Correlating Low-Level Events To Identify High-Level Bot Behaviors

Liz Stinson
John Mitchell
Stanford University

Matt Fredrikson
Somesh Jha
University of Wisconsin

Lorenzo Martignoni
University of Milan

Our anti-inspirations

- "Pe[illegible]n app is connecting to the network"

Too ambiguous

- Host-level methods that inundate us with in[illegible], file accesses/changes) without providing a higher-level assessment of what's going on

Too noisy; devoid of meaning

Problem Statement

- >5M “distinct, active” bot-infected machines detected between January - June, 2007
  - “active”: carried out at least one attack
  - Symantec Threat Report, Volume XII
- The *best* anti-virus signature scanners fail to detect anywhere from 30% to 50% of malware samples *seen in the wild*
  - NB: The best AV scanners may not be who you think they are…

Problematic Asymmetry

work **ant)**

Malware writers know they have the advantage here and they exploit it.

- AV
  mal
  exc

d
must

Tens of thousands of novel malware variants created annually

Existing behavior-based detection

- Id
  (p
  - shadow?
  - App survives reboot? Spawns/terminates other
  - nged?
- → etect
- S
- More general characterizations
  - Abstract: spyware monitors/reports user actions
  - Concrete: rootkits that load kernel modules

May identify *incidental*, rather than **fundamental** behaviors

For ML-based approaches, may be other ways to achieve same end (i.e. ways not included in model)

Broad spectrum. How to evaluate?

- How *effectively* does this method distinguish malicious behavior from benign?
- How *thoroughly* is target behavior captured?
- How *complex* is the identified behavior?
- How *fundamental* is the behavior to the malware's purpose?

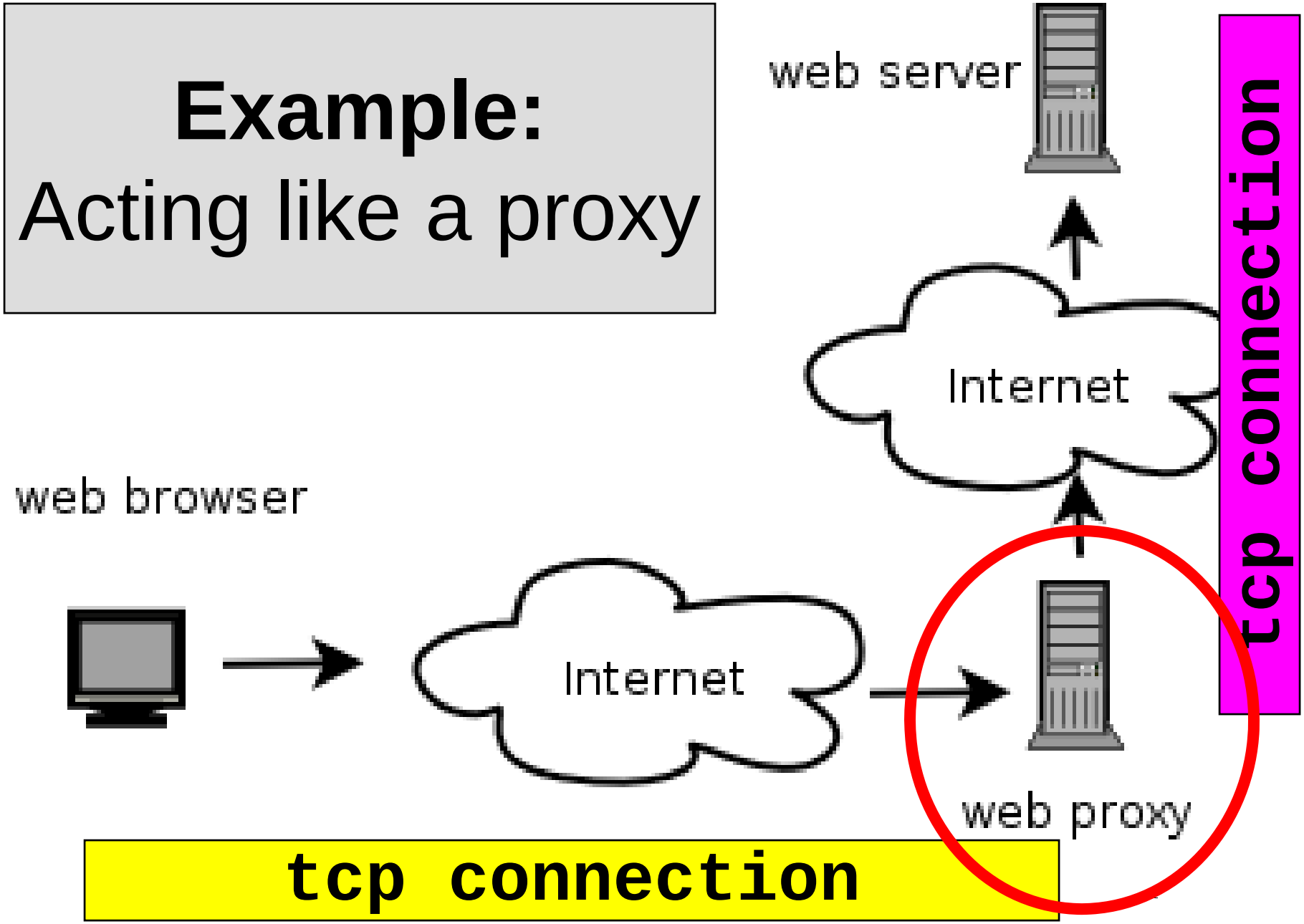
Goals

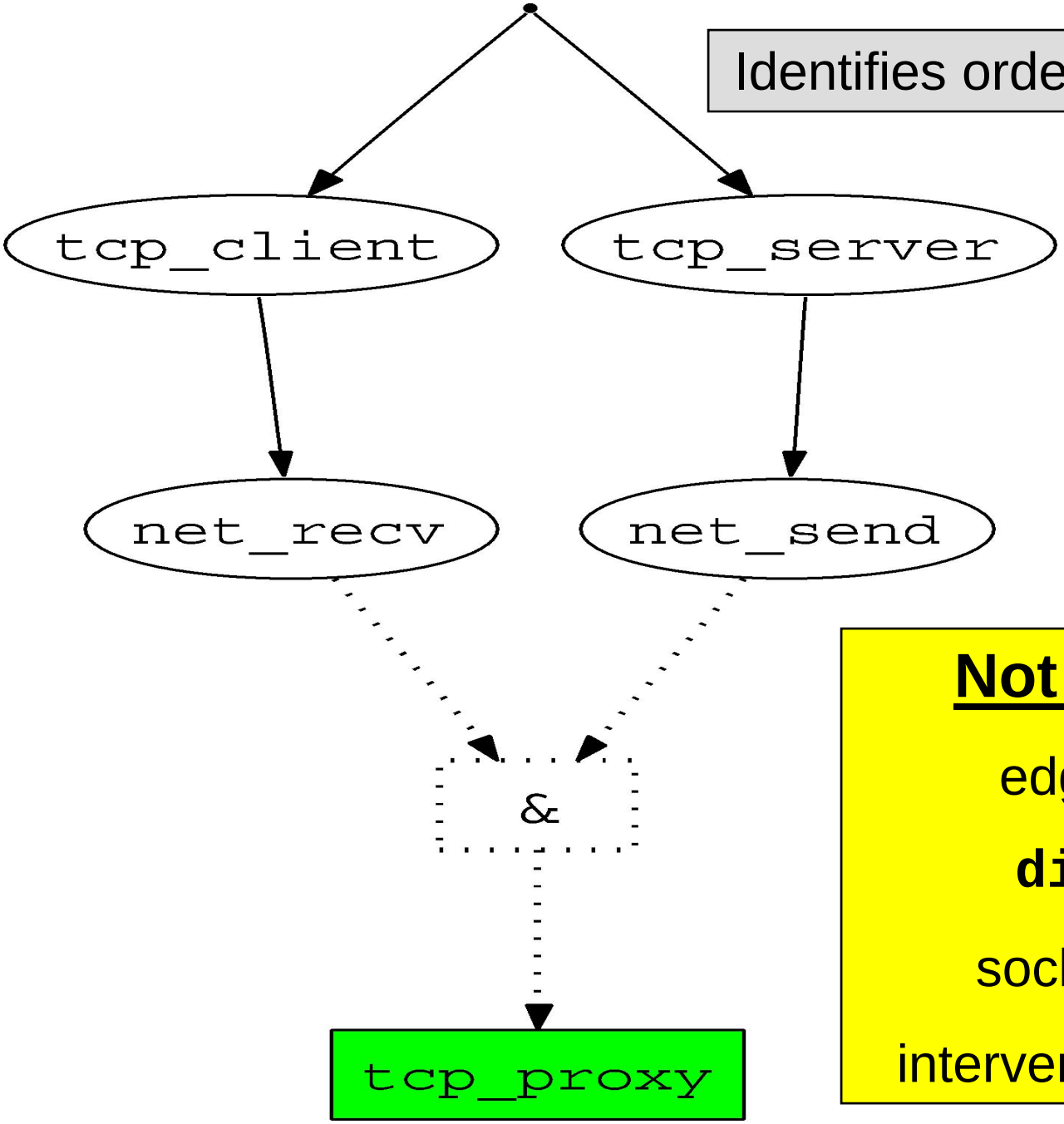
- We wa… …ors
  - "downloading and executing a program"
  - …
  - …
  - …
- B…
- Via monitoring process execution
- Distinguish malicious from benign instances of above by identifying if *remotely initiated*

Sample bot commands

```
http.execute <URL> <local_path>
harvest.registry <reg_key>
redirect <lport> <rhost> <rport>
startkeylogger
```

Example:
Acting like a proxy





Identifies ordering dependencies

Not shown here

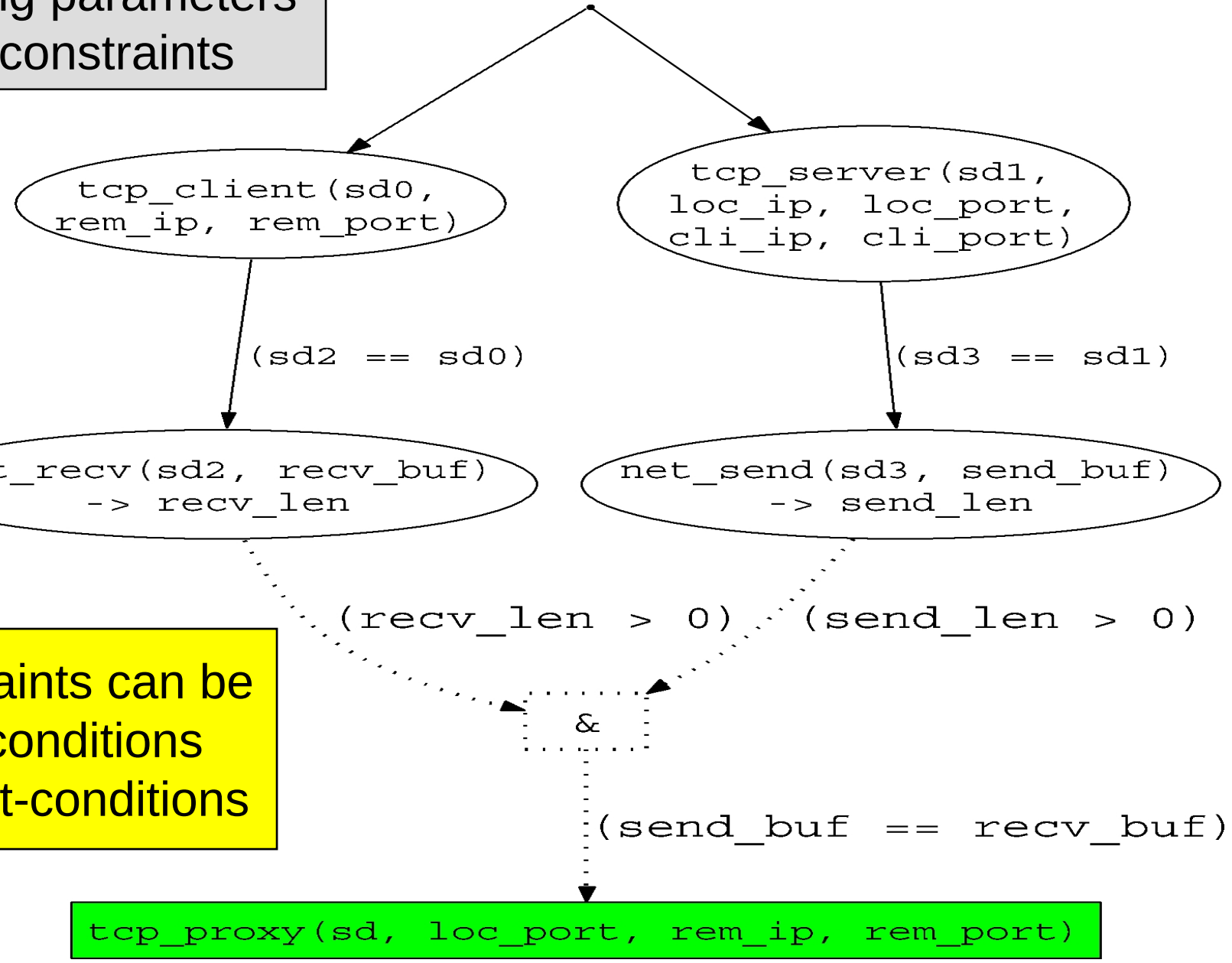
edge constraints

`die` operations

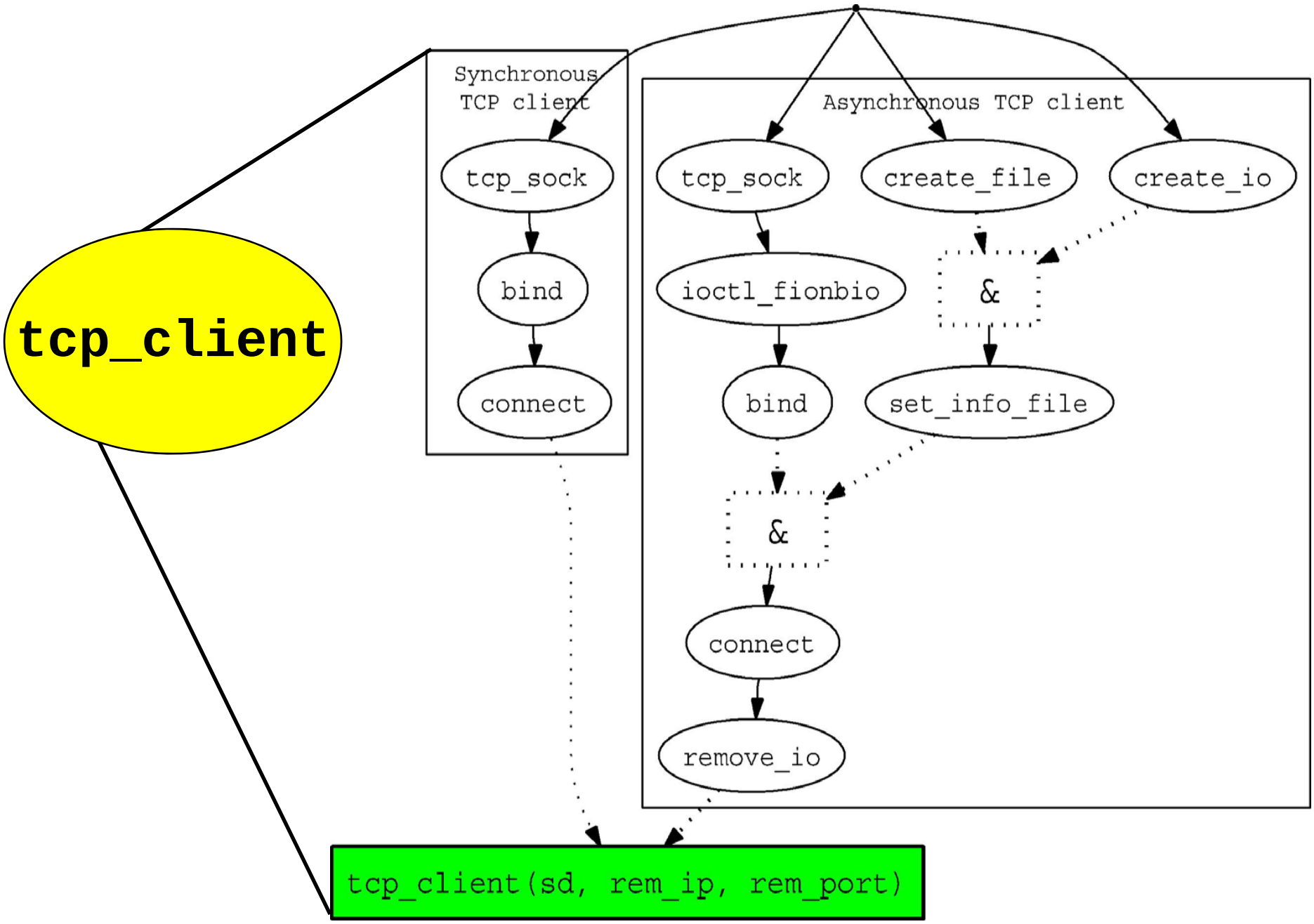
socket duplication

intervening irrelevant ops

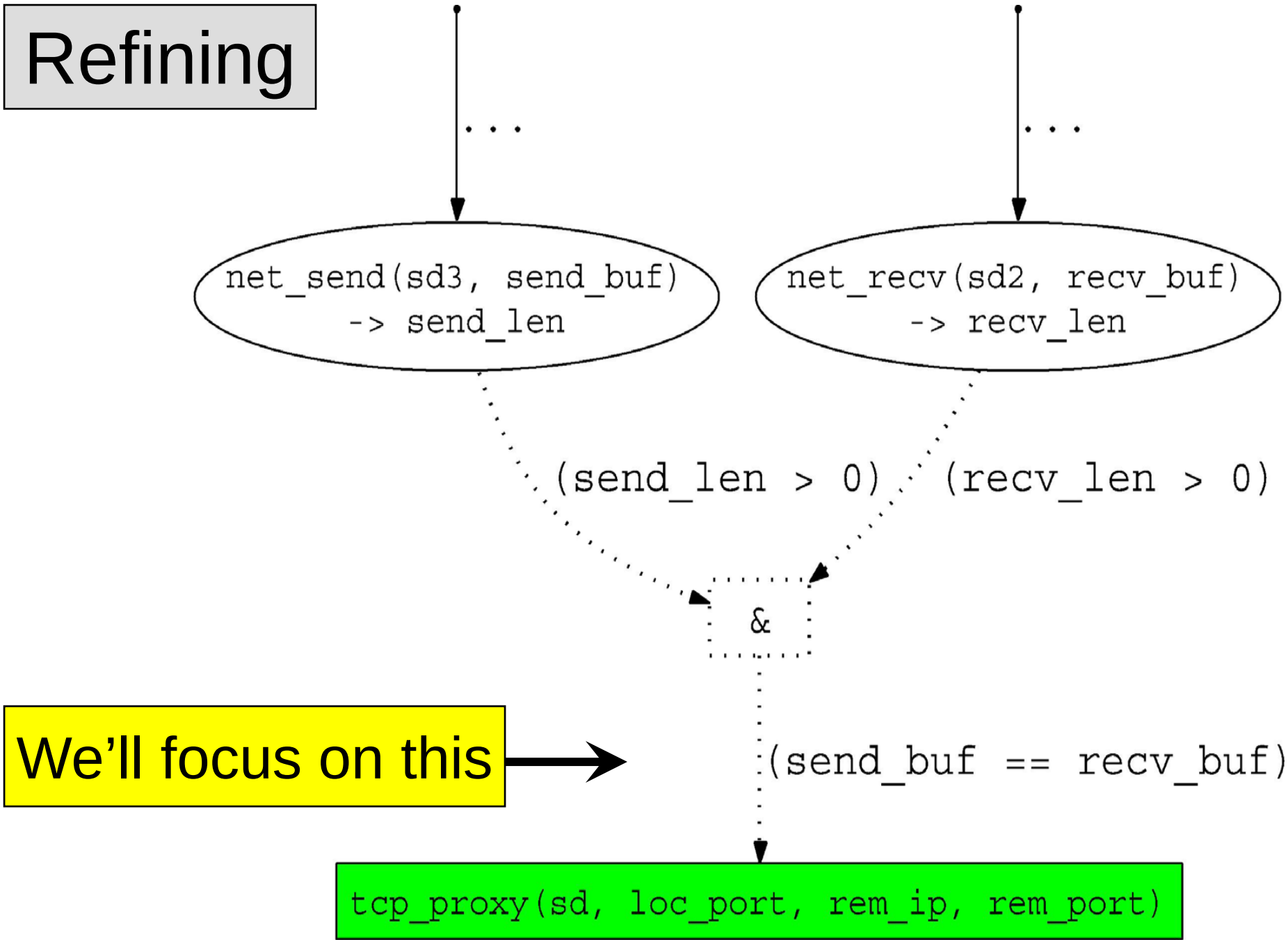
Including parameters and constraints



Constraints can be pre-conditions or post-conditions


tcp_client
Synchronous
TCP client
tcp_sock
bind
connect
Asynchronous TCP client
tcp_sock
create_file
create_io
ioctl_fionbio
&
bind
set_info_file
&
connect
remove_io
tcp_client(sd, rem_ip, rem_port)

Refining



We'll focus on this →

(send_buf == recv_buf)

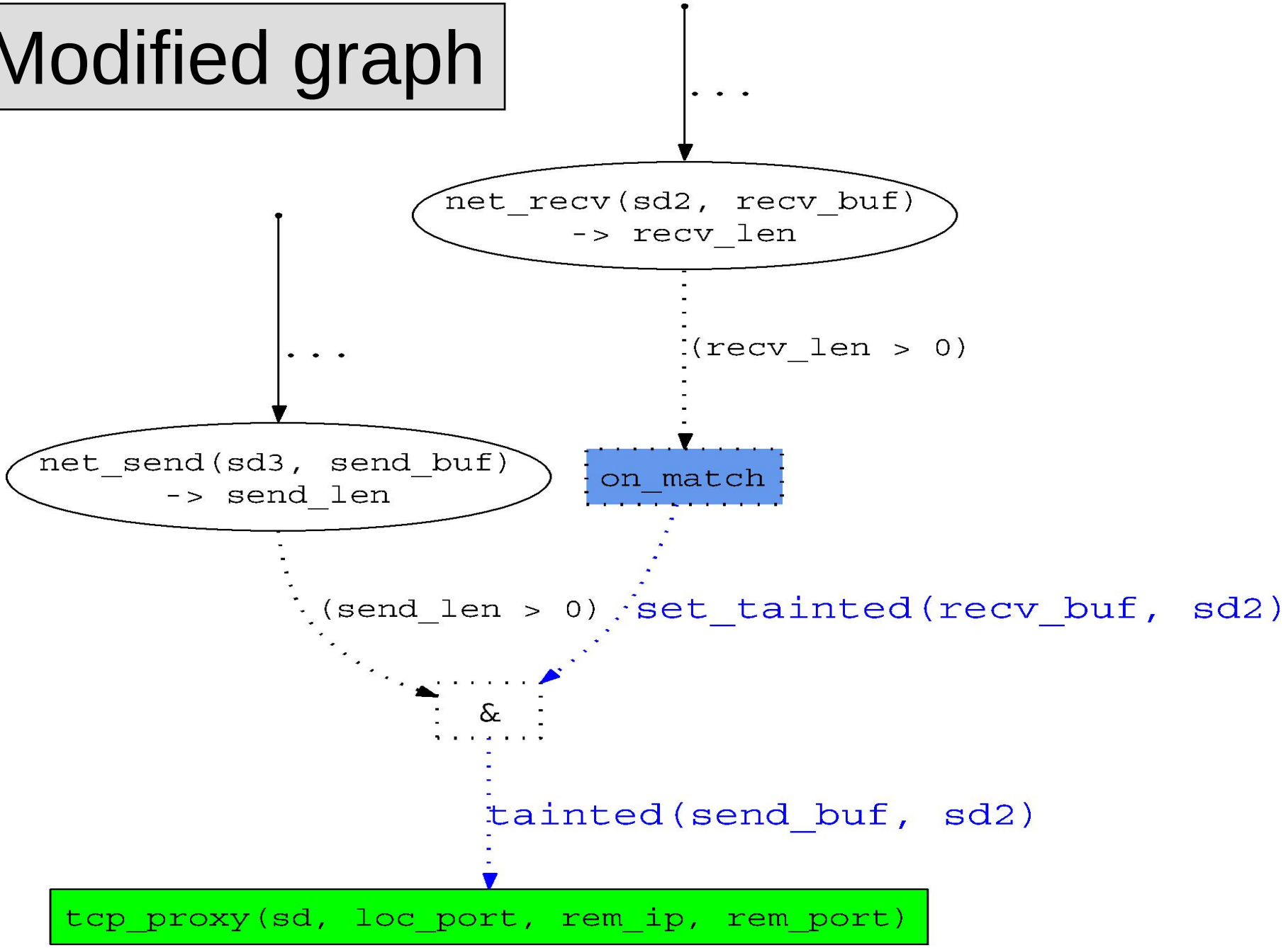
- Too constrained; really want to express: the buffer that is sent is *derived from* a buffer that is received
- Augment (add action to): **`on_match`** of **`net_recv`**

```
set_tainted( recv_buf, sd2 /*taint label*/ )
```

- Change condition to:

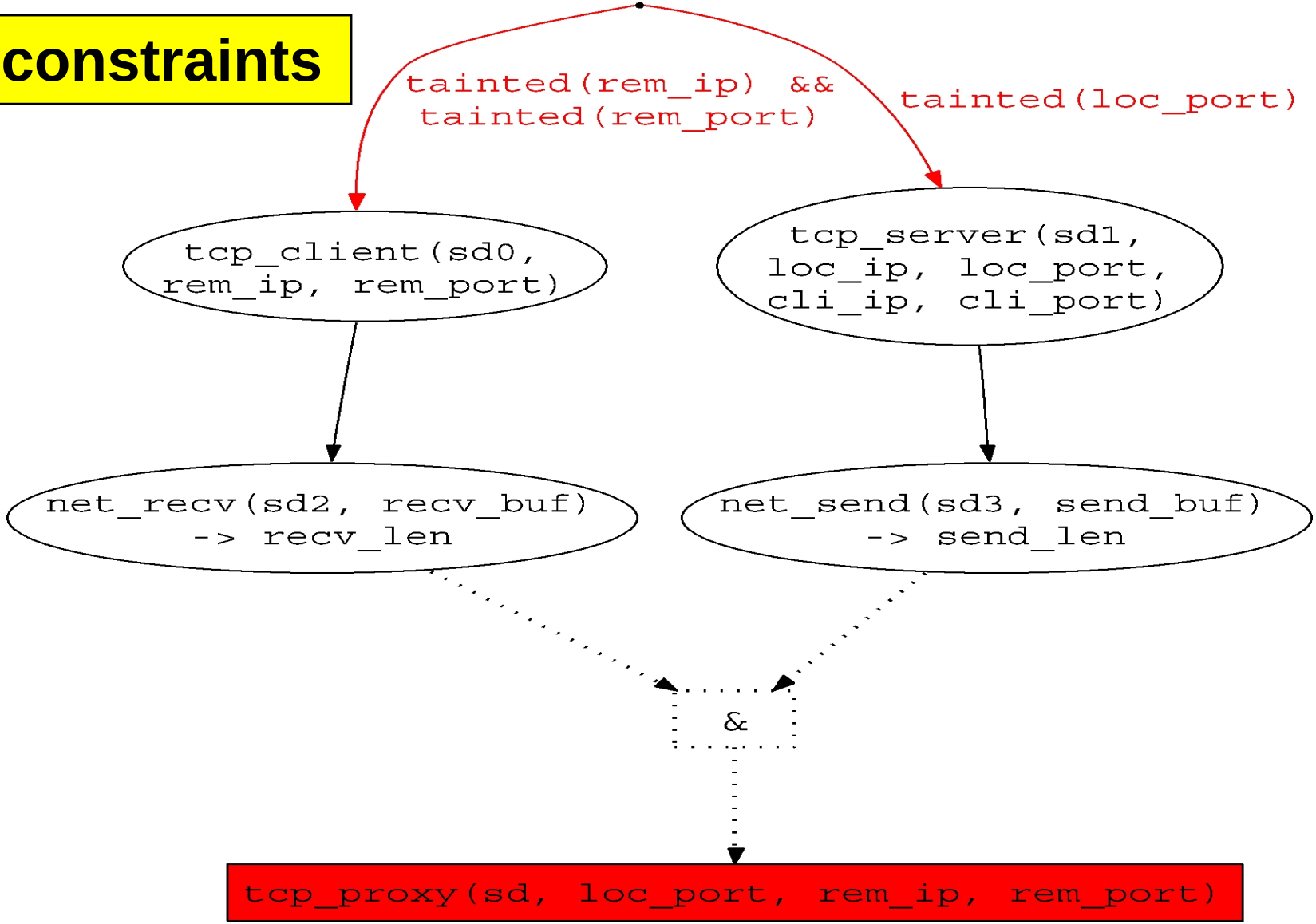
```
tainted( send_buf, sd2 /*taint label*/ )
```

Modified graph



.redirect <loc_port> <rem_host> <rem_port>

Add constraints



“Language” our system exports

- Set of high-level primitives that can be combined to describe interesting behaviors
  - **`tcp_client`**, **`tcp_server`**, **`net_send`**, **`net_recv`**, **`create_exec_file`**, **...**
- Using these, we can detect:
  - Leak private data (reg key values, file contents, system info, …)
  - Download and execute a program
  - Send email
  - Proxy
  - Keystroke logging

Challenges

- Posed by proprietary-OS environment
  - Opacity; identifying operations & constraints
  - Replicating OS semantics
- Posed by syscall interposition generally
- Posed by hypothetical attempts to evade
  - Split behavior across processes or across runs of the same application
  - Expropriate kernel functionality
    - e.g. raw sockets

Summary

- Target the behaviors that make bots useful
- Identify the essential ops in those behaviors
- Use data-flow analysis info variously
- Good initial results against bots
  - Including: rbot, agobot, dsnxbot, spybot, ...
  - Use bot commands as inspiration
  - Resilient to encryption of bot communications
- Good initial results against benign progs
  - When testing against specifications that encode remote-control requirement
  - Performing user-input tracking