Wedge: Splitting Applications into Reduced-Privilege Compartments

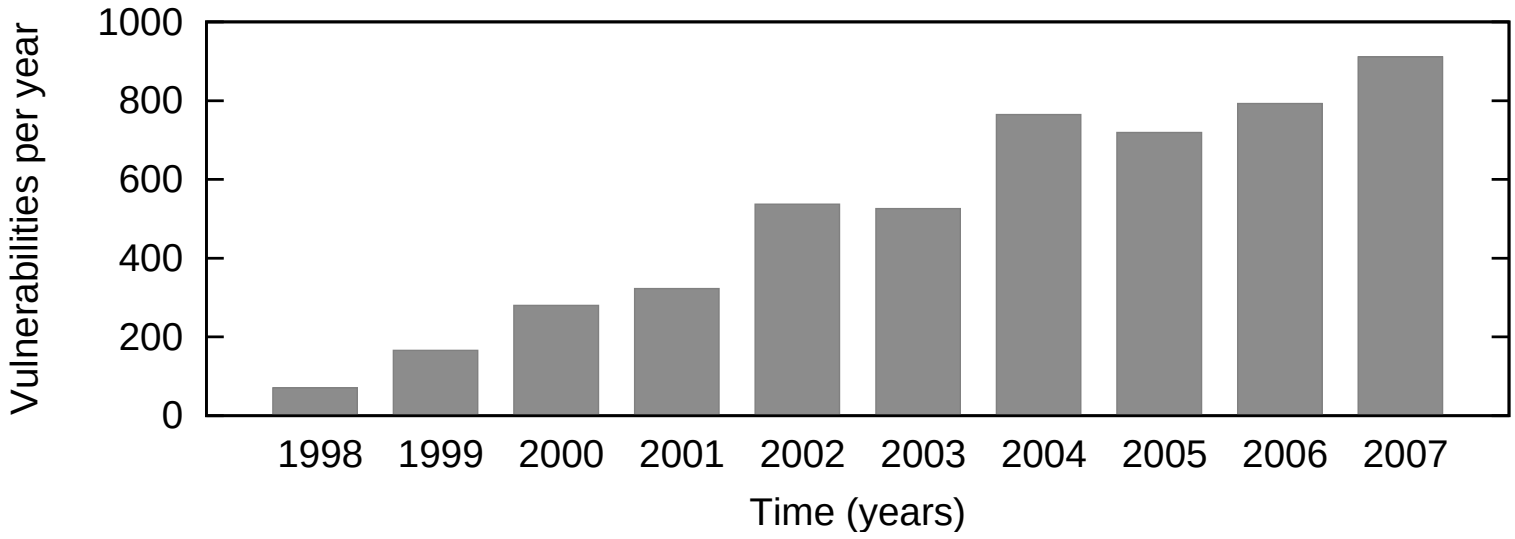
Andrea Bittau Petr Marchenko Mark Handley Brad Karp
University College London

April 17, 2008

Vulnerabilities threaten sensitive data

- Exploits allow running arbitrary code on servers.
- An exploited web server can be used to leak sensitive information such as credit card numbers.

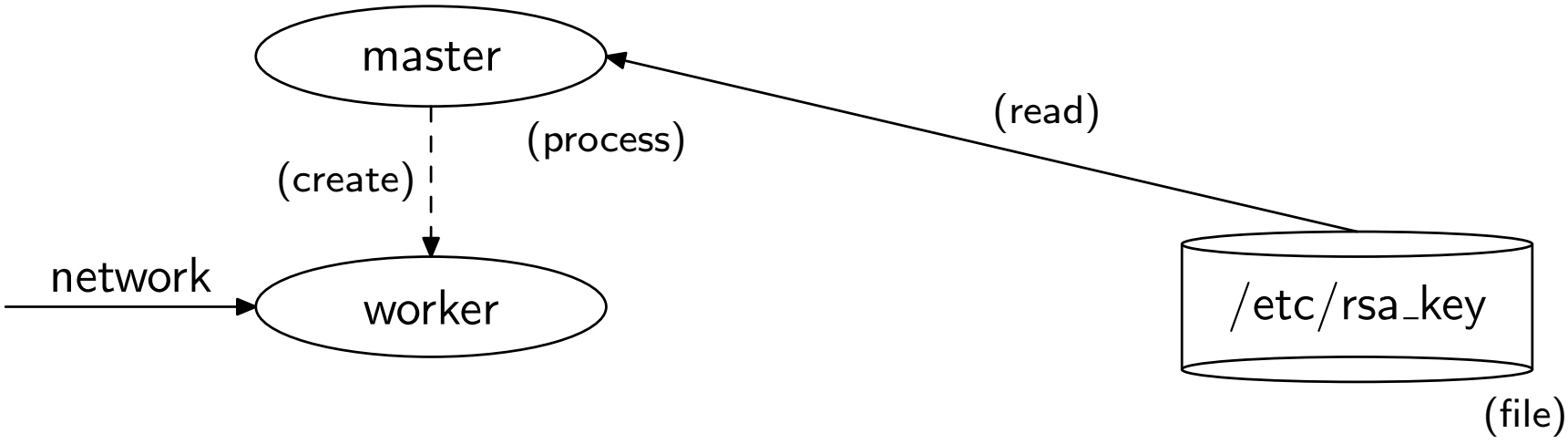
Have we managed to mitigate or prevent vulnerabilities?



Source: osvdb.org

Process-based privileges are too coarse-grained

Need to keep SSL web server's RSA private key secret.

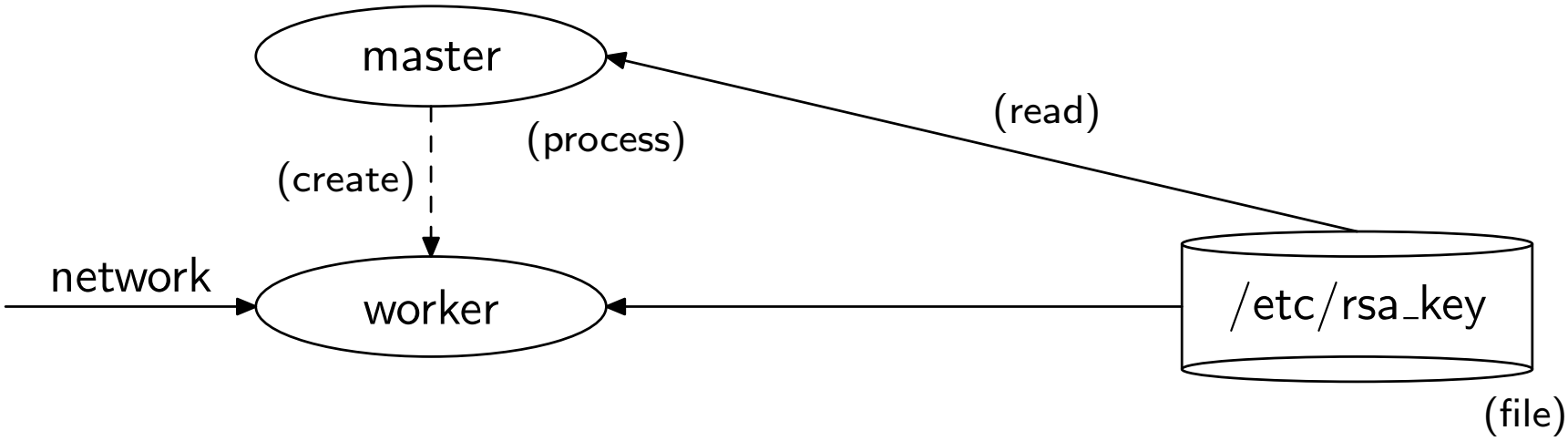


Process-based privileges are too coarse-grained

Need to keep SSL web server's RSA private key secret.

Apache worker running as root:

- Can read any file.
- Can invoke any system call.

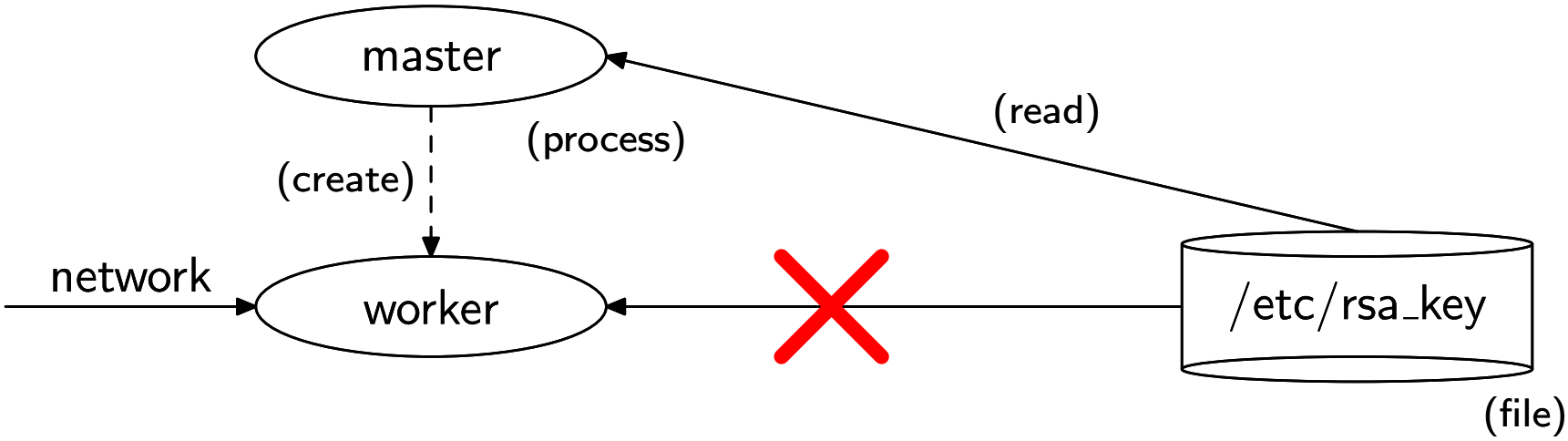


Process-based privileges are too coarse-grained

Need to keep SSL web server's RSA private key secret.

Apache worker running as root:

- Can read any file. Fix: run as nobody.
- Can invoke any system call. Fix: use systrace, SELinux, ...



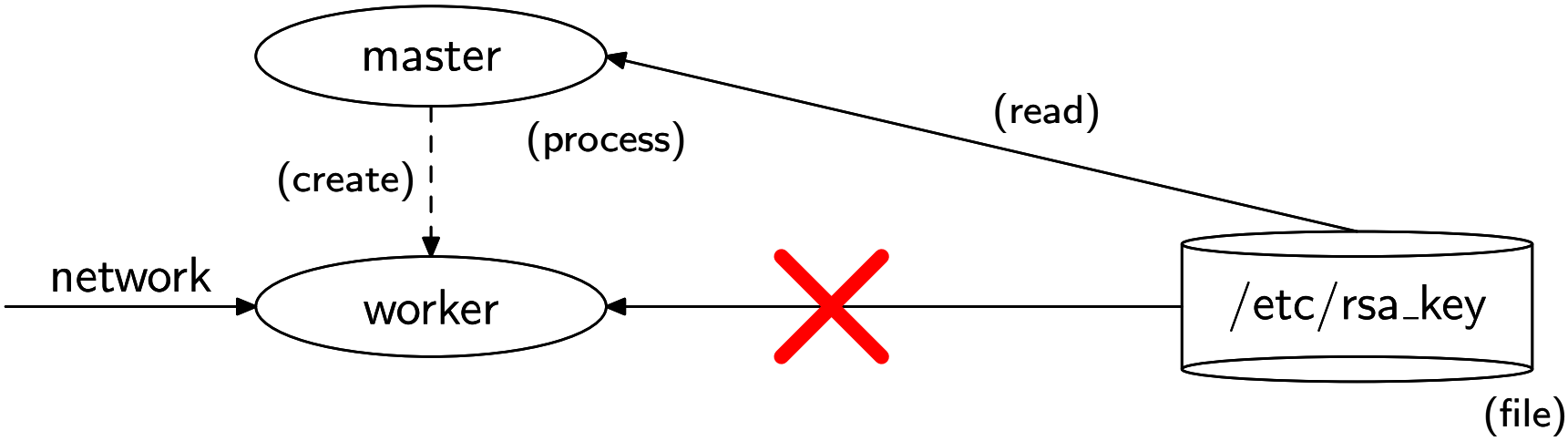
Process-based privileges are too coarse-grained

Need to keep SSL web server's RSA private key secret.

Apache worker running as root:

- Can read any file. Fix: run as nobody.
- Can invoke any system call. Fix: use systrace, SELinux, ...

Are we done protecting the private key?



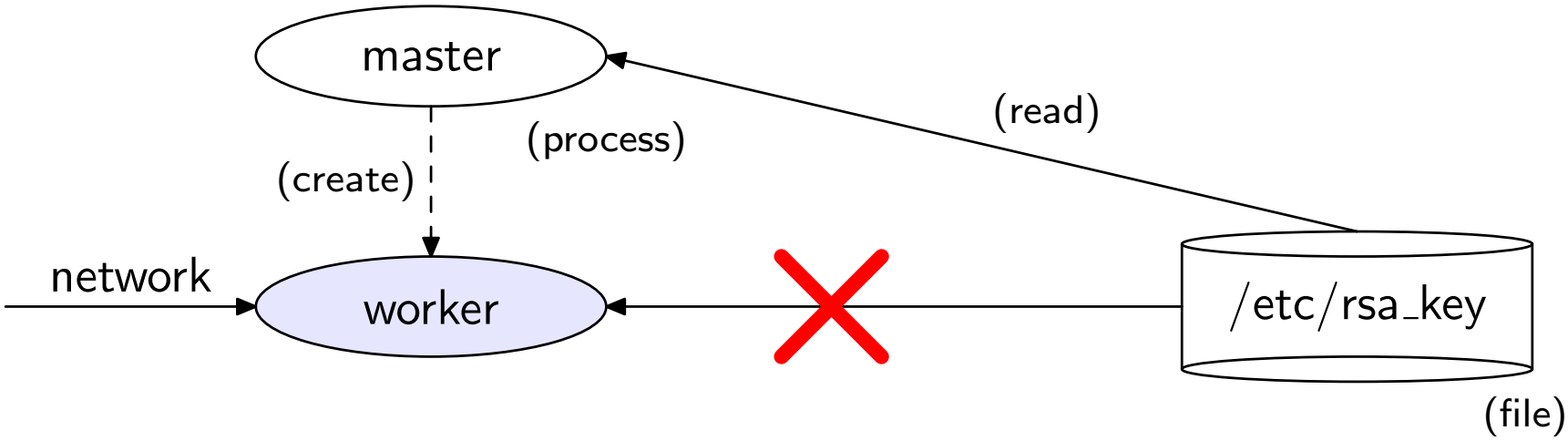
Process-based privileges are too coarse-grained

Need to keep SSL web server's RSA private key secret.

Apache worker running as root:

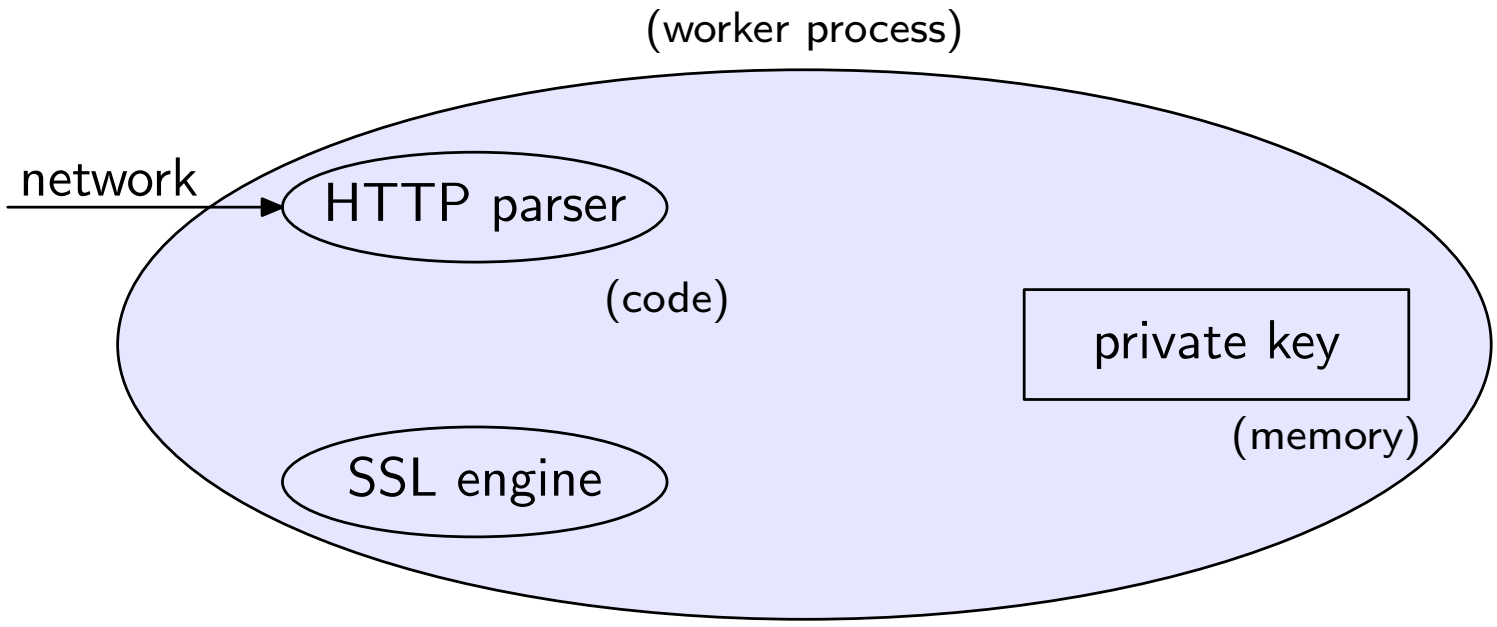
- Can read any file. Fix: run as nobody.
- Can invoke any system call. Fix: use systrace, SELinux, ...

Are we done protecting the private key?



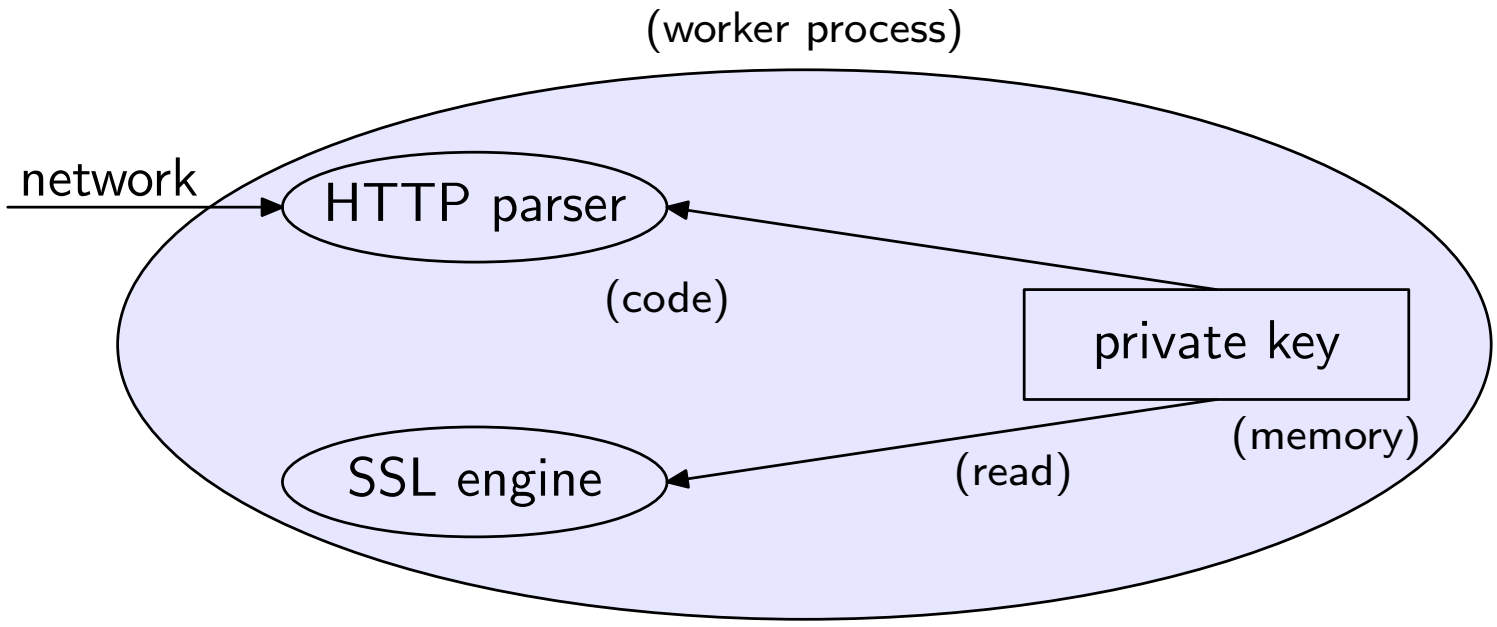
Problem: processes grant all code access to all memory

Need to keep SSL web server's RSA private key secret.



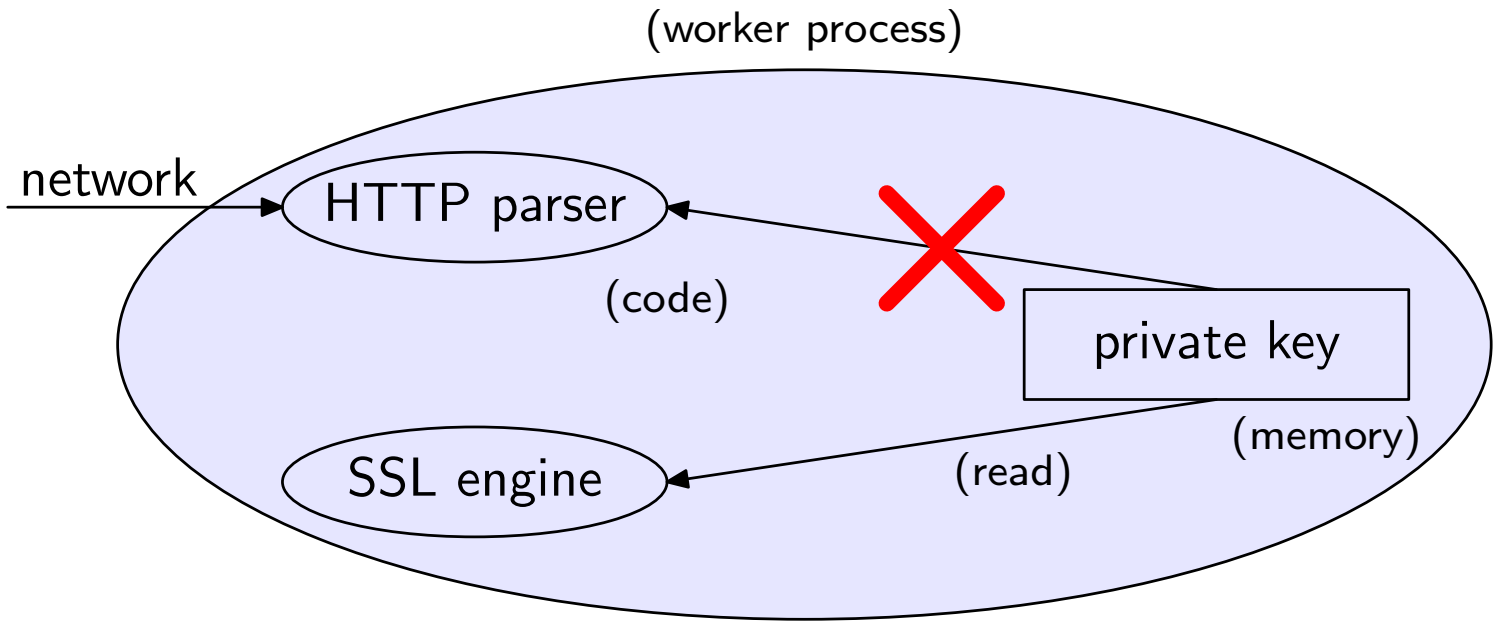
Problem: processes grant all code access to all memory

Need to keep SSL web server's RSA private key secret.



Problem: processes grant all code access to all memory

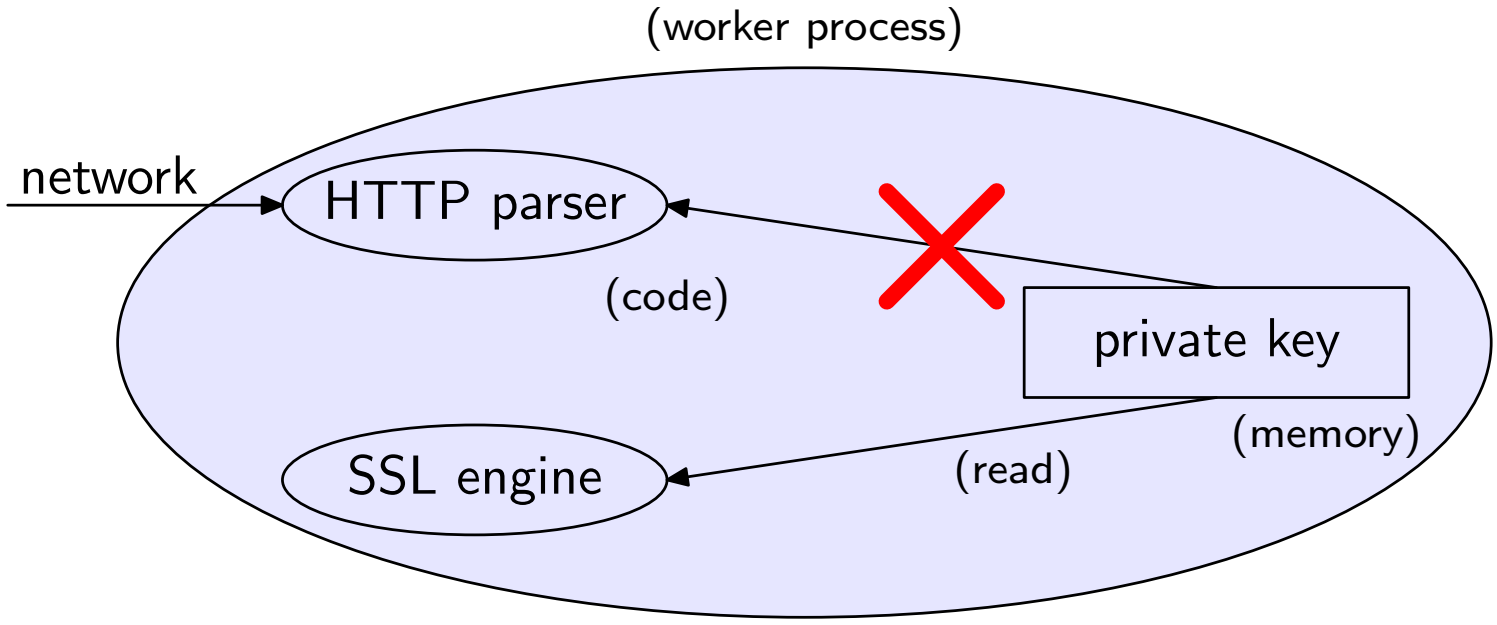
Need to keep SSL web server's RSA private key secret.



Problem: processes grant all code access to all memory

Need to keep SSL web server's RSA private key secret.

This talk: how to limit access of code to memory at fine granularity.



Old idea: principle of least privilege

Principle of least privilege:

- Partition code into *compartments*.
- Assign each compartment the minimal privileges it needs for its operation.
- Restrict interface and interactions between compartments.

How to implement compartments?

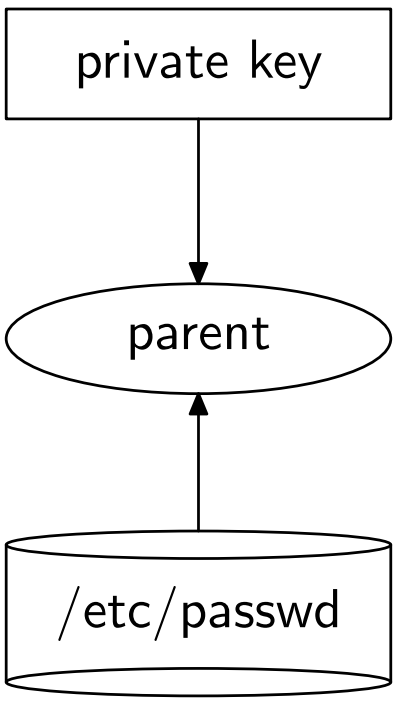
- Processes?

Why are traditional processes not sufficient?

Creating compartments with UNIX, *e.g.*, `fork`:

- Default grant. Child inherits memory map and file descriptors.

Operation of fork

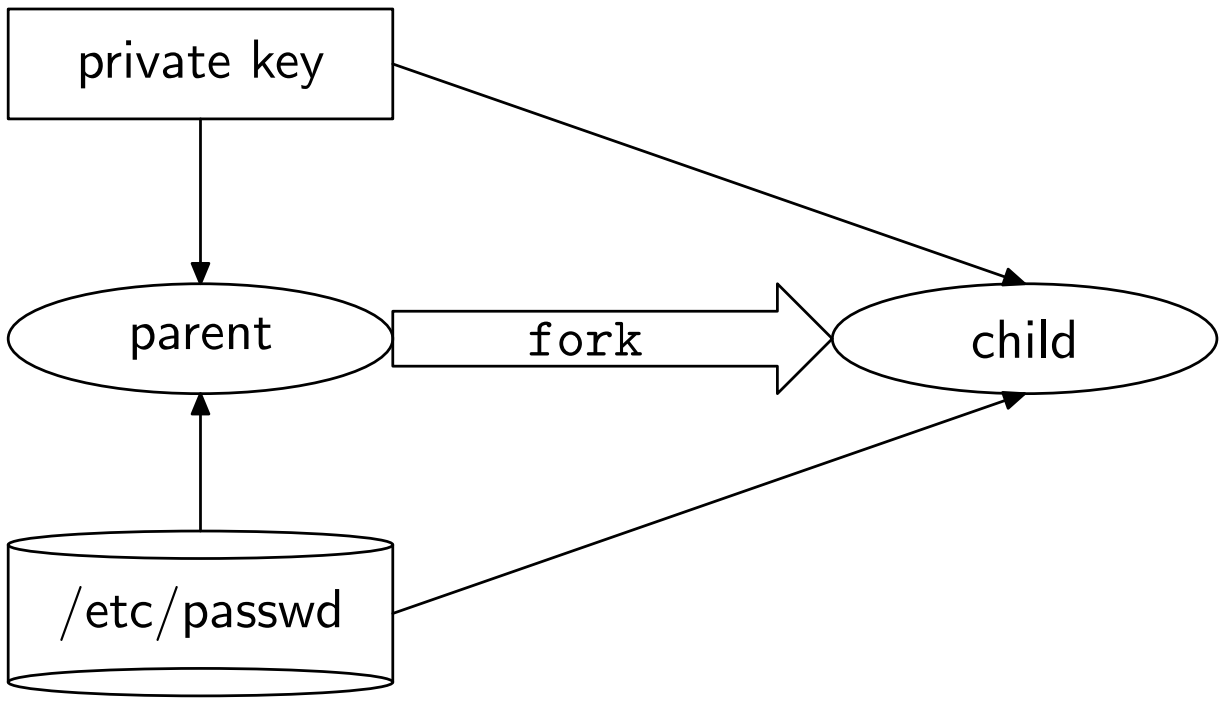


Why are traditional processes not sufficient?

Creating compartments with UNIX, *e.g.*, `fork`:

- Default grant. Child inherits memory map and file descriptors.

Operation of fork

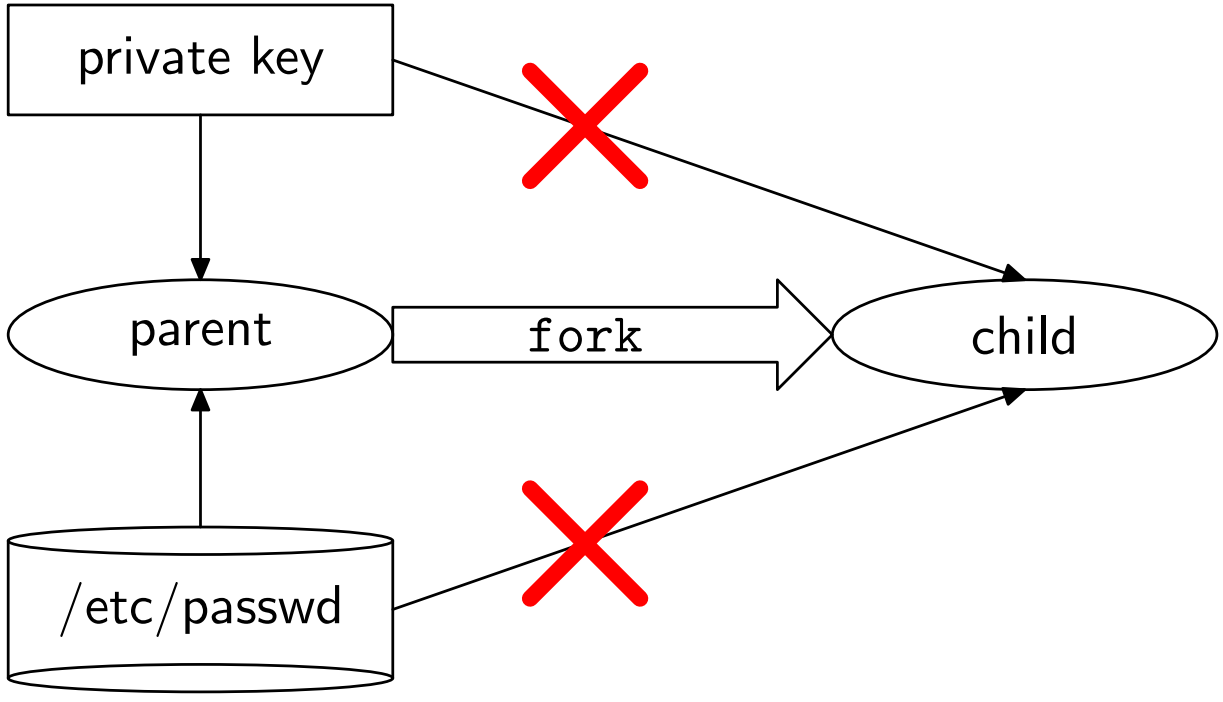


Why are traditional processes not sufficient?

Creating compartments with UNIX, *e.g.*, `fork`:

- Default grant. Child inherits memory map and file descriptors.

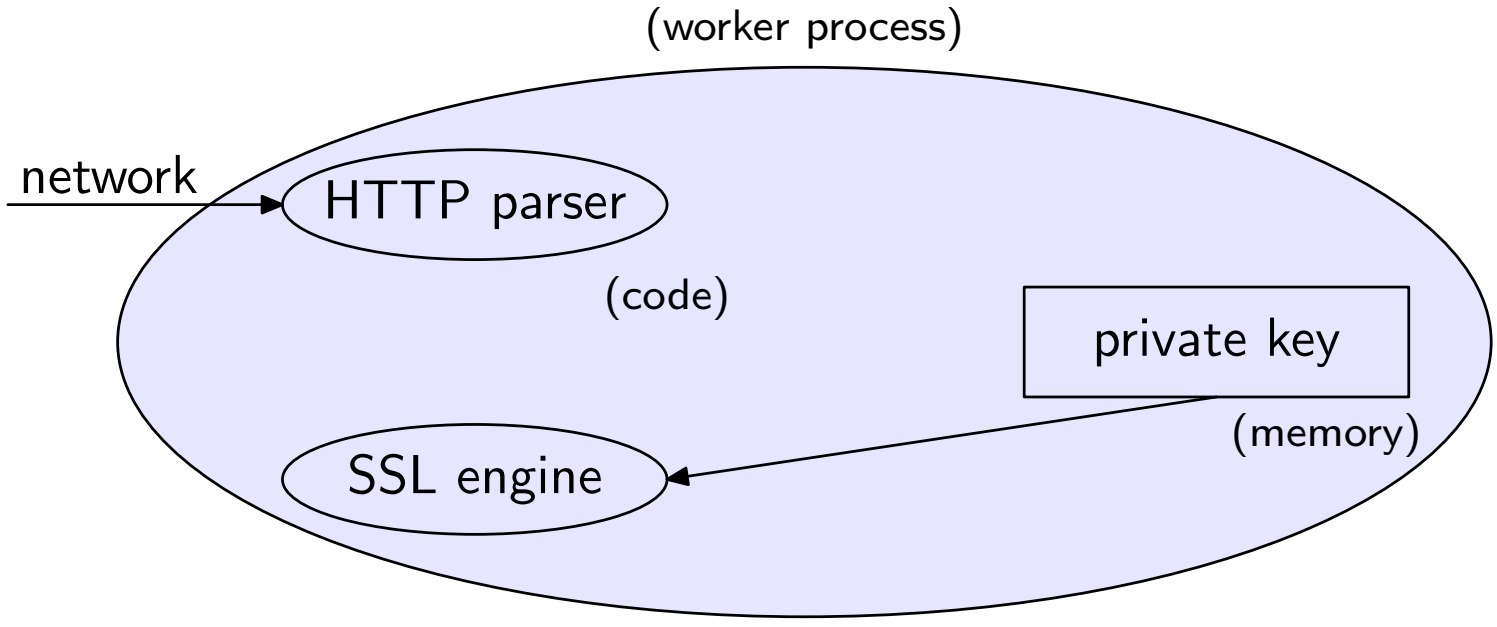
Operation of fork



Default-deny: inherit nothing from parent. Closer to least-privilege.

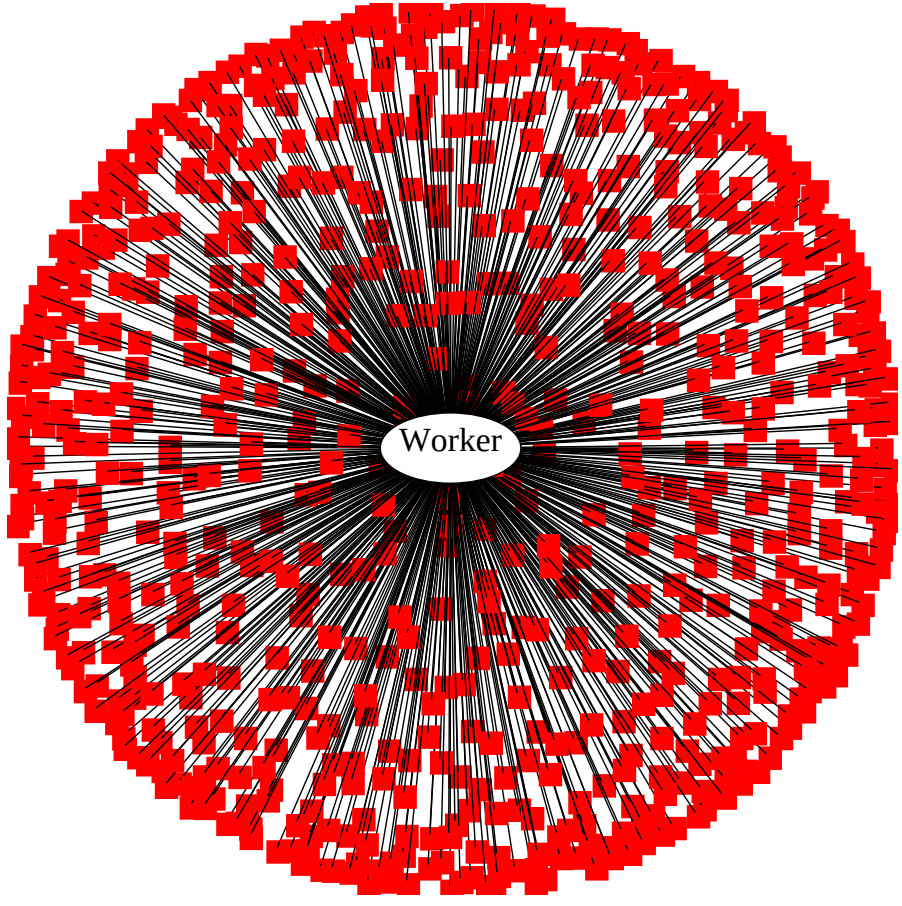
But default-deny is difficult to use for legacy code

How many permissions do we need to explicitly grant?



But default-deny is difficult to use for legacy code

How many permissions do we need to explicitly grant?



Apache's client handler uses over 600 memory objects.

Contributions

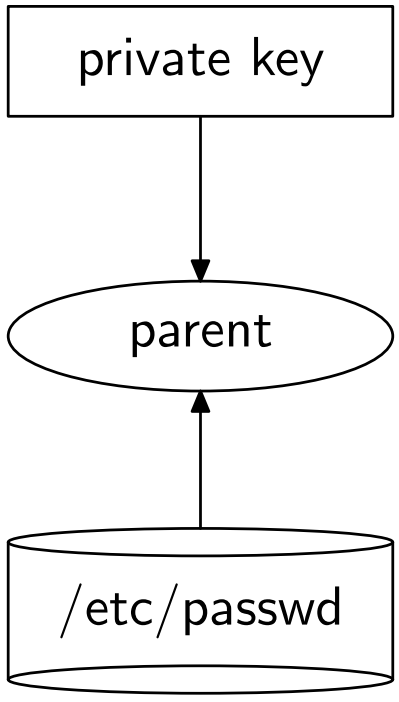
- New system calls for default-deny.
  - Creating compartments.
  - Specifying privileges.
- Tools to make default-deny usable when partitioning legacy code.
  - Identifying the privileges for compartments.

Outline

1. Wedge.
   - New system calls for default-deny.
   - Crowbar: tool for partitioning legacy code.

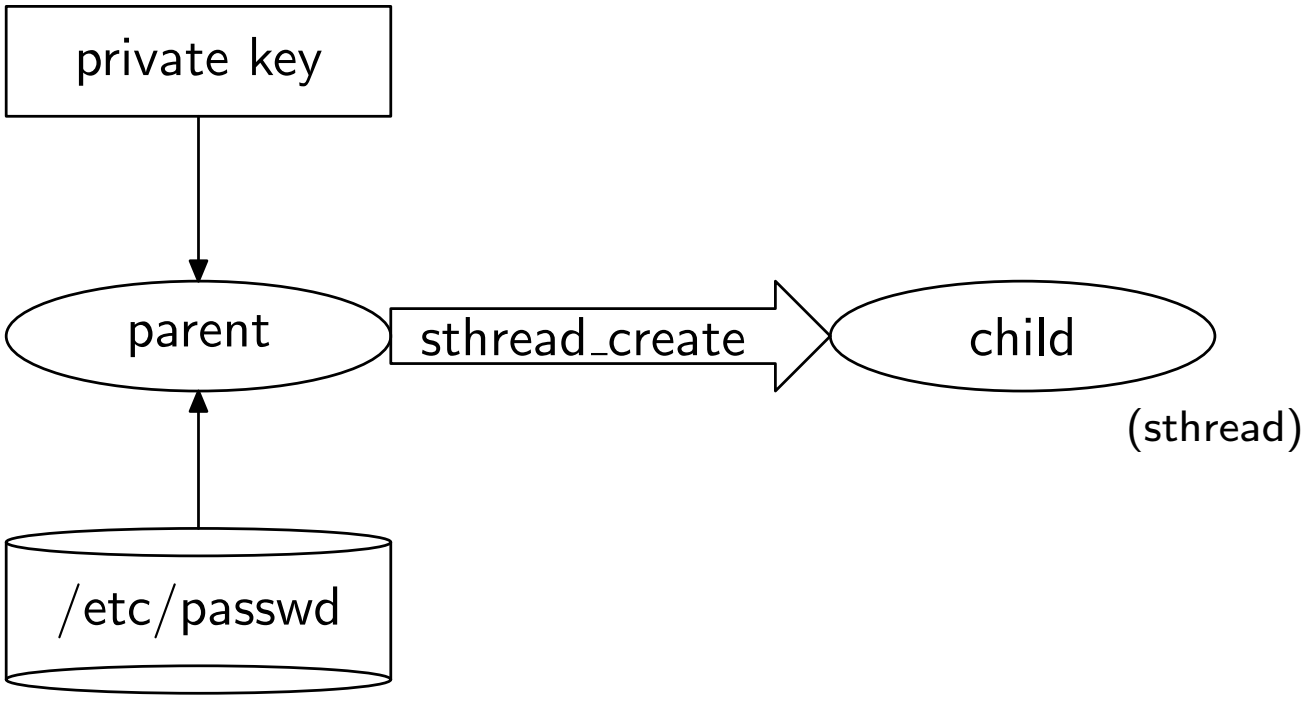
2. Wedge applied to Apache+OpenSSL.

sthreads: default-deny compartments



- Like processes, but default-deny.
- Like threads: can easily share pointers and file descriptors.
- Programmer must explicitly grant all permissions.

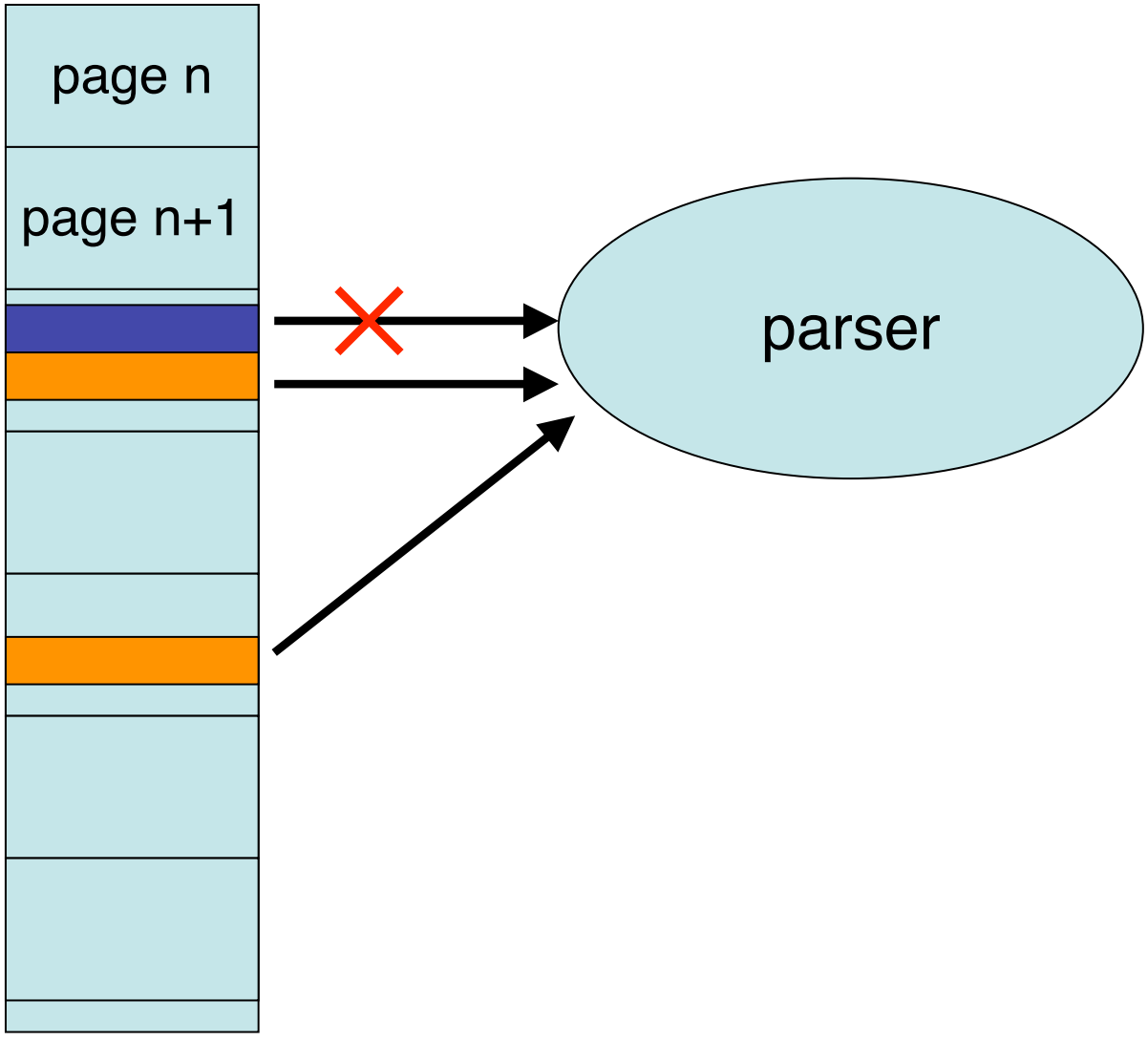
sthreads: default-deny compartments



- Like processes, but default-deny.
- Like threads: can easily share pointers and file descriptors.
- Programmer must explicitly grant all permissions.

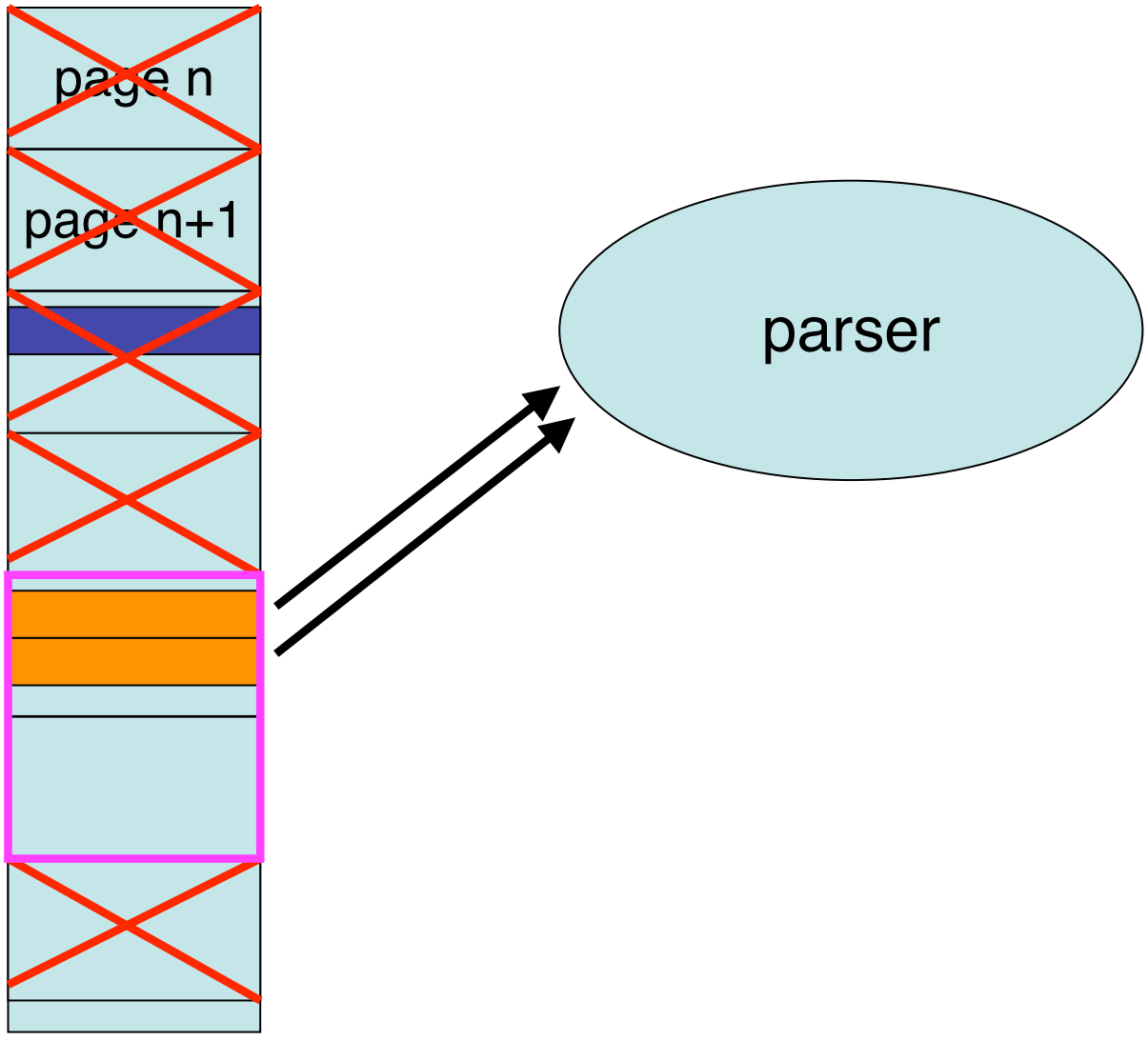
Virtual memory

```
{
    char *key, *buffer;
    char *config;
    key = malloc(16);
    buffer = malloc(80);
    ...

    config = malloc(128);
}
```



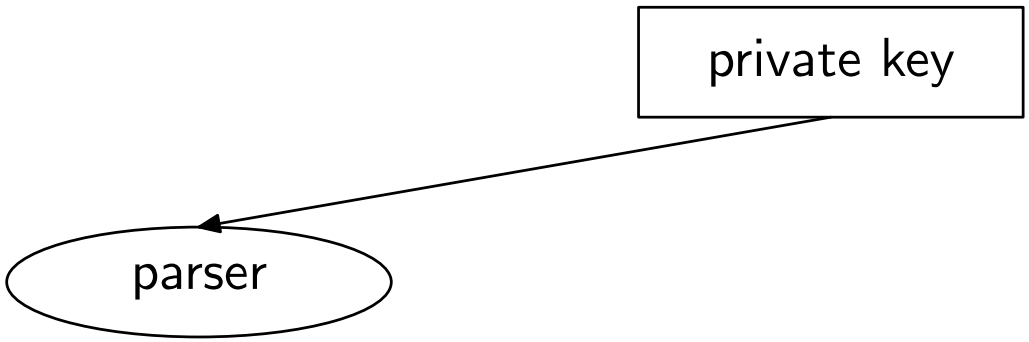
Tagged memory

```
{
    tag = tag_new();
    key = malloc(16);
    buffer =
       smalloc(80,tag);
    …

    config =
       smalloc(128,tag);
}
```



How can sthreads use sensitive data? Callgates.

Problem: unprivileged code cannot access sensitive data directly but must still use it.

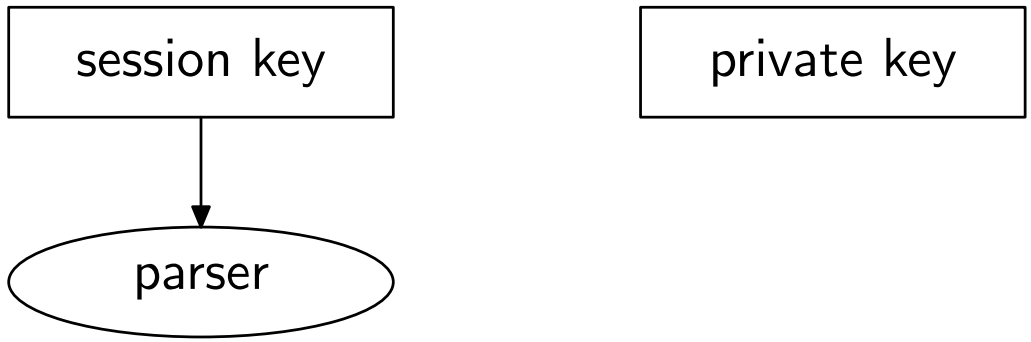


Callgates: an entry-point with predefined privileges.

- Callgates are created and invoked at a later time.
- At creation, a subset of creator's privileges is given to callgate.
- At invocation, code is run with creation privileges.

How can sthreads use sensitive data? Callgates.

Problem: unprivileged code cannot access sensitive data directly but must still use it.

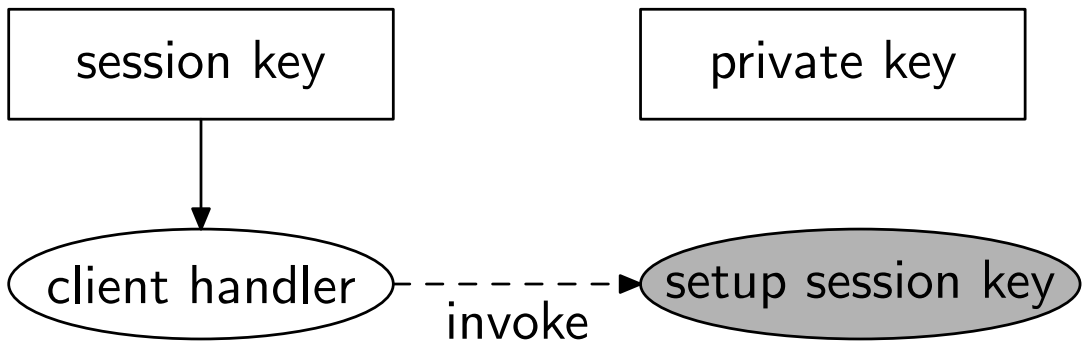


Callgates: an entry-point with predefined privileges.

- Callgates are created and invoked at a later time.
- At creation, a subset of creator's privileges is given to callgate.
- At invocation, code is run with creation privileges.

How can sthreads use sensitive data? Callgates.

Problem: unprivileged code cannot access sensitive data directly but must still use it.

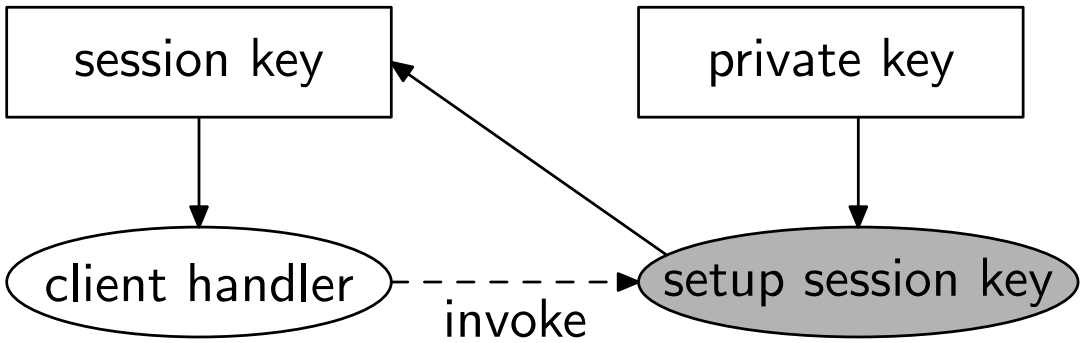


Callgates: an entry-point with predefined privileges.

- Callgates are created and invoked at a later time.
- At creation, a subset of creator's privileges is given to callgate.
- At invocation, code is run with creation privileges.

How can sthreads use sensitive data? Callgates.

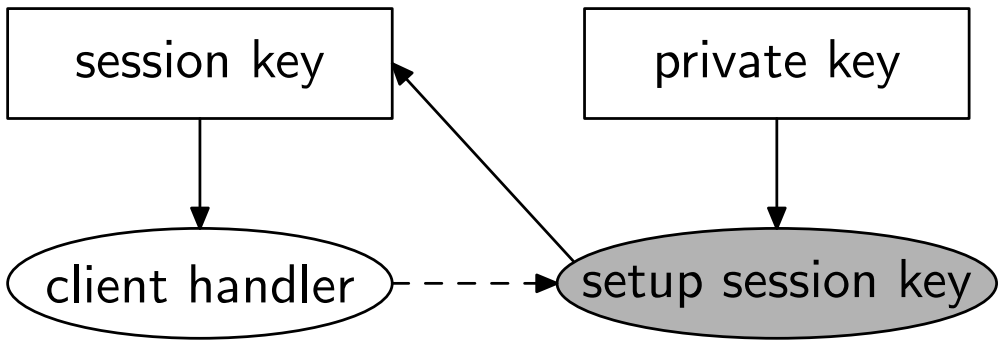
Problem: unprivileged code cannot access sensitive data directly but must still use it.



Callgates: an entry-point with predefined privileges.

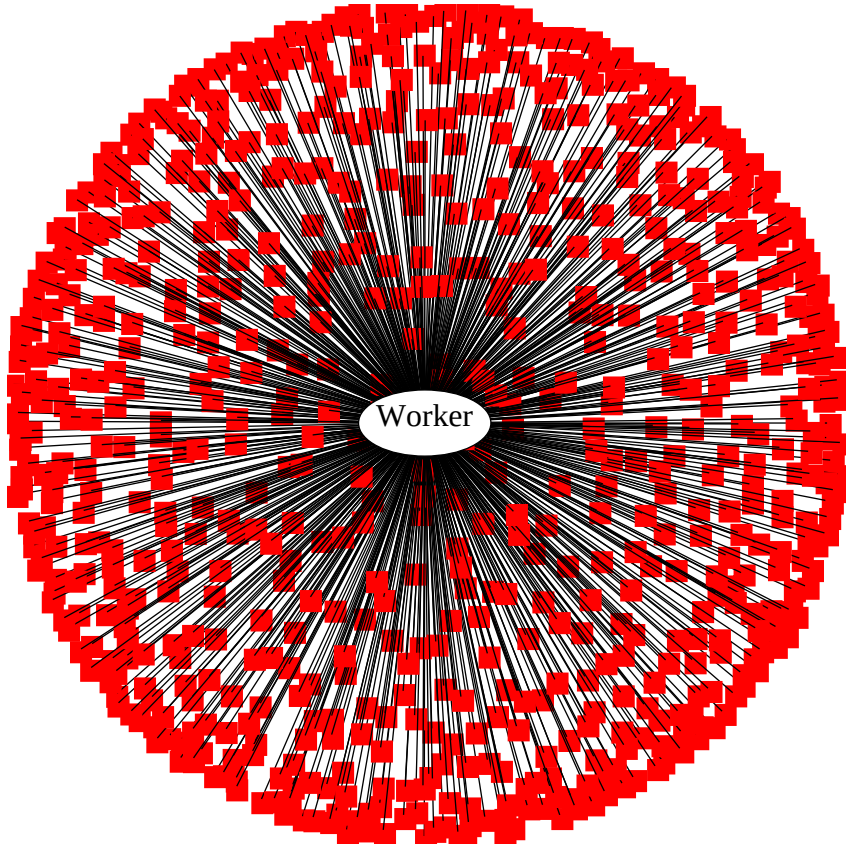
- Callgates are created and invoked at a later time.
- At creation, a subset of creator's privileges is given to callgate.
- At invocation, code is run with creation privileges.

Summary: Wedge applied to Apache



- Sthreads: default-deny compartments—low privilege.
- Callgates: privilege elevation—high privilege.
- Tagged memory: naming memory for privilege specification.

Ad-hoc code study?



Apache's client handler needs access to 222 heap objects and 389 globals. Need to read 72 source files (for heap only).

1. Which code is executed?
2. What objects do pointers point to?
3. Where were objects allocated?

Static analysis of memory accesses?

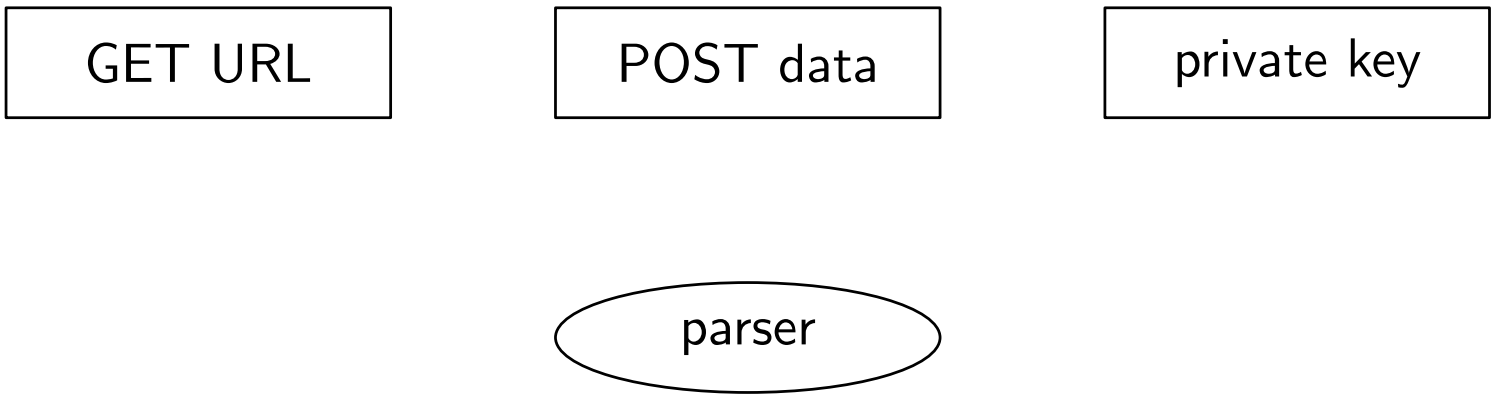
Static analysis for C code does not have runtime context (*e.g.*, format string for `printf`).

Consequences:

- May fail. *e.g.*, function pointers.
- If conservative, may give superset of privileges actually needed. *e.g.*, may follow code paths corresponding to exploits!

Crowbar: runtime analysis of memory accesses

Dynamic analysis yields least privilege:



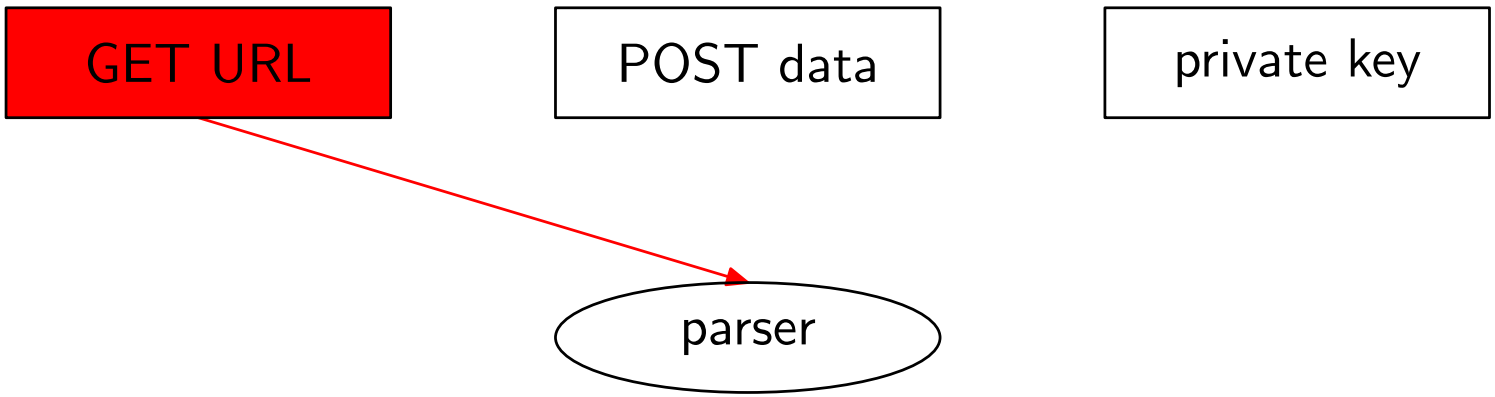
Server uses minimal privileges to execute an innocuous request.

1. Use runtime instrumentation to produce memory trace.
2. Train using benign requests.

Need to ensure high trace coverage, *e.g.*, with test suite.

Crowbar: runtime analysis of memory accesses

Dynamic analysis yields least privilege:



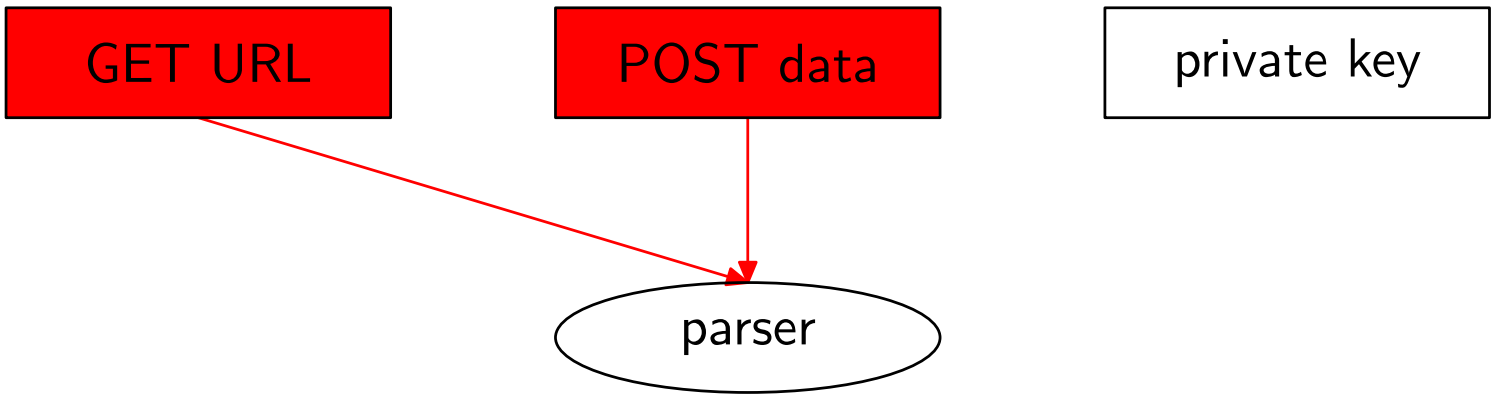
Server uses minimal privileges to execute an innocuous request.

1. Use runtime instrumentation to produce memory trace.
2. Train using benign requests.

Need to ensure high trace coverage, *e.g.*, with test suite.

Crowbar: runtime analysis of memory accesses

Dynamic analysis yields least privilege:



Server uses minimal privileges to execute an innocuous request.

1. Use runtime instrumentation to produce memory trace.
2. Train using benign requests.

Need to ensure high trace coverage, *e.g.*, with test suite.

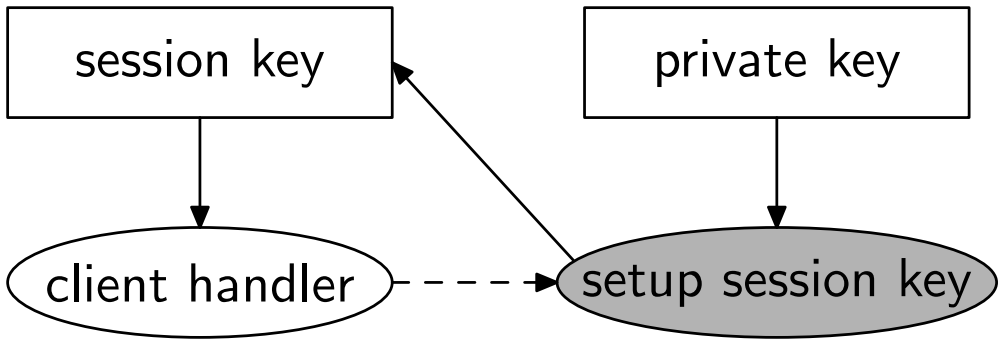
Outline

1. Wedge.
   - New system calls for default-deny.
   - Crowbar: tool for partitioning legacy code.

2. Wedge applied to Apache+OpenSSL.

Protecting keys and sensitive user data

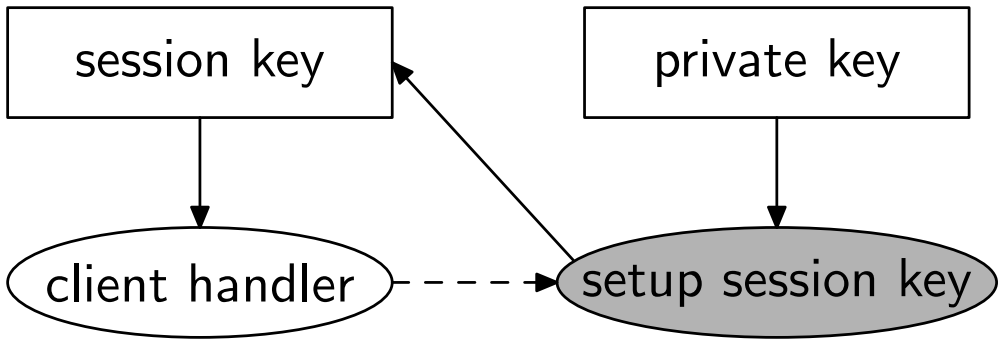
Goal: protect sensitive data (*e.g.*, credit card).



Have we protected sensitive data? Are we done?

Protecting keys and sensitive user data

Goal: protect sensitive data (*e.g.*, credit card).



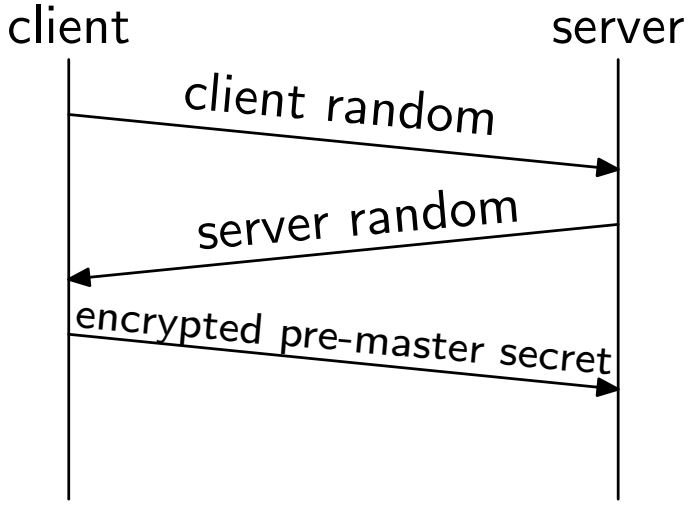
Have we protected sensitive data? Are we done?

Threat models, with increasing complexity:

1. Passive eavesdropping and server exploit.
2. Active man-in-the-middle and server exploit.

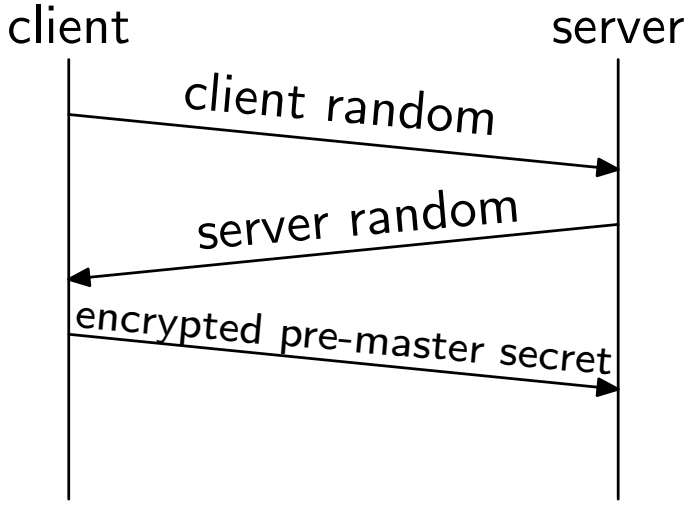
Attacker can generate arbitrary session key

Session key components exchanged during SSL handshake



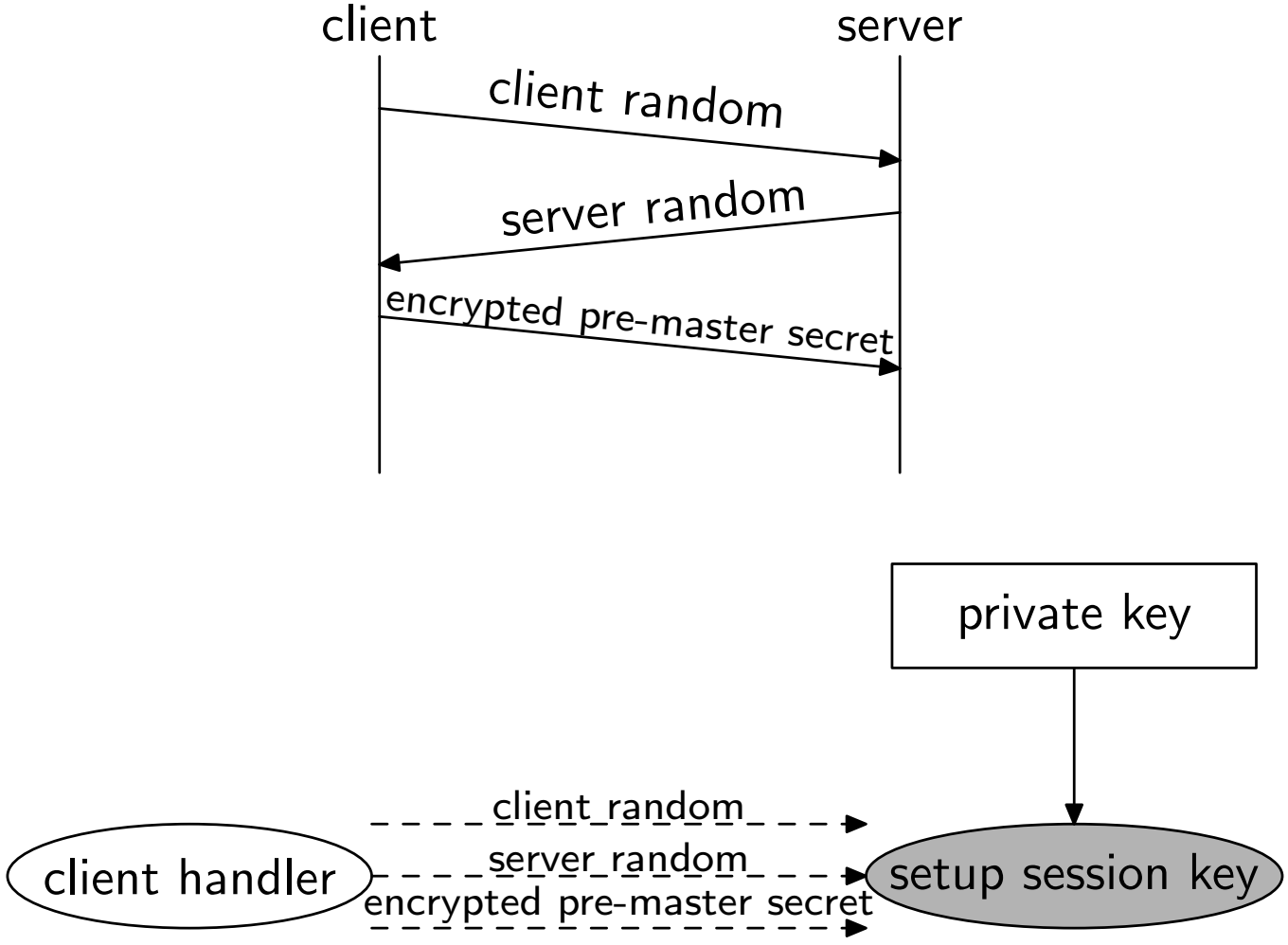
Attacker can generate arbitrary session key

Session key components exchanged during SSL handshake



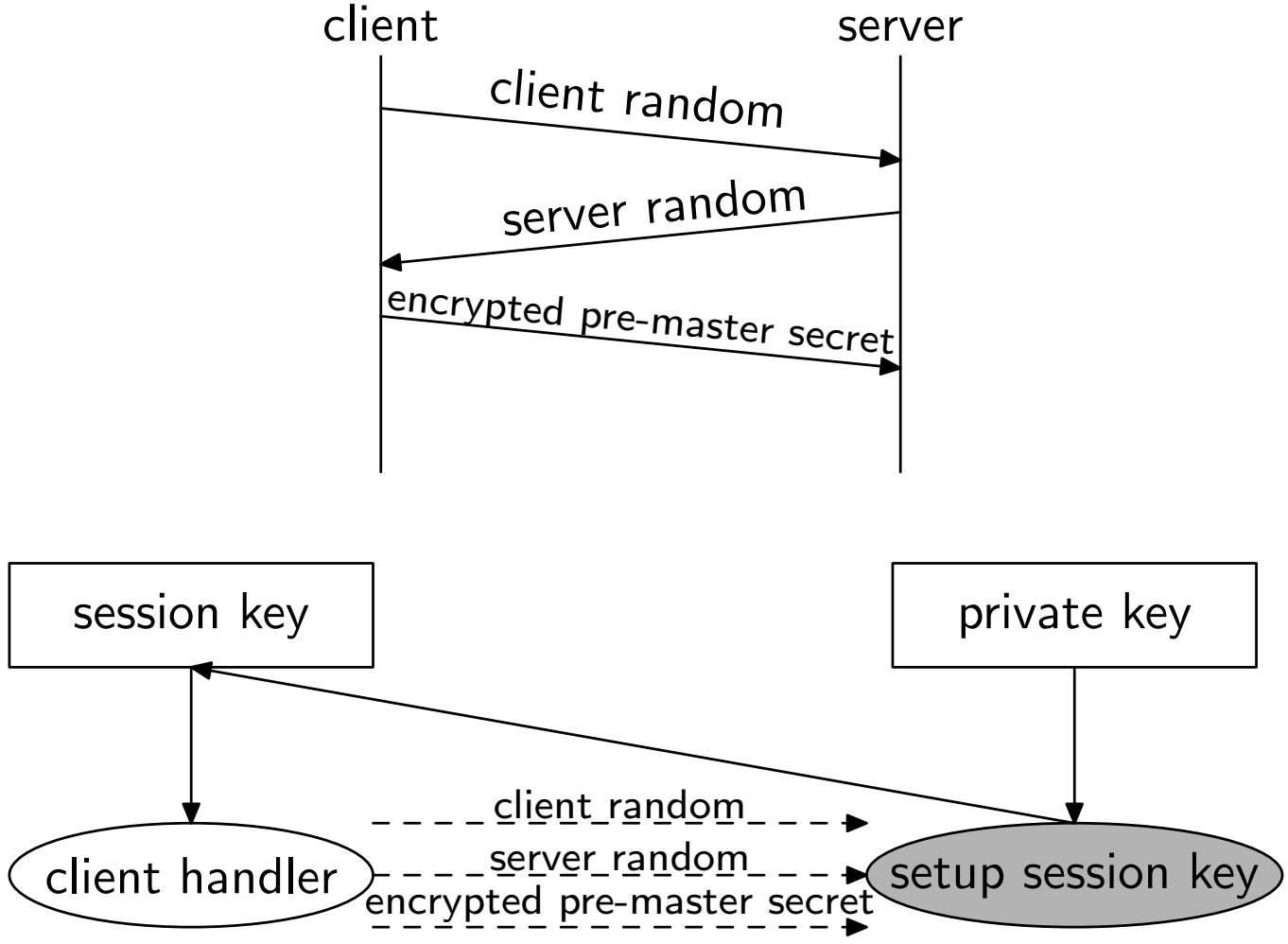
Attacker can generate arbitrary session key

Session key components exchanged during SSL handshake



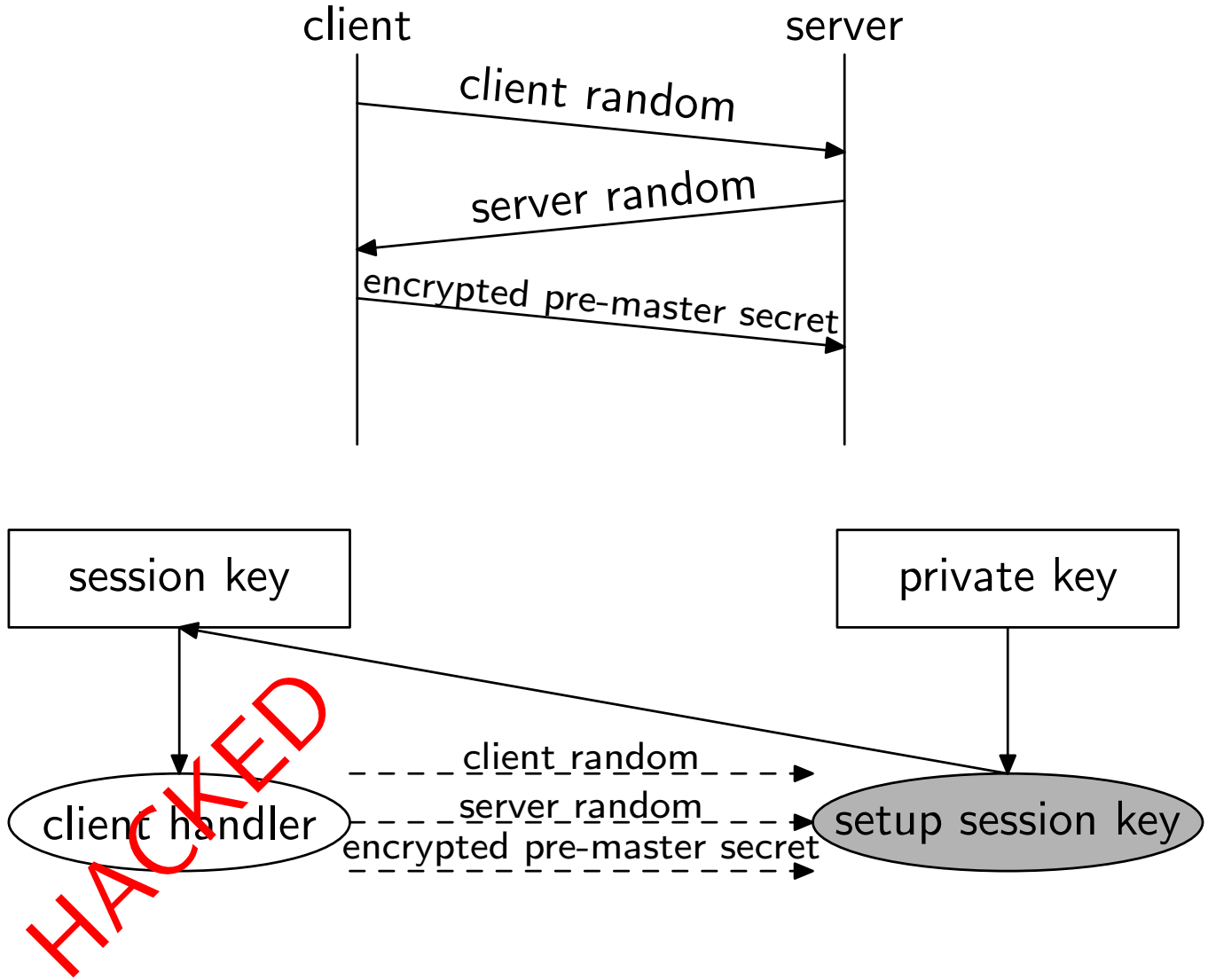
Attacker can generate arbitrary session key

Session key components exchanged during SSL handshake



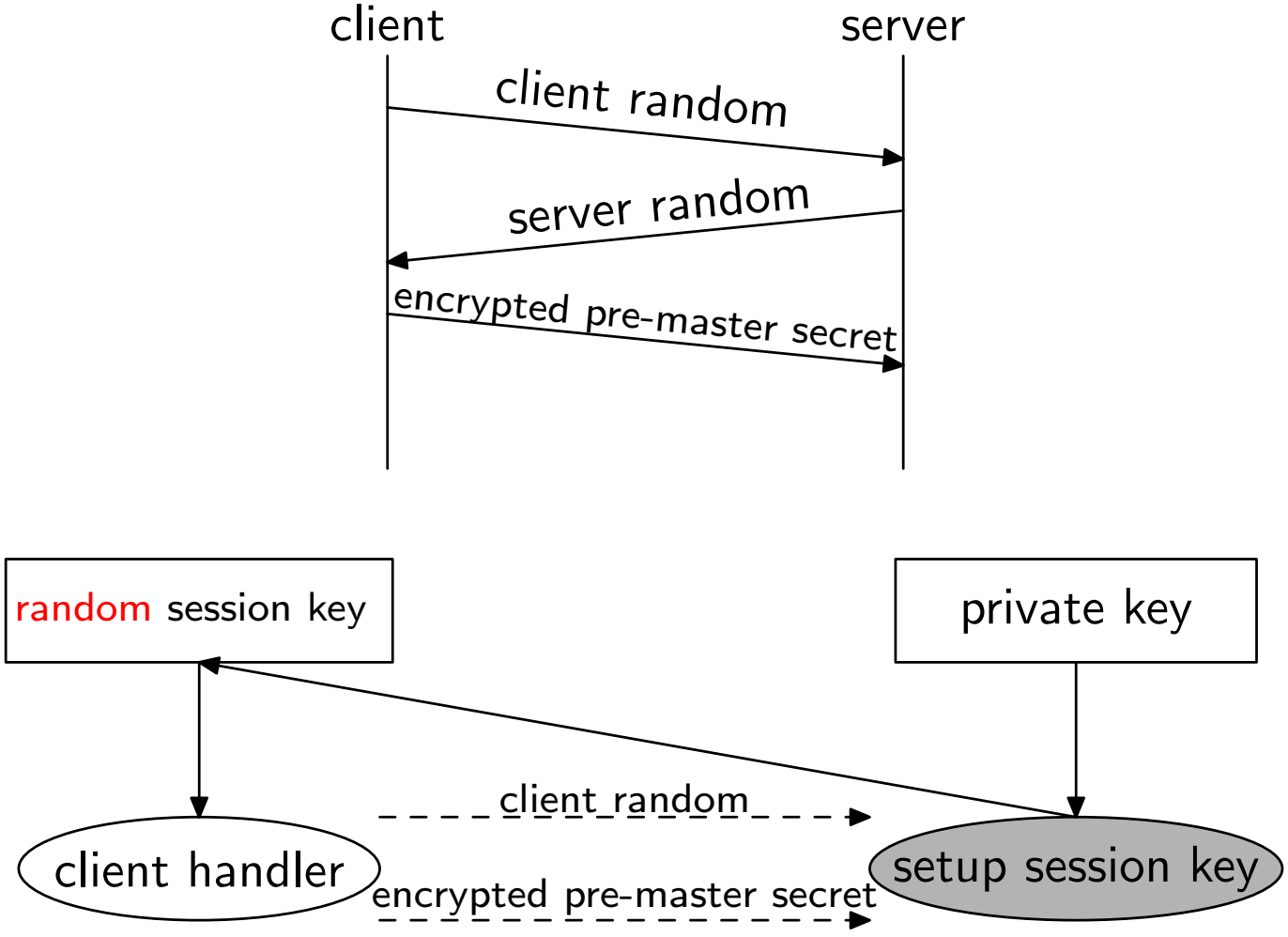
Attacker can generate arbitrary session key

Session key components exchanged during SSL handshake

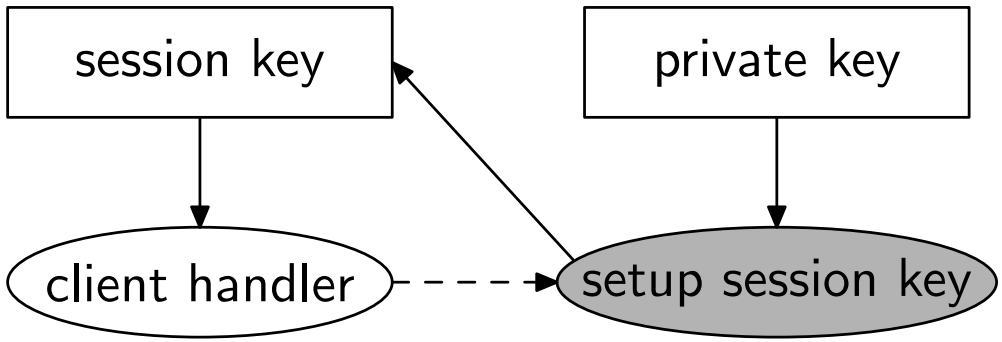


Attacker can generate arbitrary session key

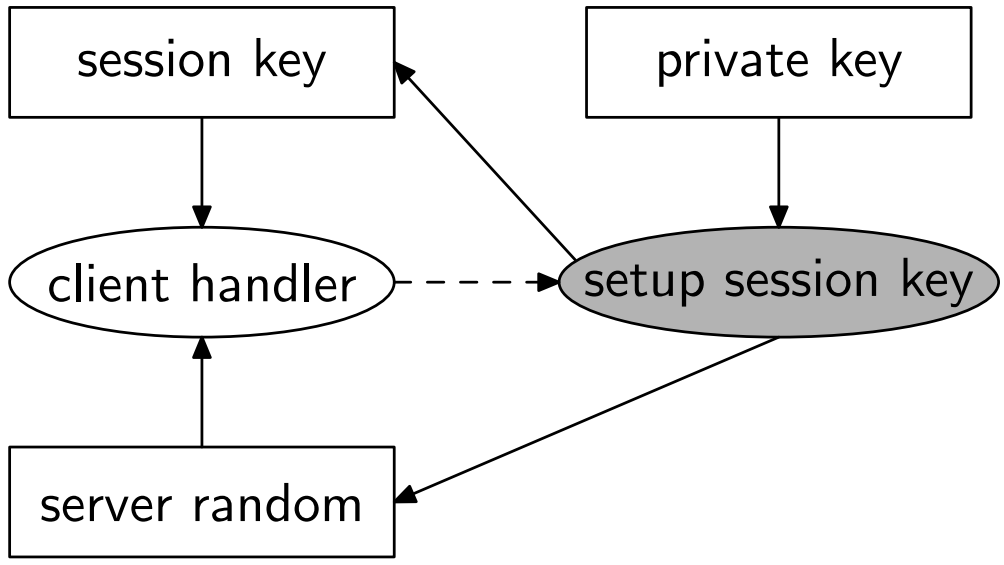
Session key components exchanged during SSL handshake



Preventing arbitrary session key leak



Preventing arbitrary session key leak

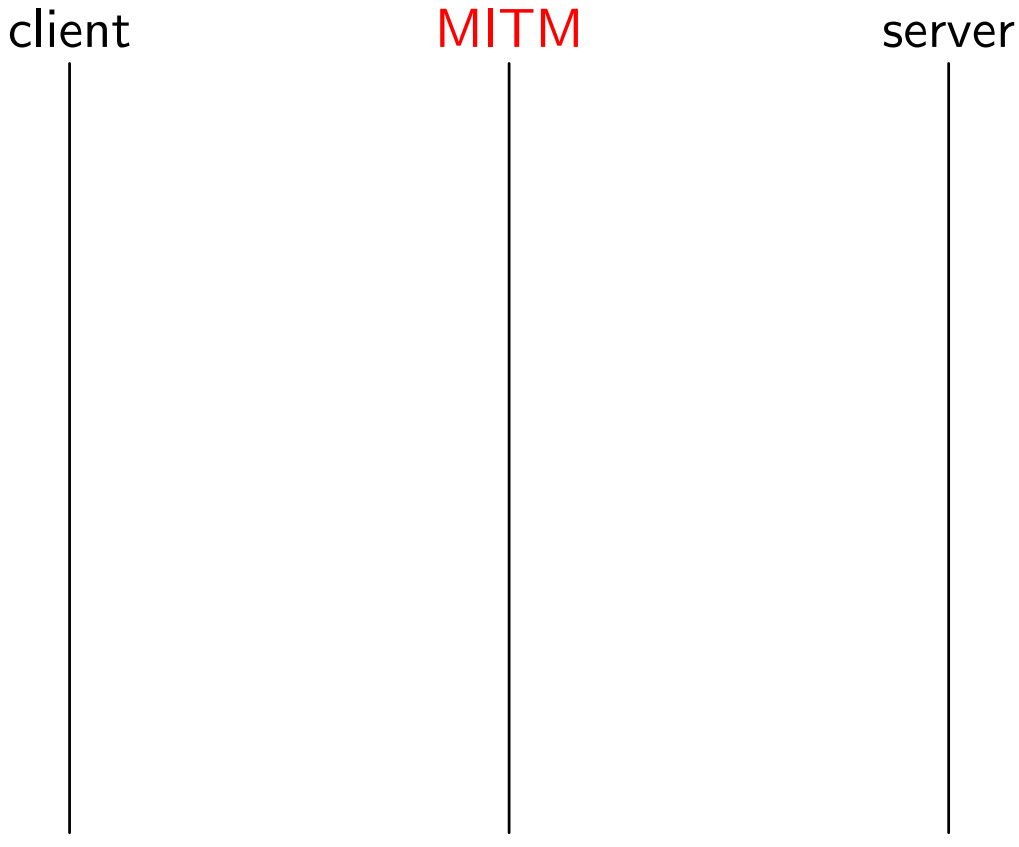


Attacker exploiting client handler:

- Has no control over server random and session key generation.
- Cannot generate session key of eavesdropped sessions.
- Can only obtain a new, personal session key.

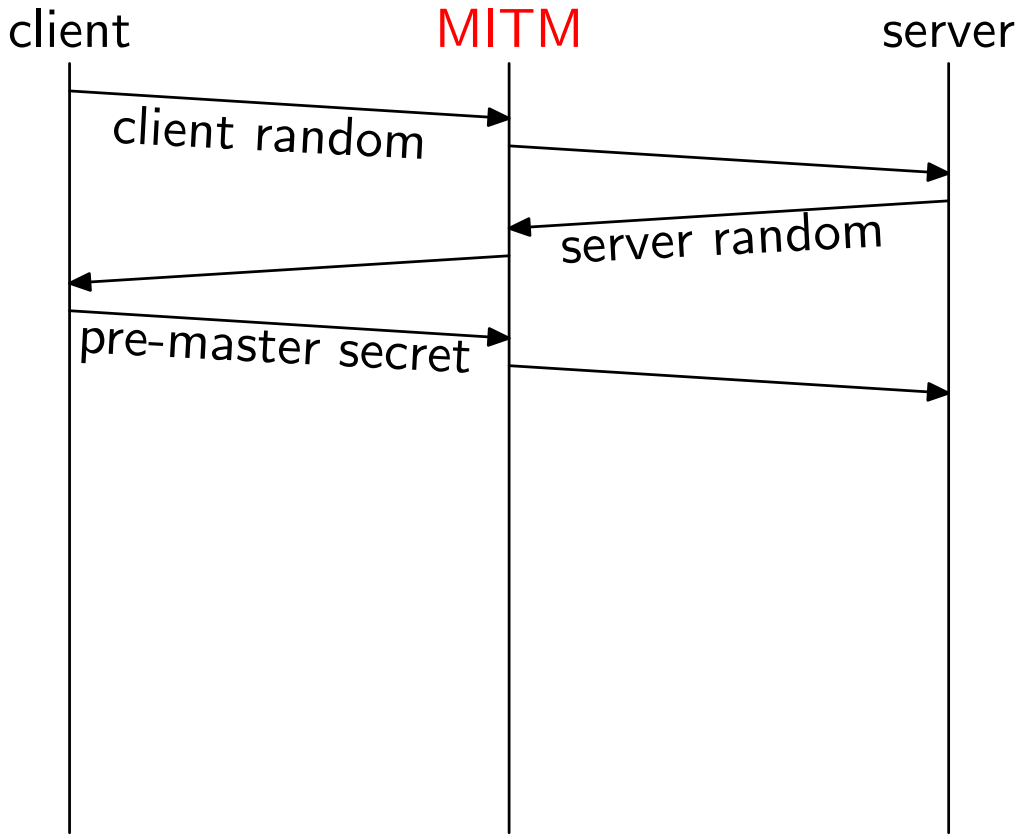
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



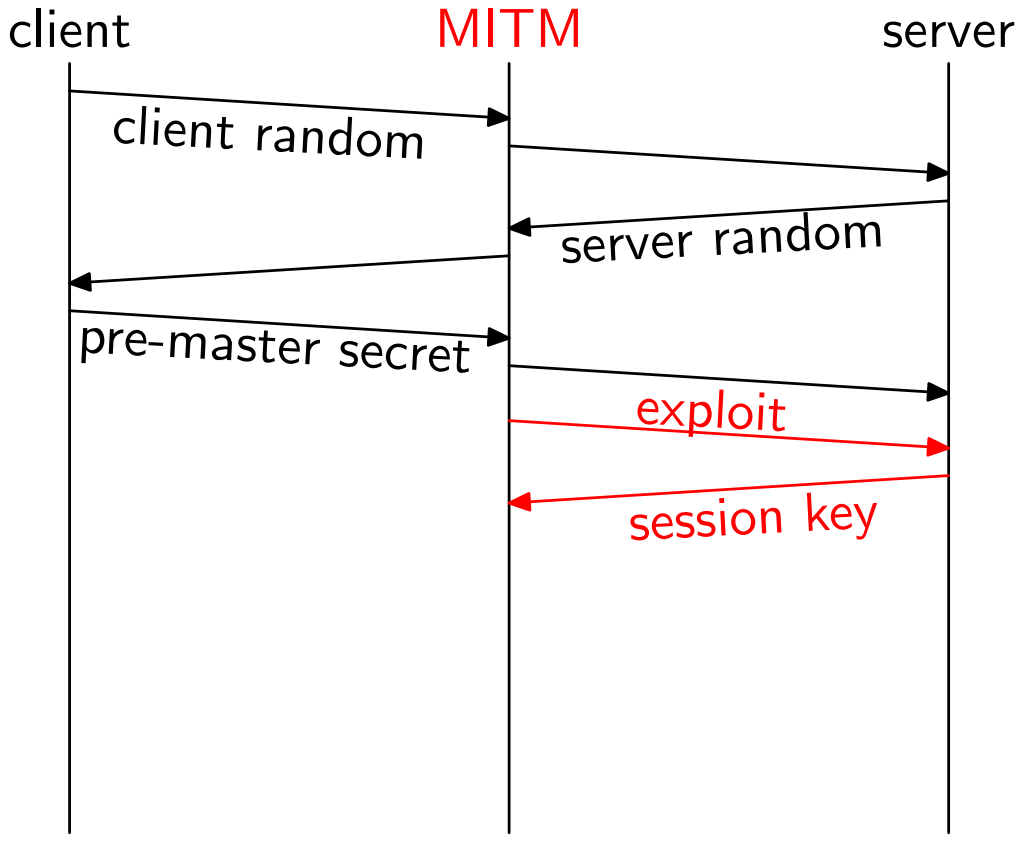
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



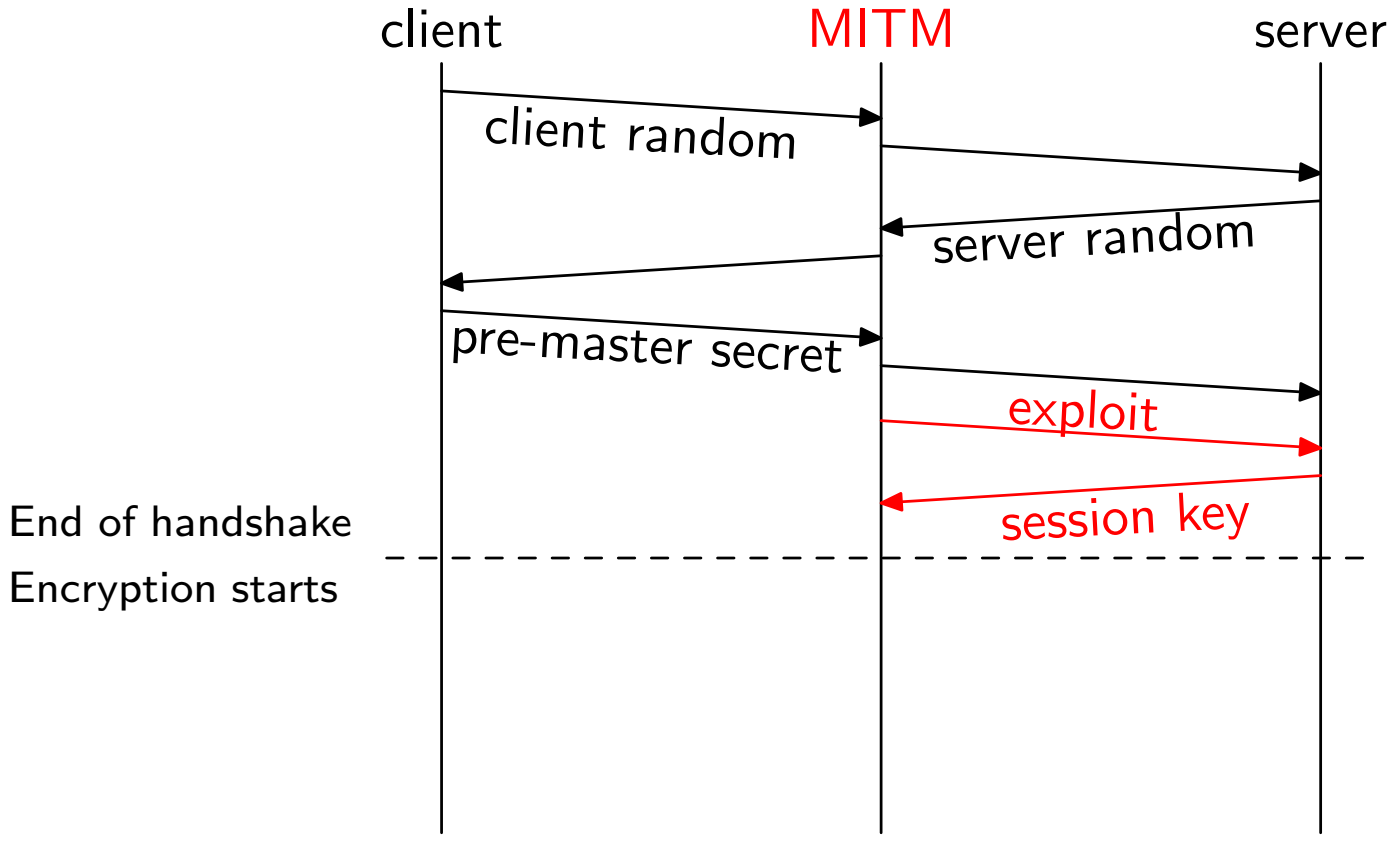
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



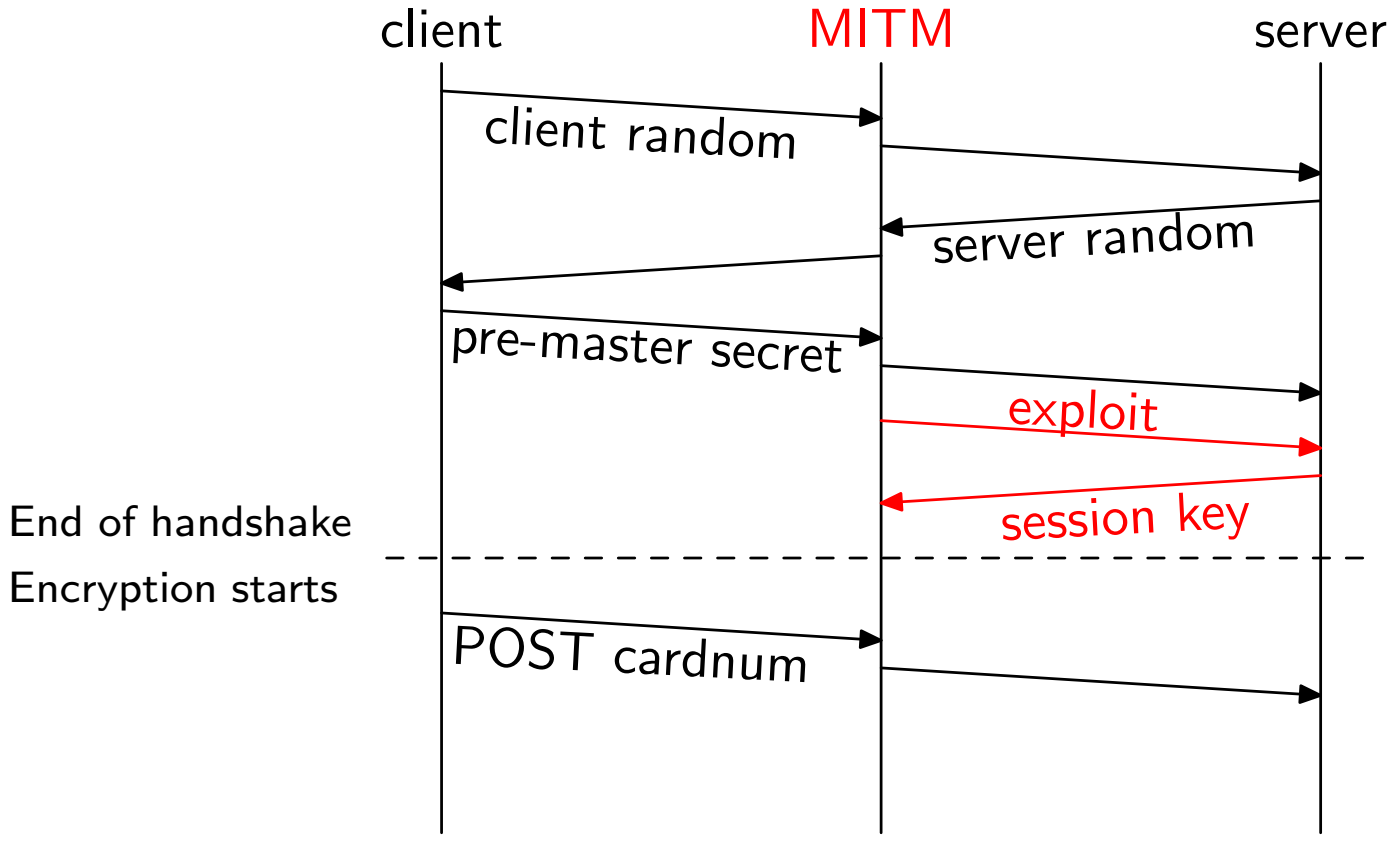
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



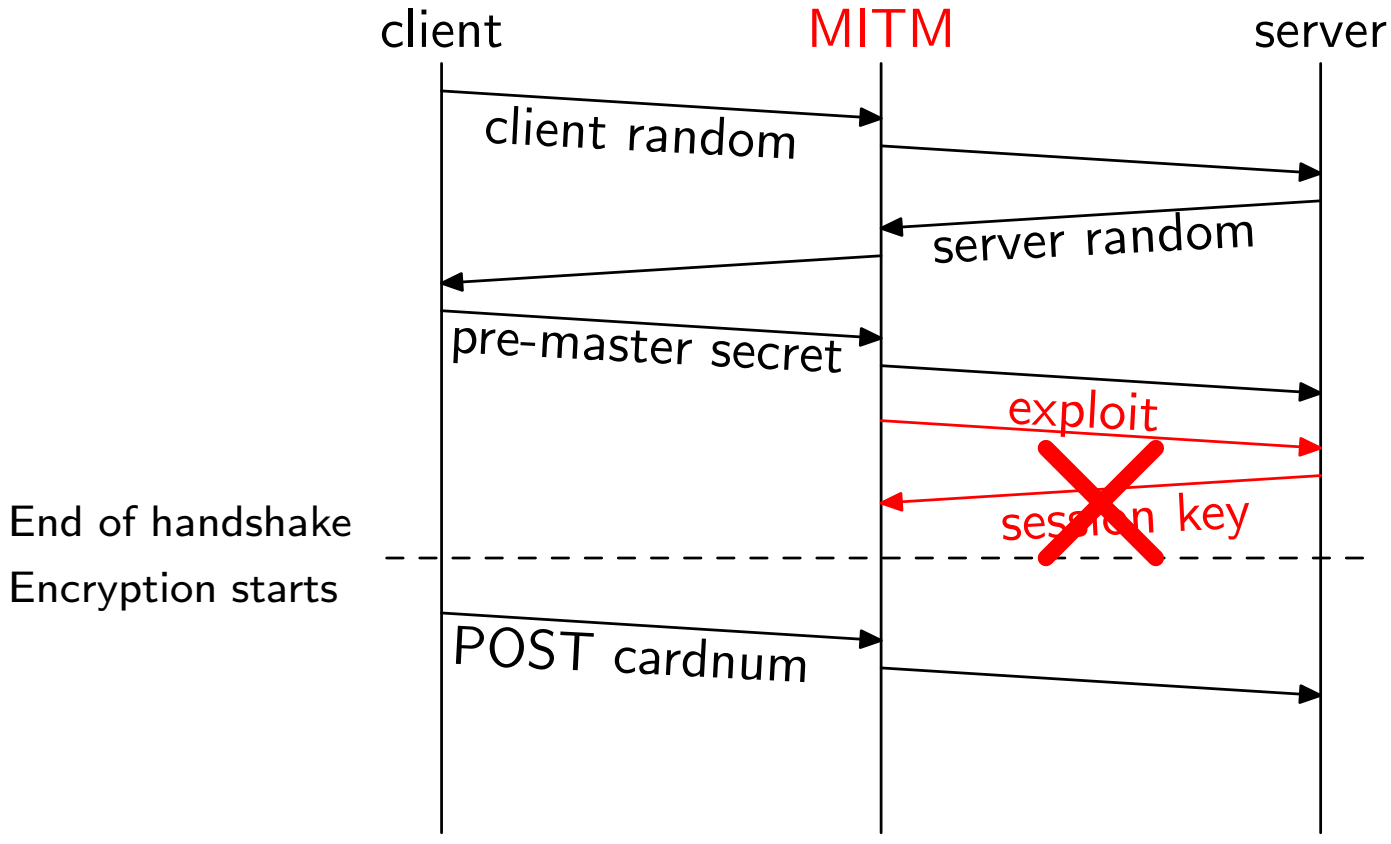
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



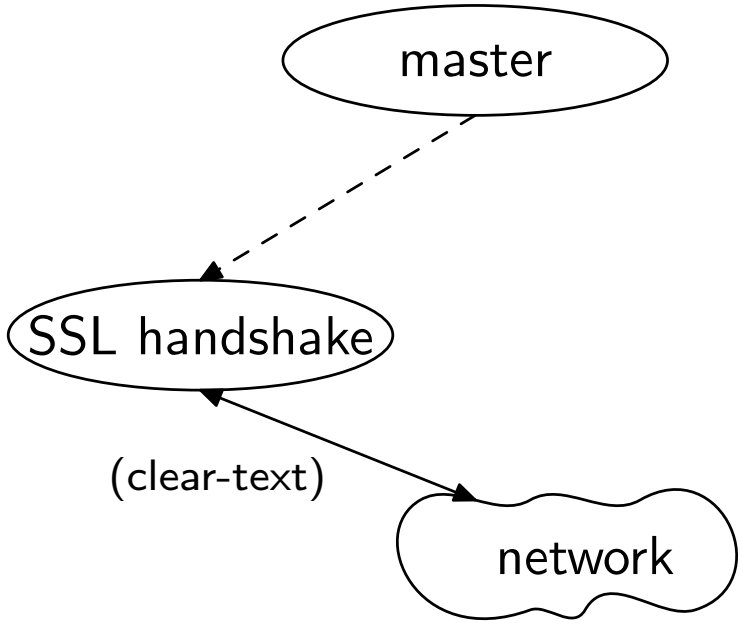
Vulnerable to man-in-the-middle

Disclosing session key causes a security breach with man-in-the-middle (MITM) attacks:



Man-in-the-middle defense overview

Can we protect against a MITM that has also exploited the server?

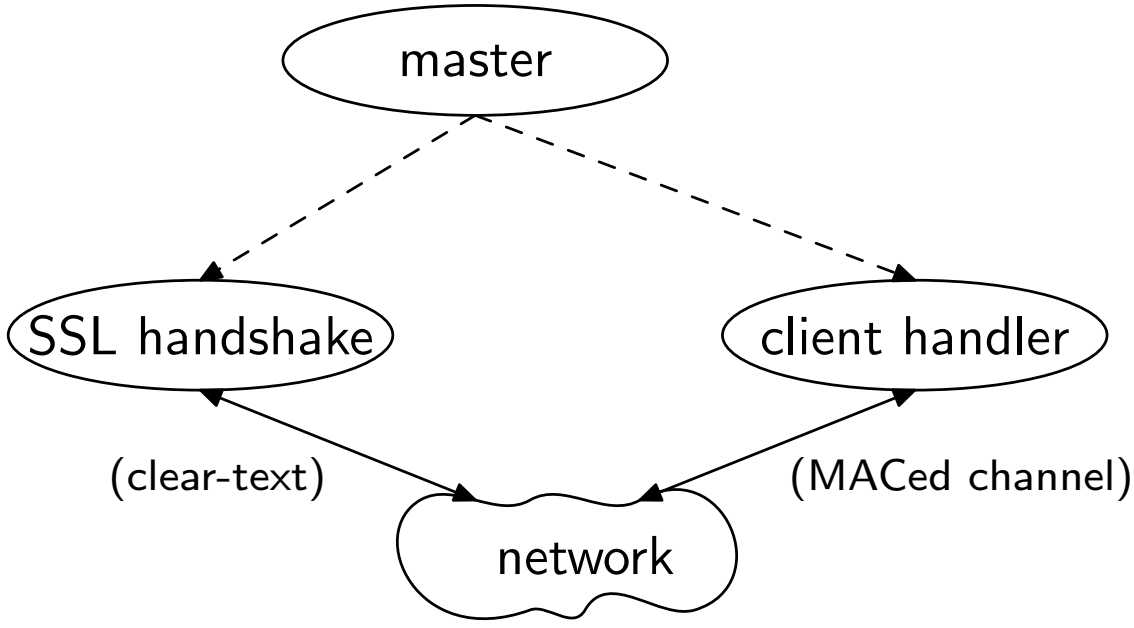


Strategy:

1. Prevent session key disclosure during handshake.

Man-in-the-middle defense overview

Can we protect against a MITM that has also exploited the server?



Strategy:

1. Prevent session key disclosure during handshake.
2. MITM cannot exploit client handler without session key: packets with invalid MAC will be dropped.

Implementation

Sthreads:

- Linux v2.6.19. 496 line diff, 1485 line module.
- Userland library: 1154 lines.

Crowbar:

- Binary instrumentation tool (using Pin): 2391 lines.
- Post processor: 959 lines.

Applications we partitioned using Wedge:

- Apache+OpenSSL.
- OpenSSH (prior to privilege separation).

Wedge reduces size of privileged code

Have we reduced the size of the privileged code?

Wedge reduces size of privileged code

Have we reduced the size of the privileged code?

Line counts in Wedge's Apache+SSL

Component	Line count	Percentage
Apache+OpenSSL total	252,030	100%
Default config after accept	60,844	
Callgates total (privileged)	15,769	6%

Lines changed when partitioning: 1,700 (0.7%).

Crowbar performs acceptably for developers

Crowbar is used by developers for partitioning. It is not an overhead seen during production run-time.

Does Crowbar perform acceptably for developers?

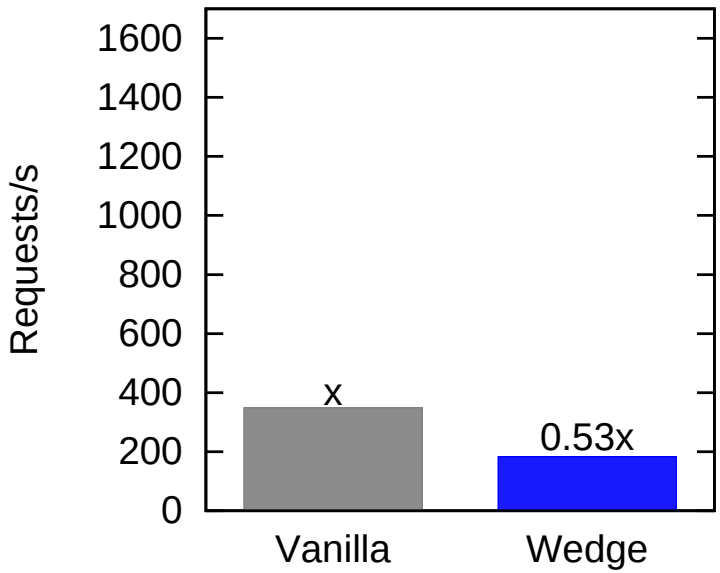
- A trace for Apache was obtained in 15s.
- Traces for SPEC applications: 82s on average.

Anecdotally, one trace was enough for our Apache (and OpenSSH) partitioning.

Enhanced privacy at acceptable cost

Throughput of many clients retrieving small static page:

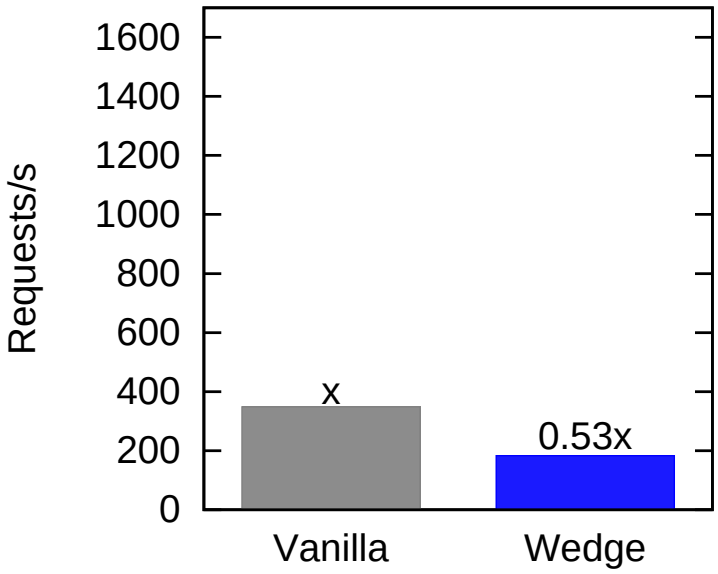
No sessions cached



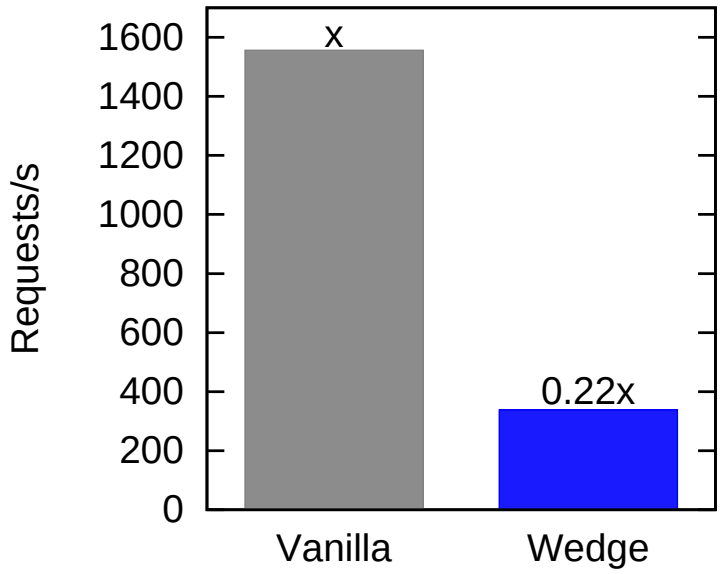
Enhanced privacy at acceptable cost

Throughput of many clients retrieving small static page:

No sessions cached



All sessions cached



- Vanilla reuses workers—we create new sthreads.
- We create many compartments & callgates per session.

Related work

We build on privilege separation: OpenSSH, OKWS, Privtrans

- Wedge allows finer-grained partitioning, and with default-deny, encourages tighter privileges for each compartment.

DIFC: JIF, Asbestos, HiStar, Flume, DStar

- Crowbar is complementary: could help partitioning legacy code in DIFC systems.
- Wedge does not allow unprivileged code to compute over sensitive data.

Conclusion

Wedge:

- Generalizes privilege separation and provides primitives for fine-grained default-deny partitioning of applications.
- Crowbar: tool to aid in partitioning legacy code.

Wedge enables fine-grained partitioning of legacy code:

- Programmers can defend applications against stronger adversaries and more complex threat models than those addressed to date.

`http://nrg.cs.ucl.ac.uk/wedge/`