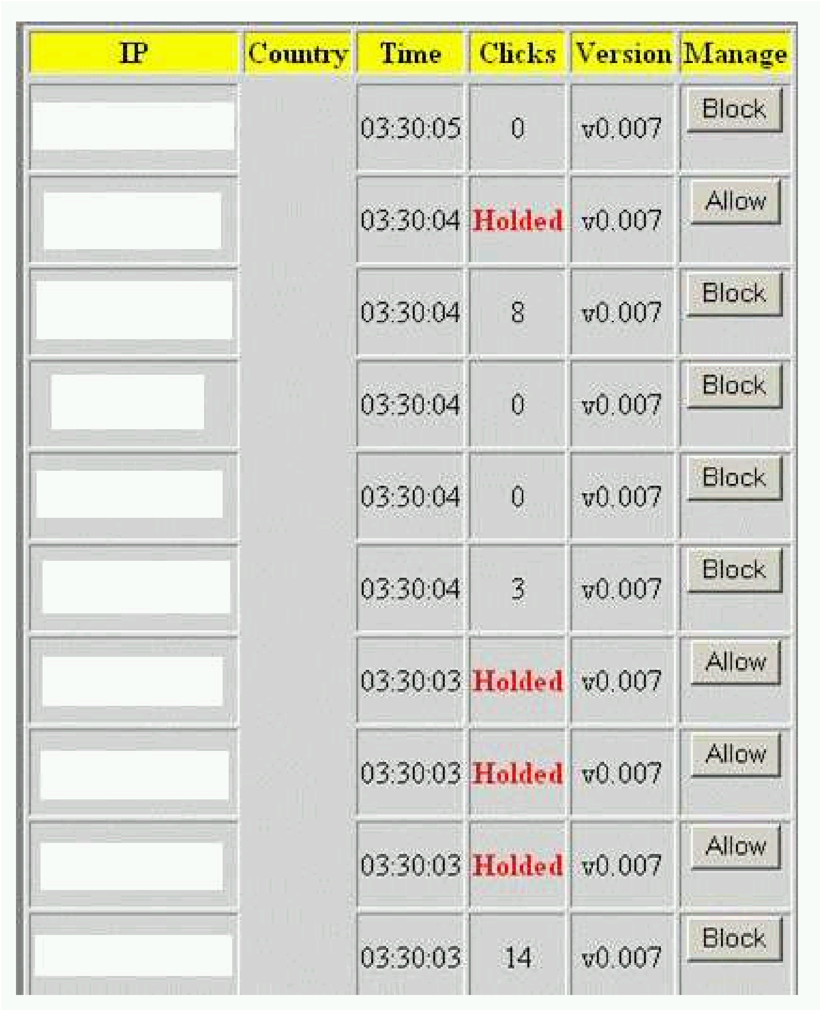
- Publishers want to inflate click counts
  - Make it look like many people clicked on ads served by their sites so as to get ad revenue
- Advertisers want to inflate competitors' counts
  - Cause lots of bogus clicks on competitors' ads
  - Maxes out competitor's ad budget
  - Ensures they only reach small audience
- Ad network profits from inflated clicks
  - But also needs to maintain perception of quality

Clickbot.A [Daswani et al.]

- Some machines infected by Trojan horse
  - Application disguised as game
  - Contacts botmaster to determine next download
  - Chain of downloads ends up with Clickbot.A
- Also probably bought existing bots
- Structured as IE browser helper object
  - Simplified parsing HTML
  - Made HTTP requests look ordinary
- Running on 100,000 machines by June 2006

Clickbot.A bot master

- Used PHP & MySQL
- Hosted by ISP with compromised accounts
- Compromised accounts also used to host “doorway” sites



IP	Country	Time	Clicks	Version	Manage
[redacted]		03:30:05	0	v0.007	Block
[redacted]		03:30:04	Holded	v0.007	Allow
[redacted]		03:30:04	8	v0.007	Block
[redacted]		03:30:04	0	v0.007	Block
[redacted]		03:30:04	0	v0.007	Block
[redacted]		03:30:04	3	v0.007	Block
[redacted]		03:30:03	Holded	v0.007	Allow
[redacted]		03:30:03	Holded	v0.007	Allow
[redacted]		03:30:03	Holded	v0.007	Allow
[redacted]		03:30:03	14	v0.007	Block

How Clickbot.A worked

- Contact botmaster to register
- Loop every 15 minutes:
  - Learn about a “doorway” site from bot master
  - Receive instructions on queries
- Bot queried doorway site based on instructions
  - Clicked through advertising
  - Used “redirector” to strip off Referer header
  - Made it harder to track bad doorway sites
- Google claims to have identified all Clickbot.A clicks by pattern and not charged for them

Badvertisements [Gandhi et al.]

- Attack identified by researchers, not yet seen
- Attacker creates two web sites:
  - `nastyporn.com` – lots of legitimate traffic, but content unacceptable to most advertisers (called the "Facade page")
  - `niceflorist.com` – site that carries advertising (called the "dual-personality page")

Generating clicks

- Facade site (nastyporn) includes “dual personality” site (niceflorist) in a tiny iframe (not visible to user)
  - Passes unique ID to niceflorist
- If niceflorist sees user ID for first time
  - Sends “badvertisement” javascript to generate clicks
- Otherwise
  - Sends innocuous javascript

Thwarting detection

- If you go back to inspect niceflowers
  - With already seen unique ID, get innocuous javascript
- Prevent crawlers from understanding nastyporn
  - Iframe is generated with javascript
  - Crawlers don't execute javascript
  - Can also use tricks to obfuscate javascript

Google AdSense not vulnerable

- Also include Javascript

```
<script type="text/javascript"
  src="http://pagead2.googlesyndication.com/pagead/show_ads.js">
</script>
```

- But ad not generated by javascript
- Instead, generates code to include Ad

```
( function () {
   function print_ads () {
     document . write ( " < iframe src = url of ad server > " );
   }
   print_ads ();
})()
```

- Inline frame generated by Google's servers
- Possibly makes adblocking easier?