TLS as an Operating System Service

Daniel Zappala

from Mark O'Neill's PhD Dissertation and USENIX Security papers

Part 1
TrustBase
Simplified and Centralized Certificate Validation

certificate validation problems

certificate authorities (CAs)

- generally can sign certificates for any host (Eckersley et al.)
- have been hacked, sometimes repeatedly (Marlinspike)
- can be influenced and operated by governments (Soghoian et al.)
- don’t always follow best practices (see CNNIC)

for application developers

- mobile and desktop apps have validation problems
  - Brubaker et al., Georgiev et al., Onwuzurike et al., Fahl et al.
- security libraries are complicated
- security may not be a priority
- security may be a hassle

threat model

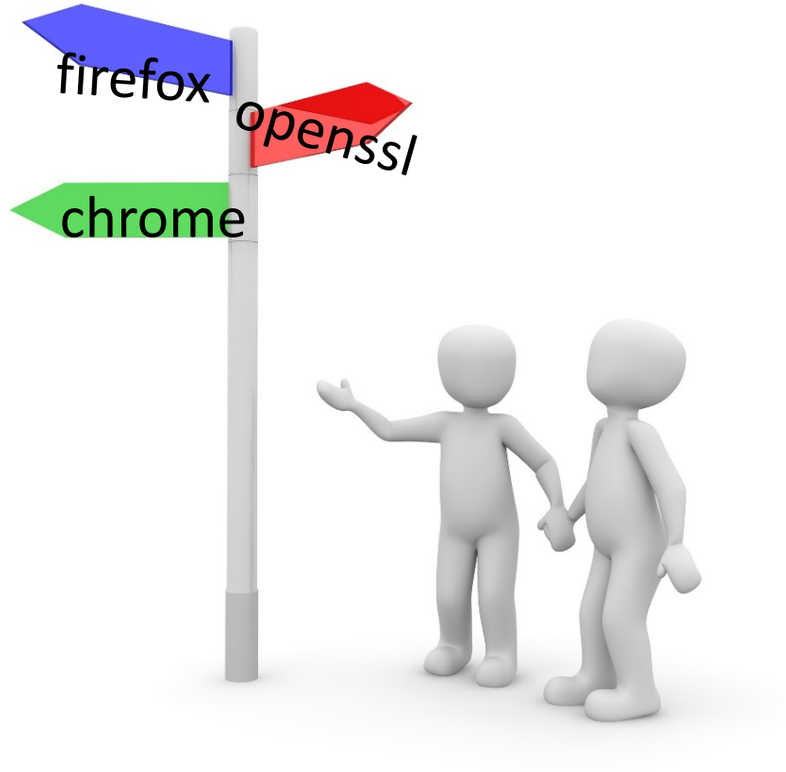
client

Bad guys, Inc.

amazon

alternate and reinforcing strategies

- deal with many of these issues
- have no common platform or API
- have difficulty being adopted



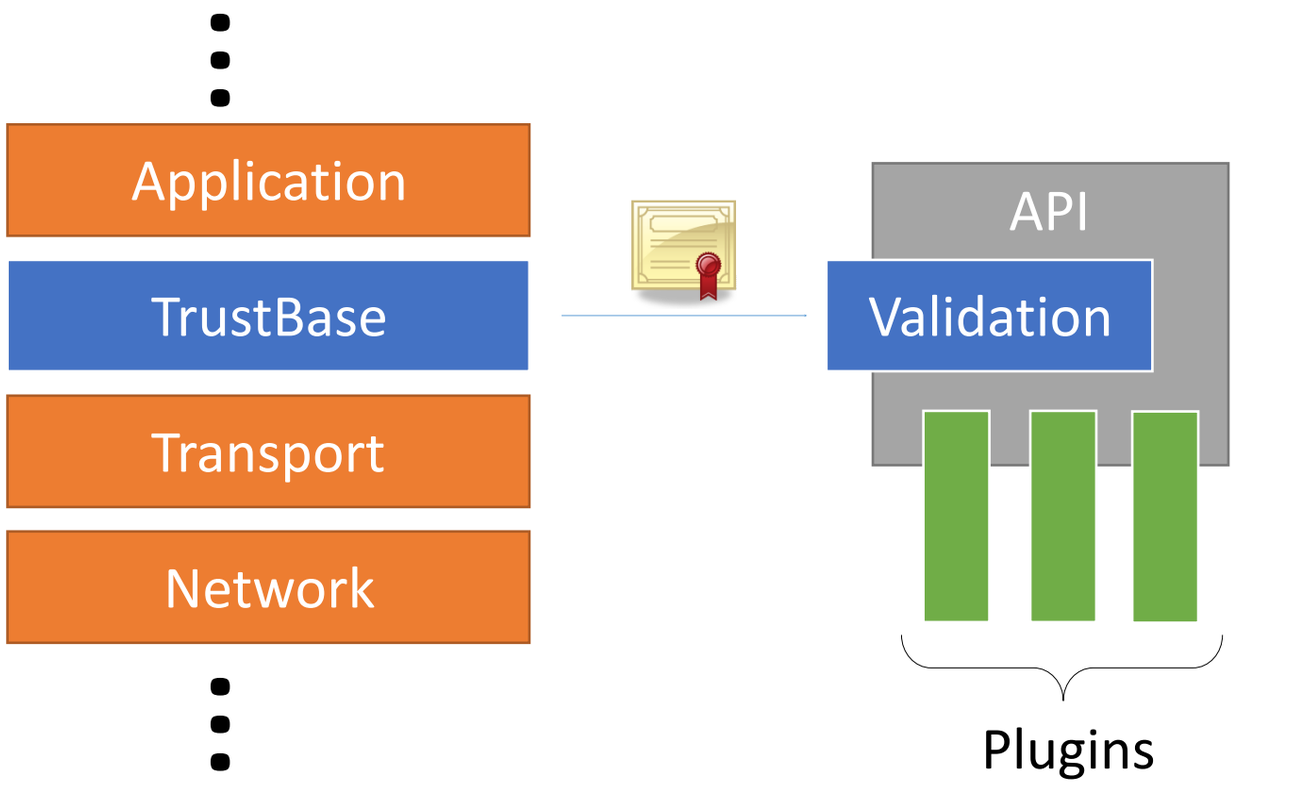
trust decisions are outsourced

how do we enable admins to control the trust decisions of their own machines?

TrustBase

- motivating principles
  - centralize authentication as an OS service
  - empower system admins to dictate how trust decisions are made on their own machines
- design goals
  - secure all existing applications
  - prohibit unprivileged applications from acting against administrator rules
  - provide easy deployment of authentication systems
  - negligible overhead

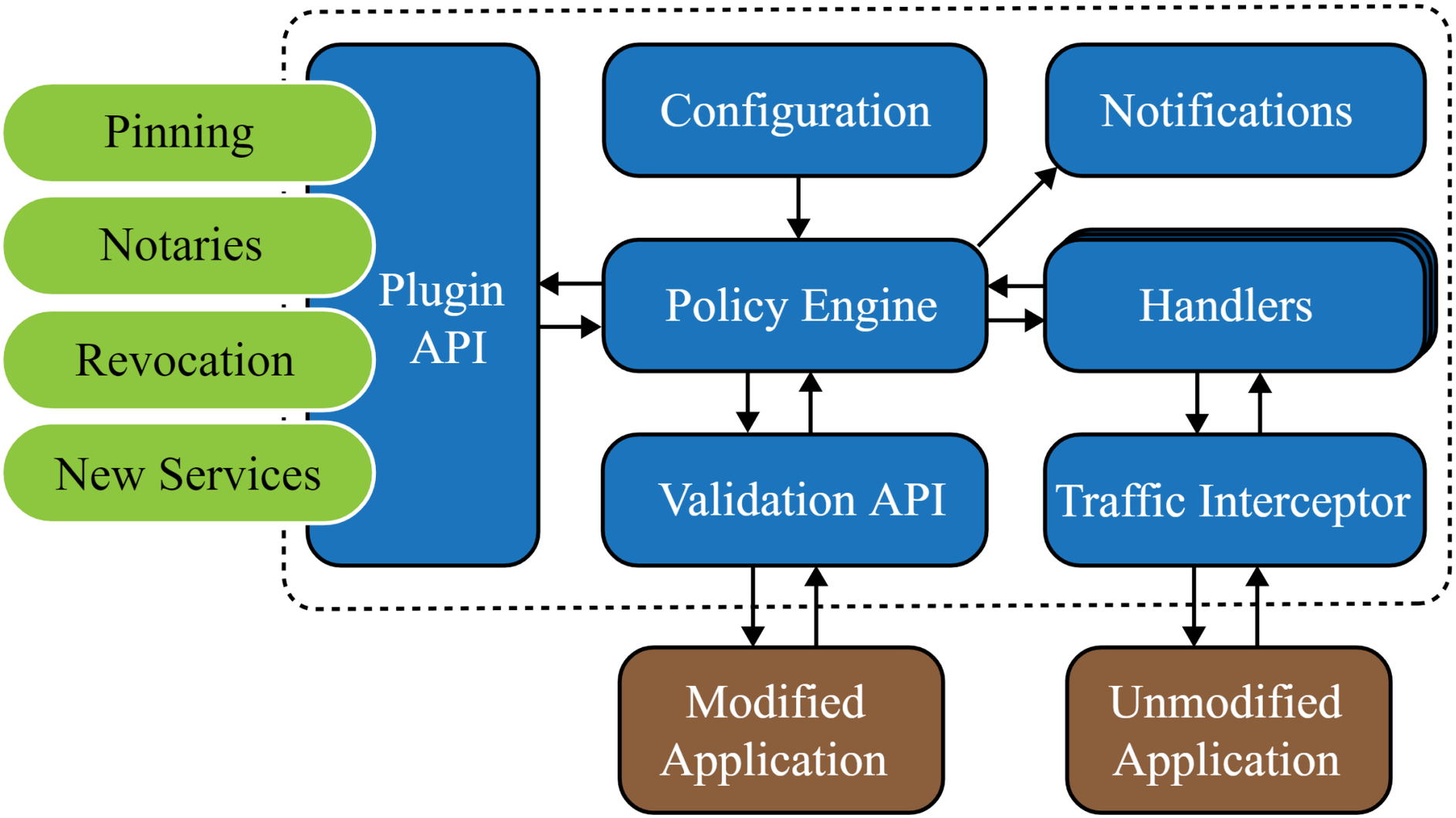
moving trust to the OS



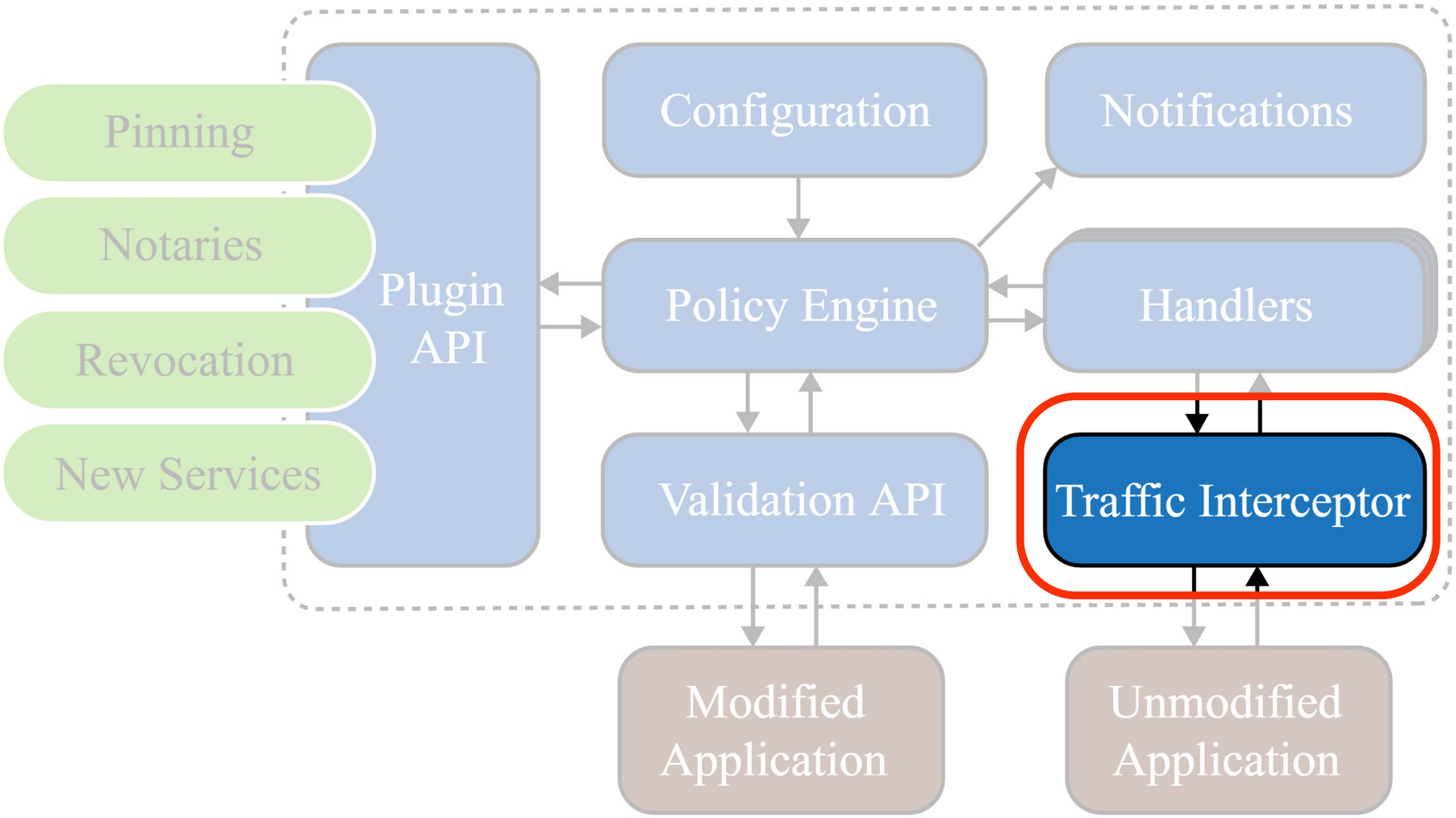
Prototypes for

- Linux
- Android (nonrooted)
- Windows

TrustBase architecture

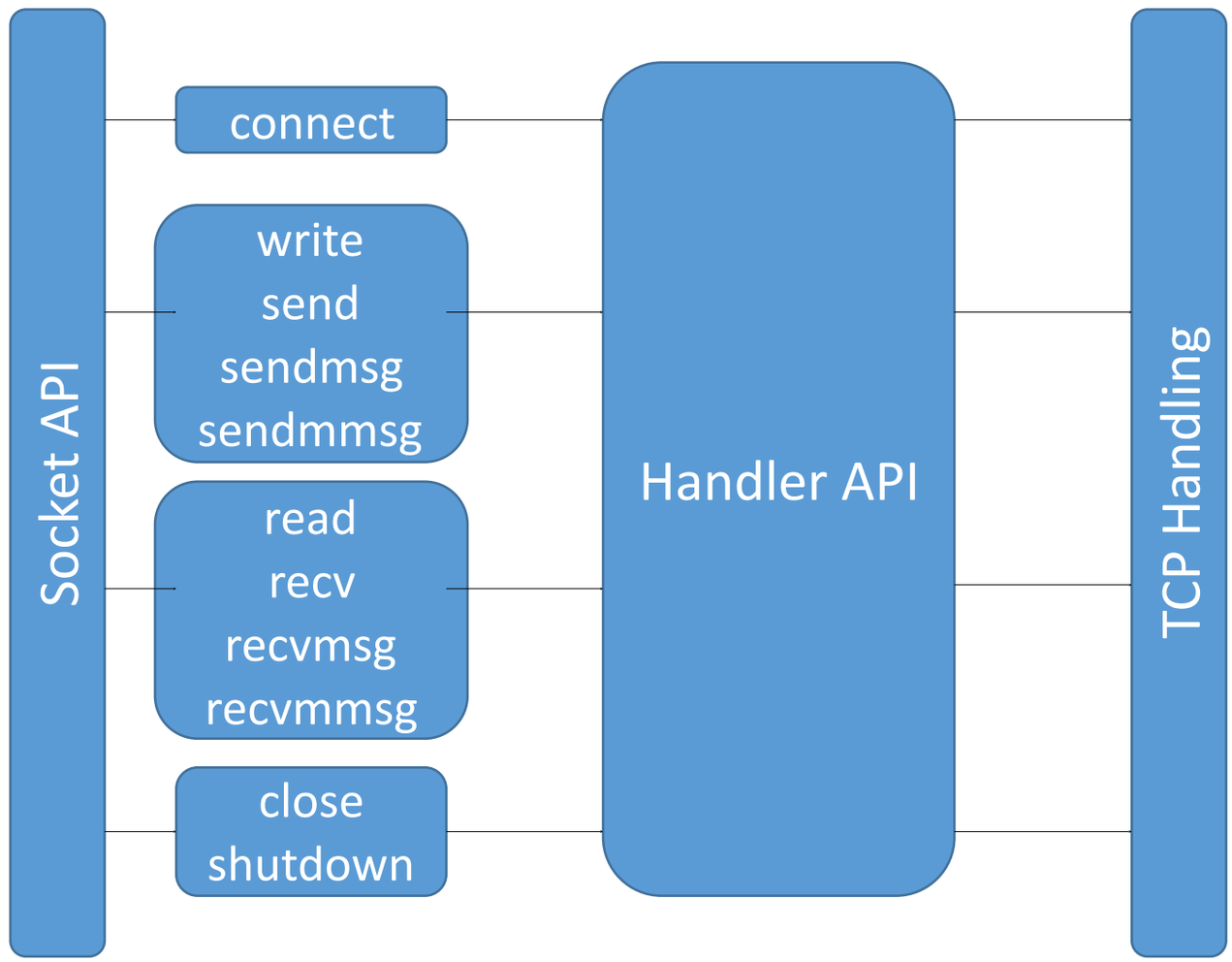


TrustBase architecture

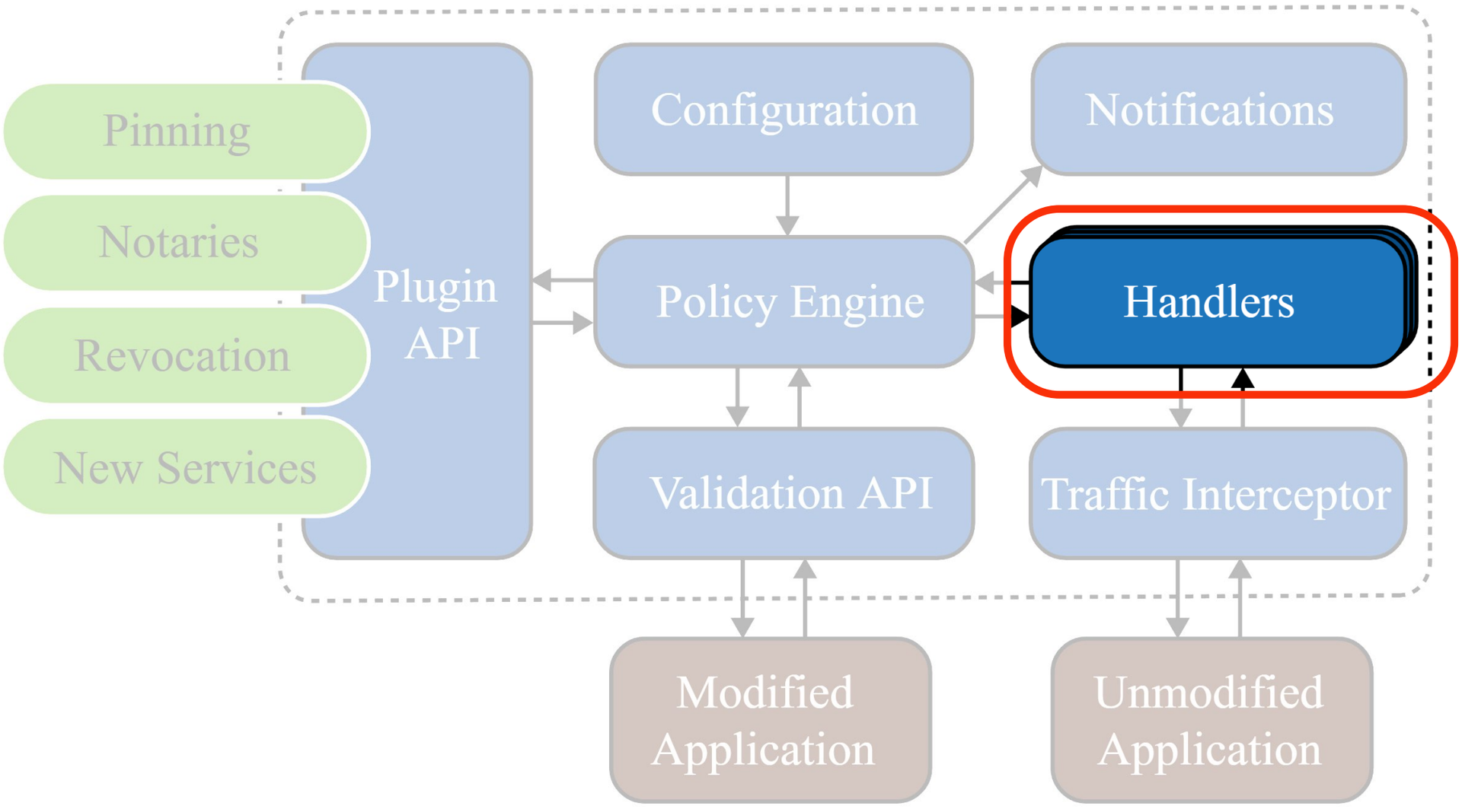


traffic interception (Linux)

- loadable kernel module
- hooks into native transport protocol functionality
- provides generic inspection/modification API



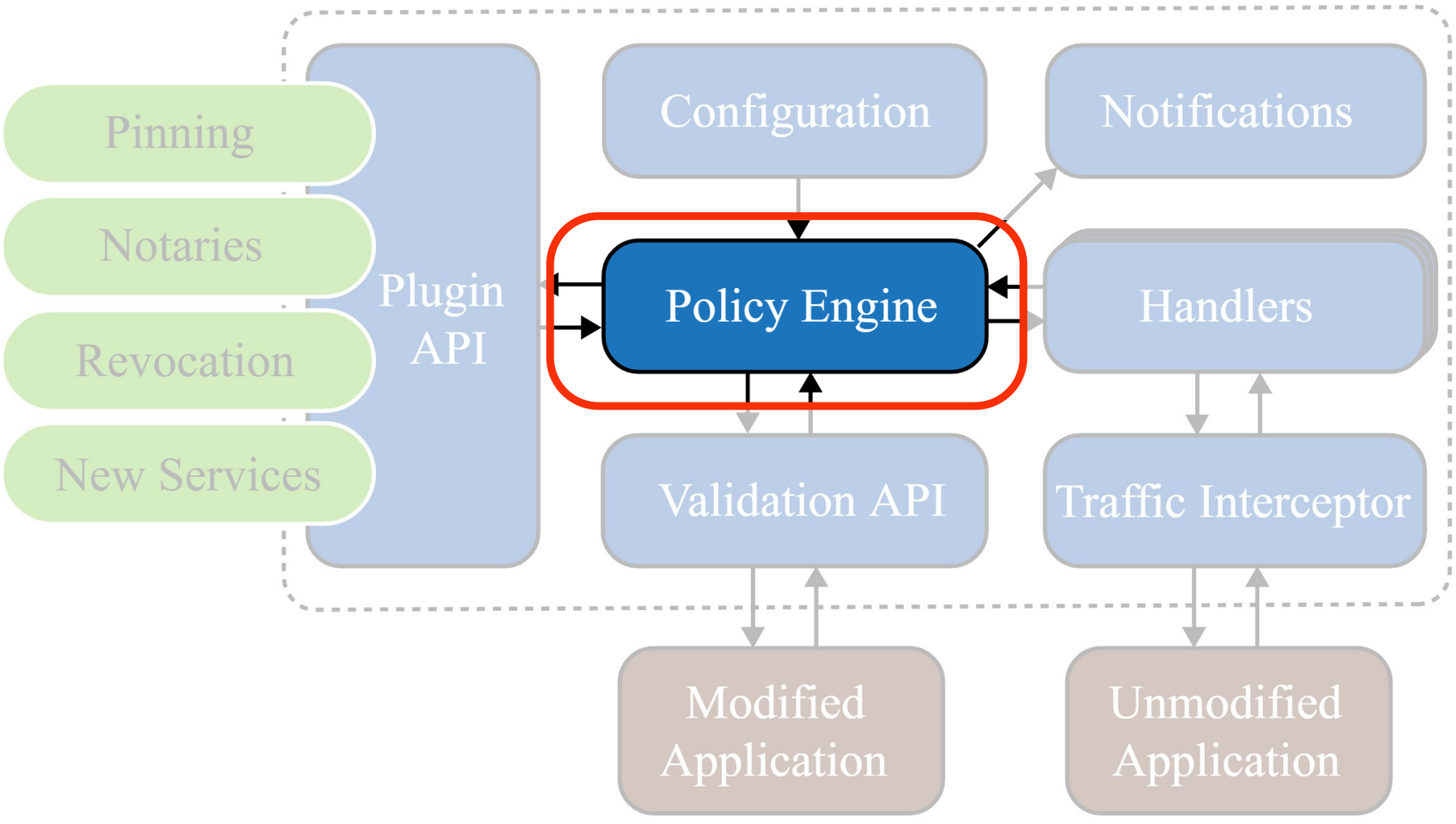
TrustBase architecture



TLS handler

1. monitor traffic for TLS records
2. record handshake messages
3. query policy engine with handshake data
4. receive policy response
    1. block connection if invalid
    2. allow if valid

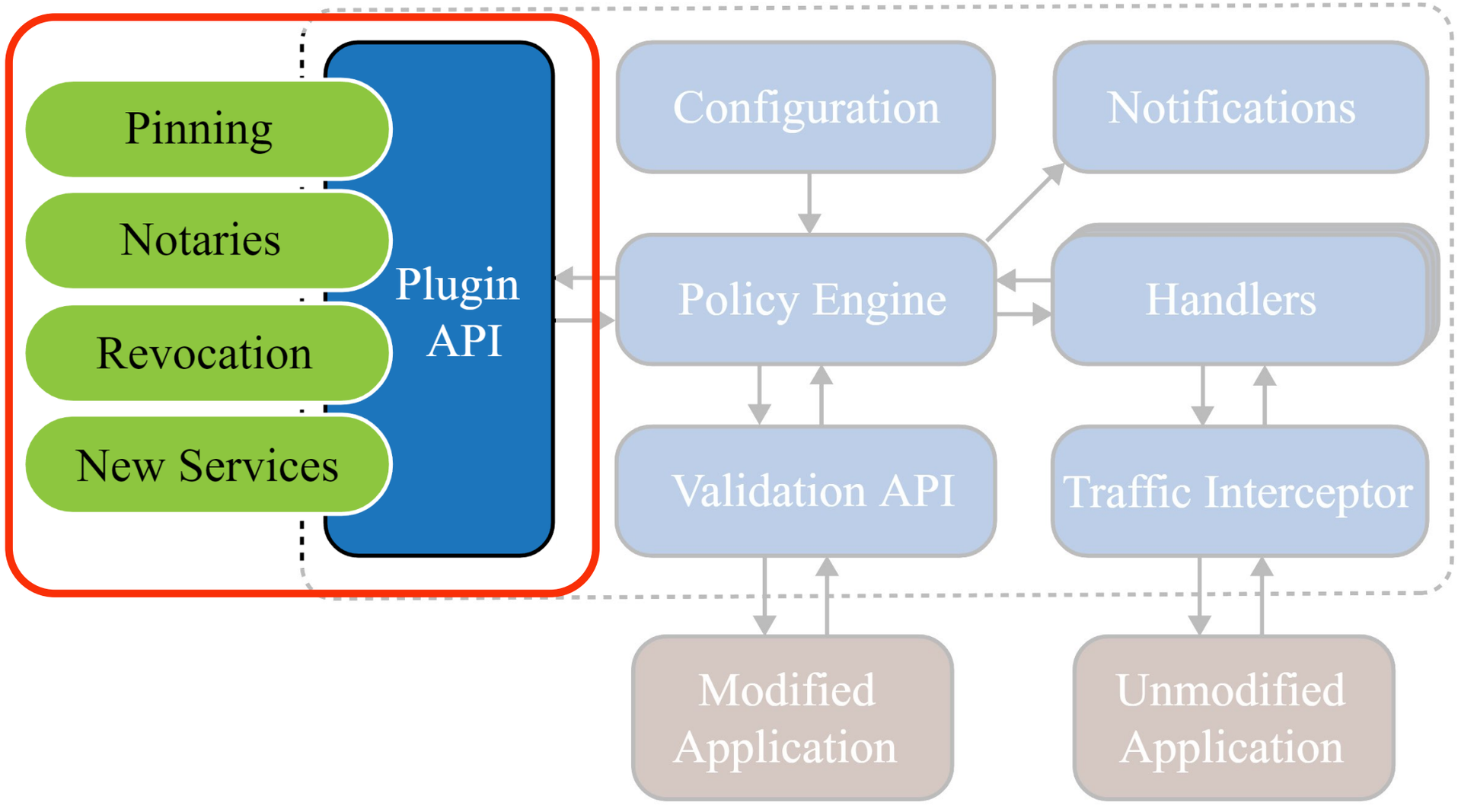
TrustBase architecture



policy engine

- receives queries via Netlink
- implements basic CA validation
- aggregates decisions from plugins
  - necessary
  - voting
- provides native API
  - Linux capabilities

TrustBase architecture



plugins

- API allows synchronous and asynchronous plugins
  - openssl STACK_OF(X509) or ASN.1 DER
- can report back yes/no/abstain/error for each chain
- have access to all handshake info (and more)

addons

- provide additional language support for plugins
- currently have native C and python addons
- API to add additional language support

example plugins and uses

- CA Validation (builtin)
- Certificate Pinning
- OSCP checking
- CRLSet blocking
- DANE
- Notary
- Cipher Suite Auditor

evaluation

centralization and coverage

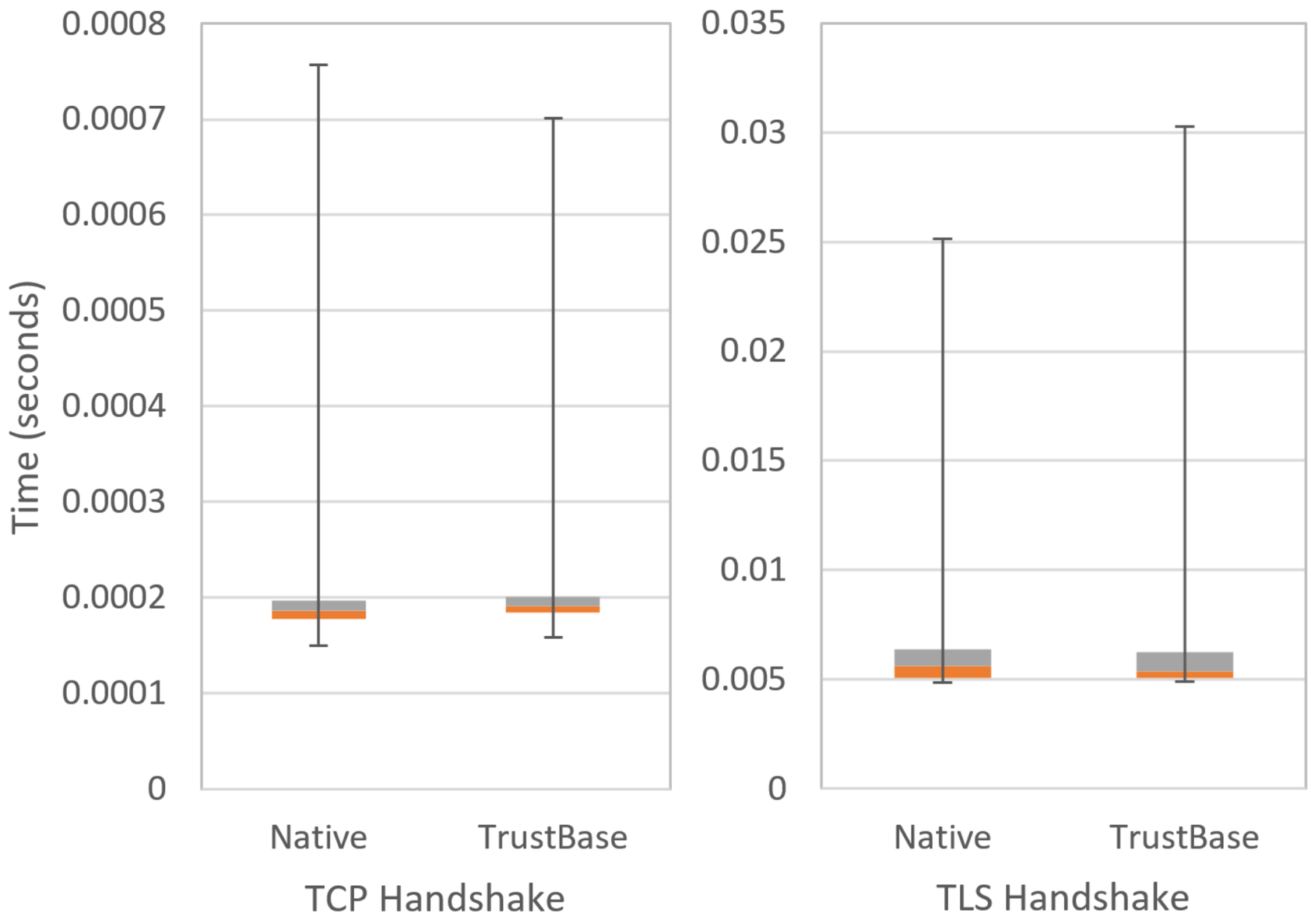
- bugs are global
- disruption is a DOS
- updates are global
- many eyeballs
- in line with other services

Library	Tool
C++	gnutls-cli
libcurl	curl
libgnutls	sslscan
libssl	openssl s_client
libnss	openssl s_time
JAVA	lynx
SSLSocketFactory	fetchmail
PERL	firefox
socket::ssl	chrome/chromium
PHP	mpop
fsockopen	w3m
php_curl	ncat
PYTHON	wget
httplib	steam
httplib2	thunderbird
pycurl	kmail
pyOpenSSL	pidgin
python ssl	
urllib, urllib2, urllib3	
requests	

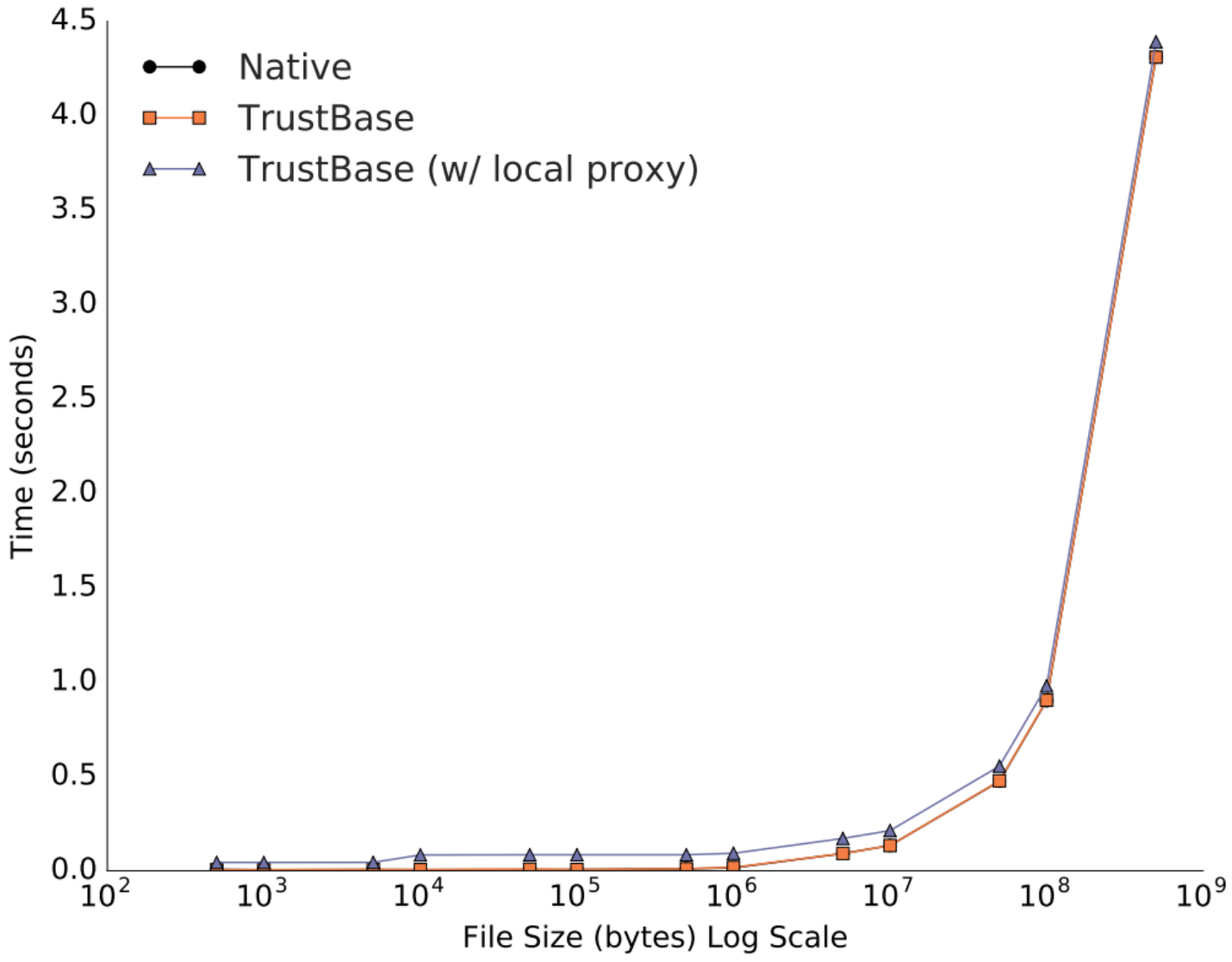
hardening

- unprivileged malware cannot unload interception
- CAP_NET_RAW is required to use raw sockets (default) and to bypass TrustBase interception
- CAP_NET_ADMIN required to receive and respond to queries
- configuration is writable only by privileged users
- daemons run nonroot with only required permissions

performance



performance



trustbase lets **you** trust

who you want

how you want

Part 2
Secure Socket API
Simplified and Centralized TLS API

your apps are vulnerable

Application Errors

We demonstrate that SSL certificate validation is completely broken in many security-critical applications and libraries. Vulnerable software includes Amazon's EC2 Java library and all cloud clients based on it; Amazon's and PayPal's merchant SDKs responsible for transmitting payment details from e-commerce sites to payment gateways; integrated shopping carts such as osCommerce, ZenCart, Ubercart, and PrestaShop; AdMob code used by mobile websites; Chase mobile banking and several other Android apps and libraries; Java Web-services middleware—including Apache Axis, Axis 2, Codehaus XFire, and Pusher library for Android—and all applications employing this middleware. Any SSL connection from any of these programs is insecure against a man-in-the-middle attack.

Georgiev, Martin, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. "The most dangerous code in the world: validating SSL certificates in non-browser software." In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 38-49. ACM, 2012.

“The root cause of most of these vulnerabilities is the terrible design of the APIs to the underlying SSL libraries”

--The most dangerous code in the world: validating SSL certificates in non-browser software. Martin Georgiev et al., 2012. ACM CCS.

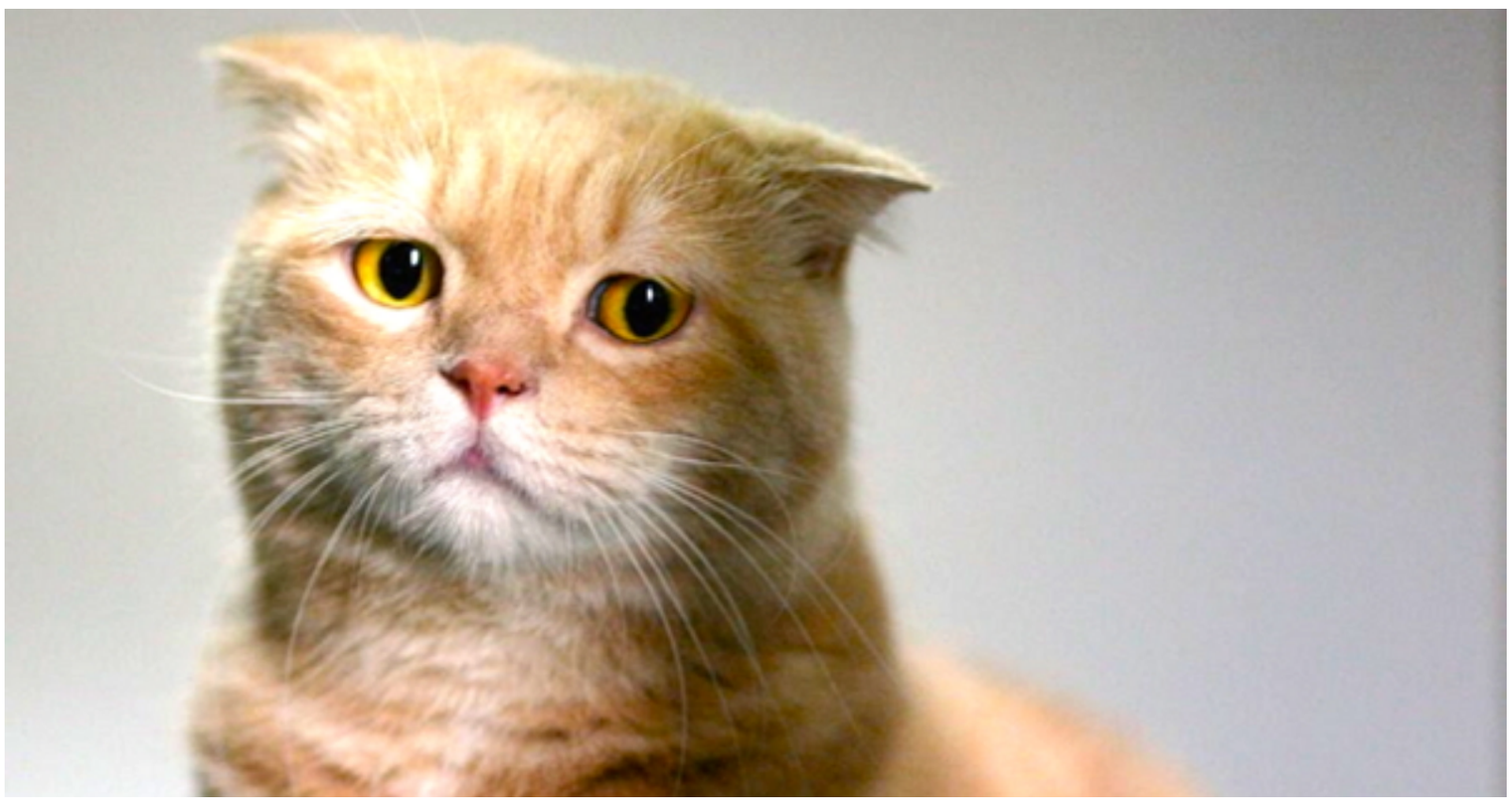
using TLS is hard

Symbols in libssl: 504

`SSL_CTX_set_verify()`

`X509_verify_cert()`

`SSL_CTX_set_cert_verify_callback()`




Activities
Terminal
Thu 14:42
mark@localhost:~/bene-master/examples/openssl_examples/simplessl
File Edit View Search Terminal Help
317
Lines

can we do better?

can we use the POSIX socket API?

can we use the POSIX socket API?

```
int socket = socket(PF_INET, SOCK_STREAM,
                    IPPROTO_TCP);
```

can we use the POSIX socket API?

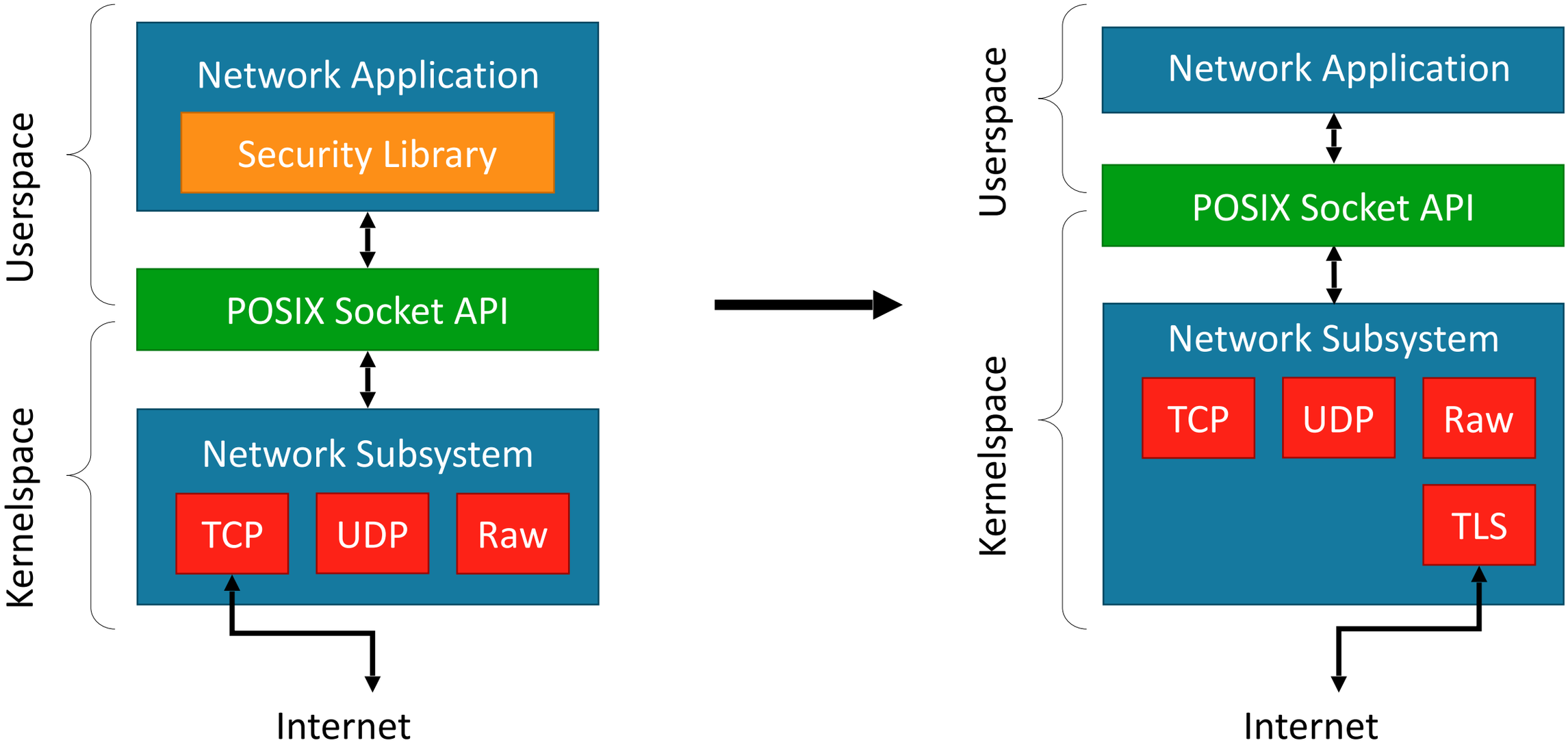
```
int socket = socket(PF_INET, SOCK_STREAM,
                    IPPROTO_TLS);
```

the Secure Socket API (SSA)

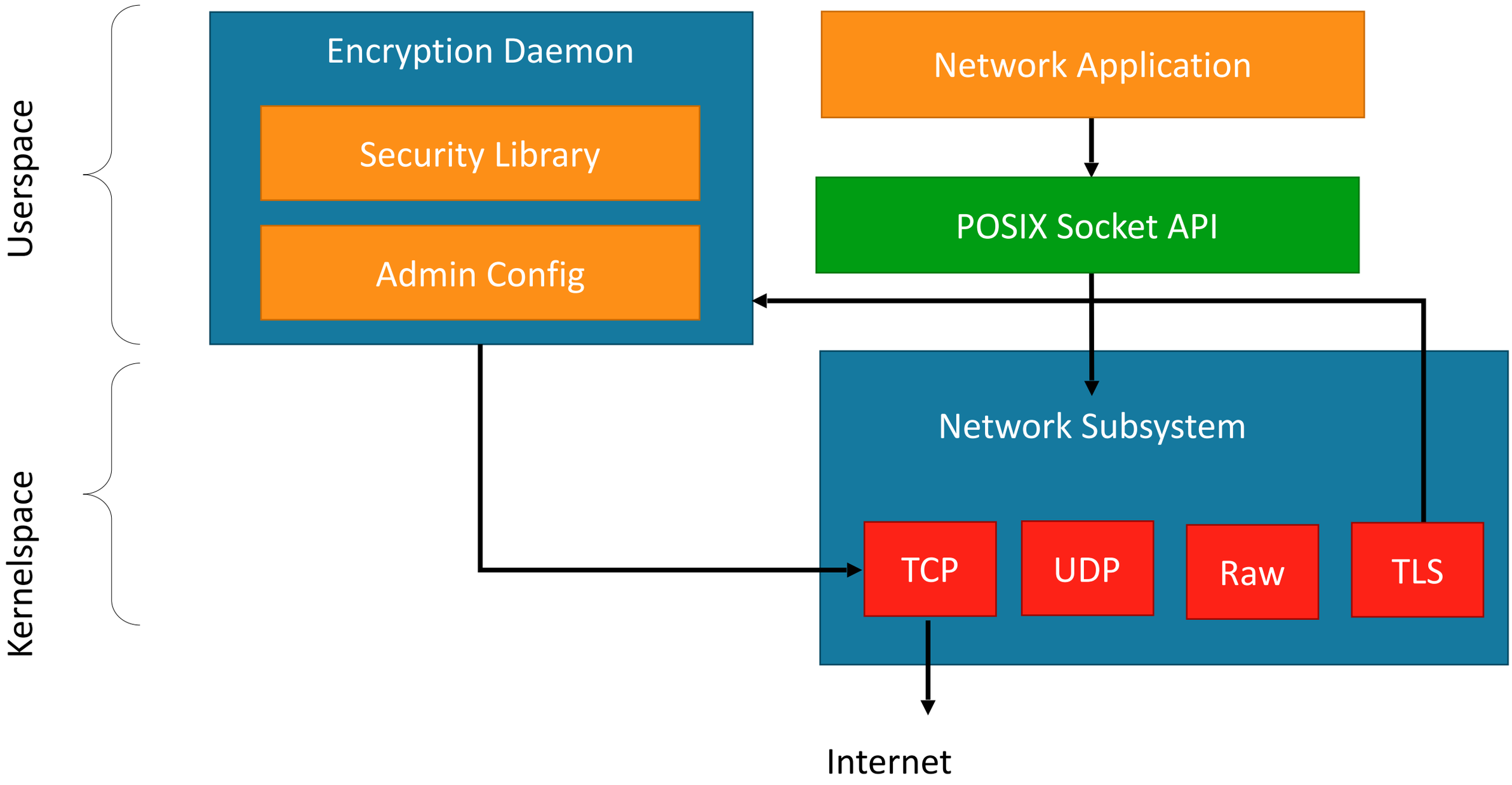
the Secure Socket API (SSA)

`connect`	→	TLS handshake
`send/sendto/ etc.`	→	encrypt and send
`recv/recvfrom/ etc.`	→	decrypt and read
`close`	→	TLS close notify

TLS via the POSIX socket API



Userspace Encryption Daemon



TLS API reduction

OpenSSL
SSL_CTX_new
SSL_CTX_set_verify
SSL_new
SSL_set_fd
TLS_method
SSL_exts_set_hostname
SSL_do_handshake
SSL_set_verify_callback
SSL_get_peer_certificate
And 495 more…

Symbol Count

504 ⟶ **14**

Secure Socket API
socket
bind
listen
connect
setsockopt
getsockopt
close
recv/recvfrom/recvmsg
send/sendto/sendmsg
getaddrinfo

```c
int main() {
        struct sockaddr_host addr;
        addr.sin_family = AF_HOSTNAME;
        strcpy(addr.sin_addr.name, "www.google.com");
        addr.sin_port = htons(443);

        int sock_fd = socket(PF_INET, SOCK_STREAM, IPPROTO_TLS);
        connect(sock_fd, (struct sockaddr*)&addr, sizeof(addr));

        char http_request[] = "GET / HTTP/1.1\r\nHost: www.google.com\r\n\r\n";
        char http_response[2048];
        memset(http_response, 0, 2048);
        send(sock_fd, http_request, sizeof(http_request)-1, 0);
        recv(sock_fd, http_response, 2047, 0);
        close(sock_fd);
        printf("Received:\n%s", http_response);
        return 0;
}
```

reconnaissance

Features	Symbols
version selection	29
cipher suite selection	39
extension management	68
certificate/key management	73
certificate/key validation	51
session management	61
configuration	19
allocation	33
connection management	41
miscellaneous	64
instrumentation	26

analyzed 410 Ubuntu packages that depended on libssl

used developer behavior to guide our design

developer options

```
setsockopt
getsockopt
```

developer options

```
...
fd = socket (PF_INET, SOCK_STREAM, IPPROTO_TLS);
/* Bind to local address and port */
bind (fd, &addr, sizeof(addr));
/* Assign certificate chain */
setsockopt(fd, IPPROTO_TLS, TLS_CERTIFICATE_CHAIN,
CERT_FILE, sizeof(CERT_FILE));
/* Assign private key */
setsockopt(fd, IPPROTO_TLS, TLS_PRIVATE_KEY,
KEY_FILE, sizeof(KEY_FILE));
...
```

developer options

```
setsockopt
getsockopt
```

Option
TLS_REMOTE_HOSTNAME
TLS_HOSTNAME
TLS_TRUSTED_PEER_CERTIFICATES
TLS_CERTIFICATE_CHAIN
TLS_PRIVATE_KEY
TLS_ALPN
TLS_SESSION_TTL
TLS_DISABLE_CIPHER
TLS_PEER_IDENTITY
TLS_PEER_CERTIFICATE_CHAIN

administrator options

- global configuration file assigns TLS defaults
- per-application profiles can further customize settings

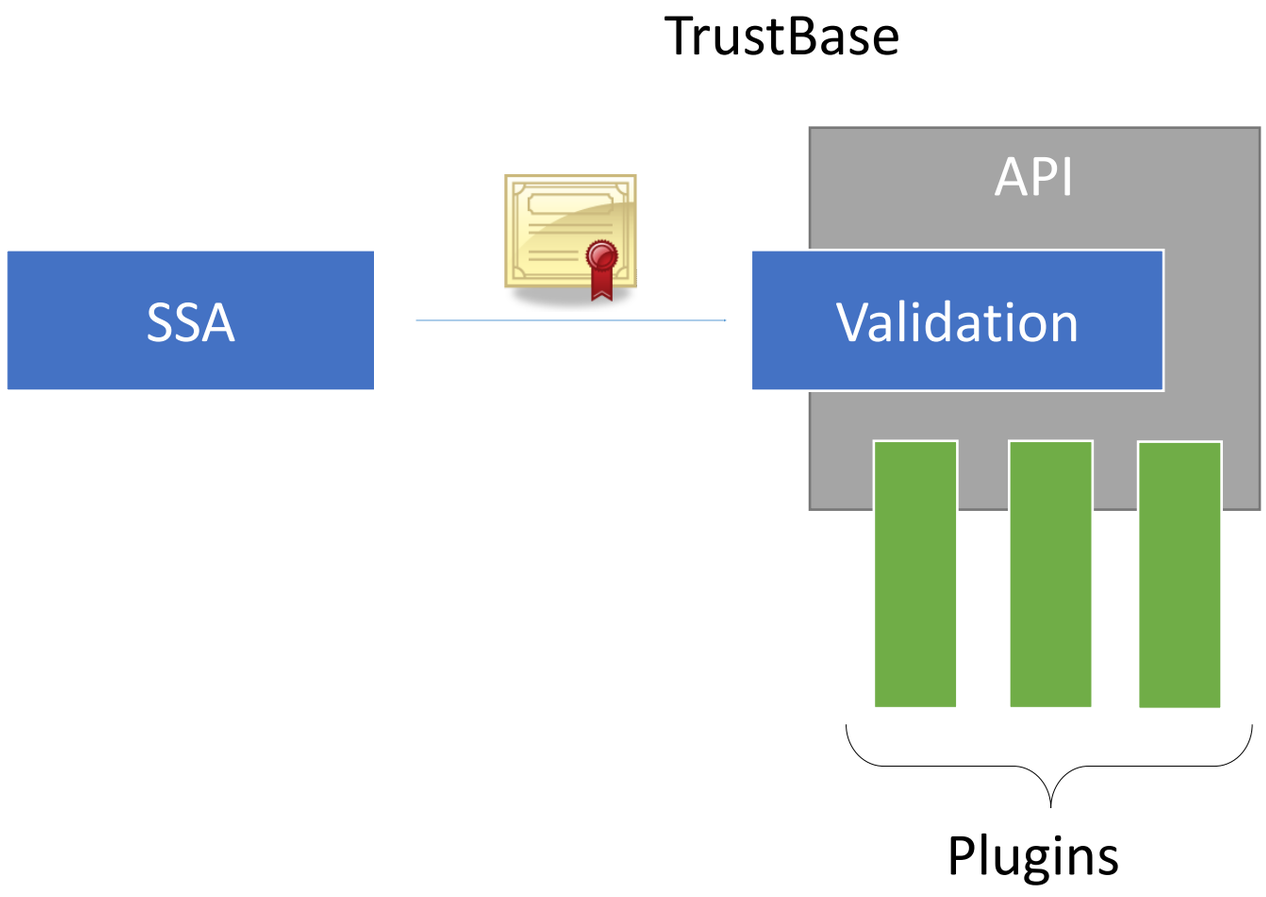
Option	Description
TLS Version	Enabled TLS versions, in order of preference
Cipher Suites	Allowed cipher suites, in order of preference
Certificate Validation	Specified root store for certificate validation, or custom validation engine like TrustBase
Enabled Extensions	Specified TLS extensions to use (e.g., ALPN)
Session Caching	Specified session cache parameters
Default cert/key paths	Specify location of certificates and keys to use when application does not specify

certificate validation

- admin's choice
  - standard validation
  - TrustBase

- TrustBase is an OS service that validates certificates according to admin config

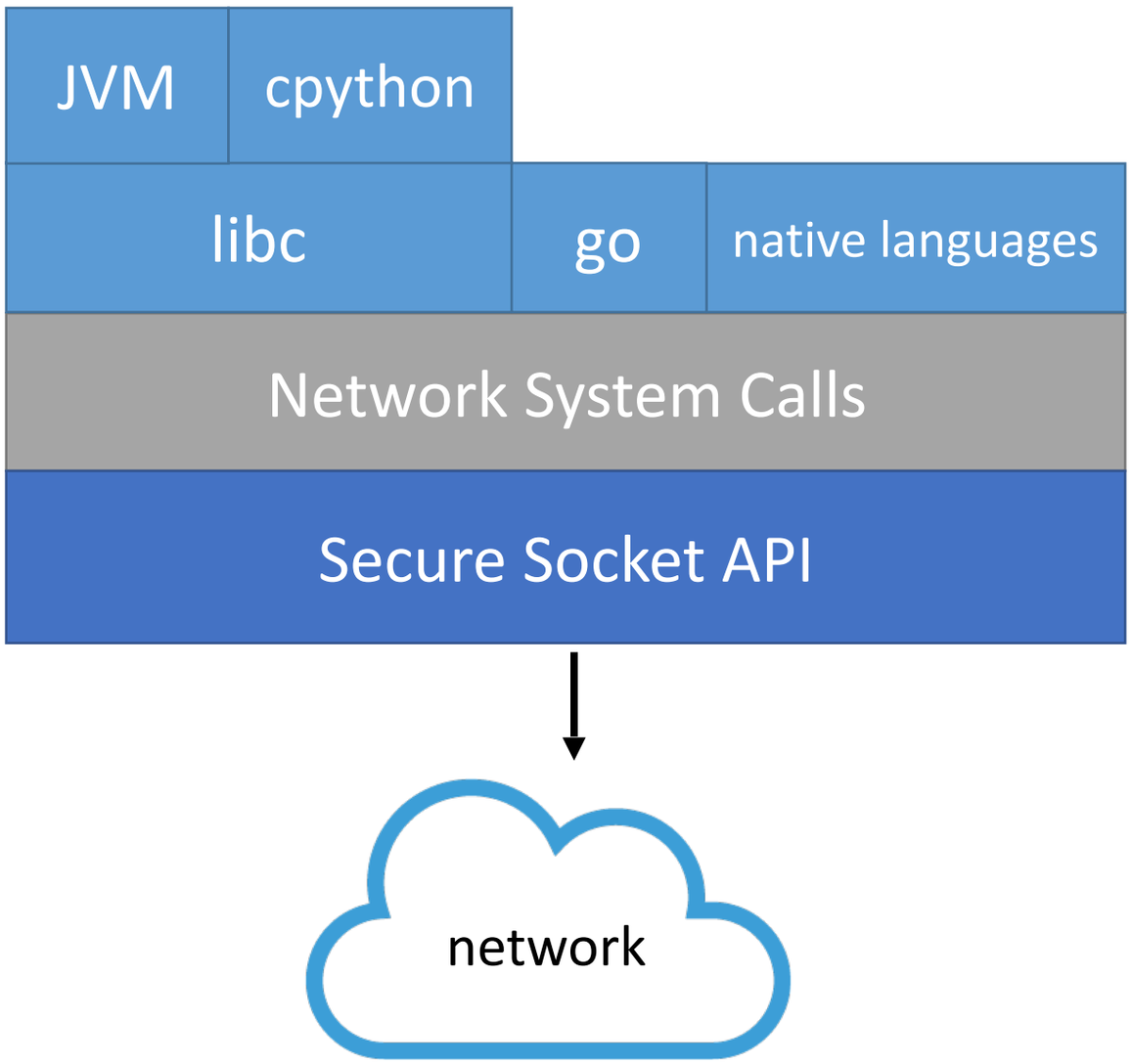
- can enable multiple services (OSCP, CRLsets, custom root stores, Convergence, etc.)



using the SSA

Application	LOC Modified	LOC Removed	Familiar with Code?	Time Taken
Already using TLS				
wget	15	1,020	No	5 Hrs.
lighttpd	8	2,063	No	5 Hrs.
Not using TLS				
in-house webserver	5	0	Yes	5 Min.
netcat	5	0	No	10 Min.

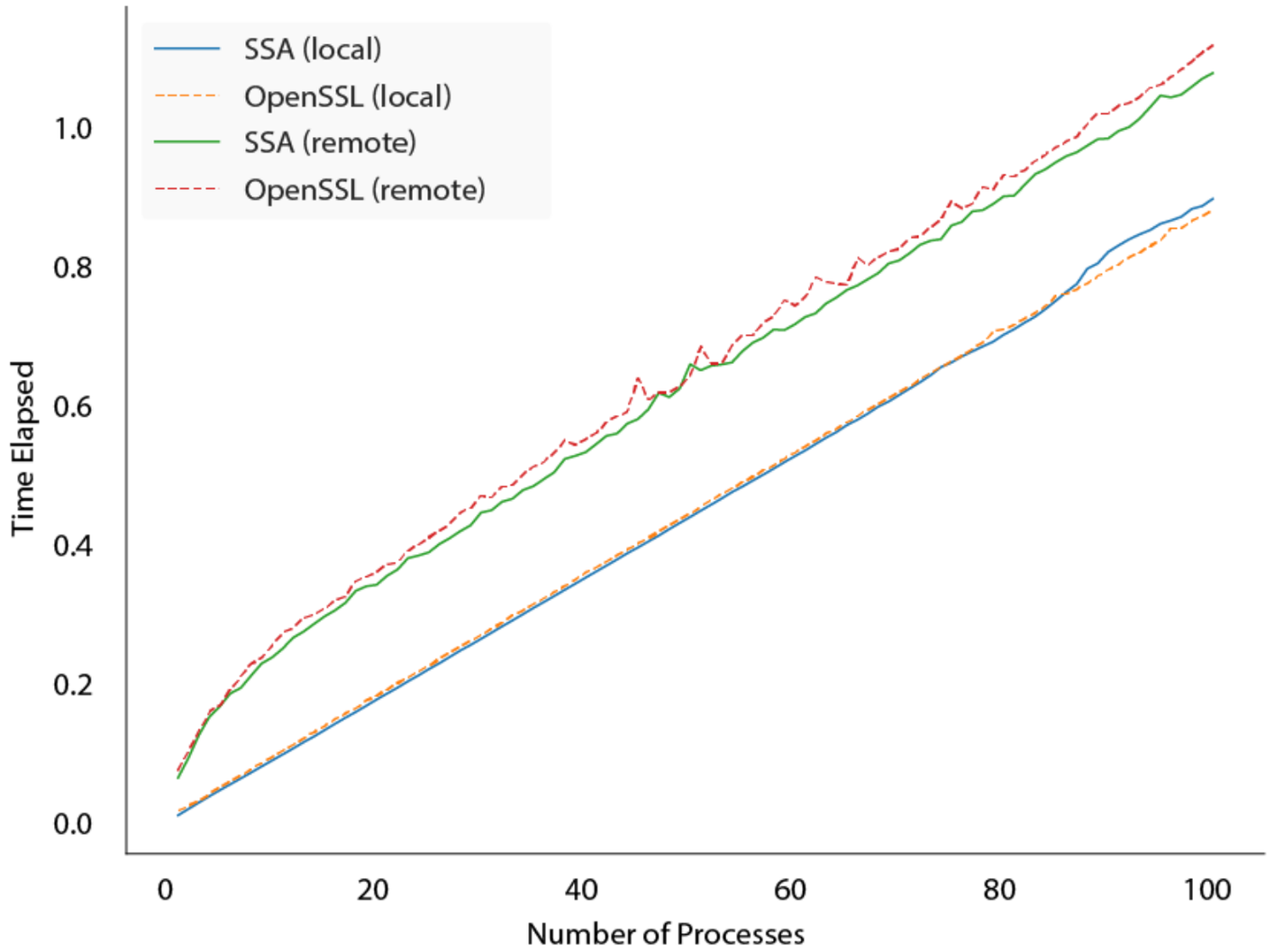
language support



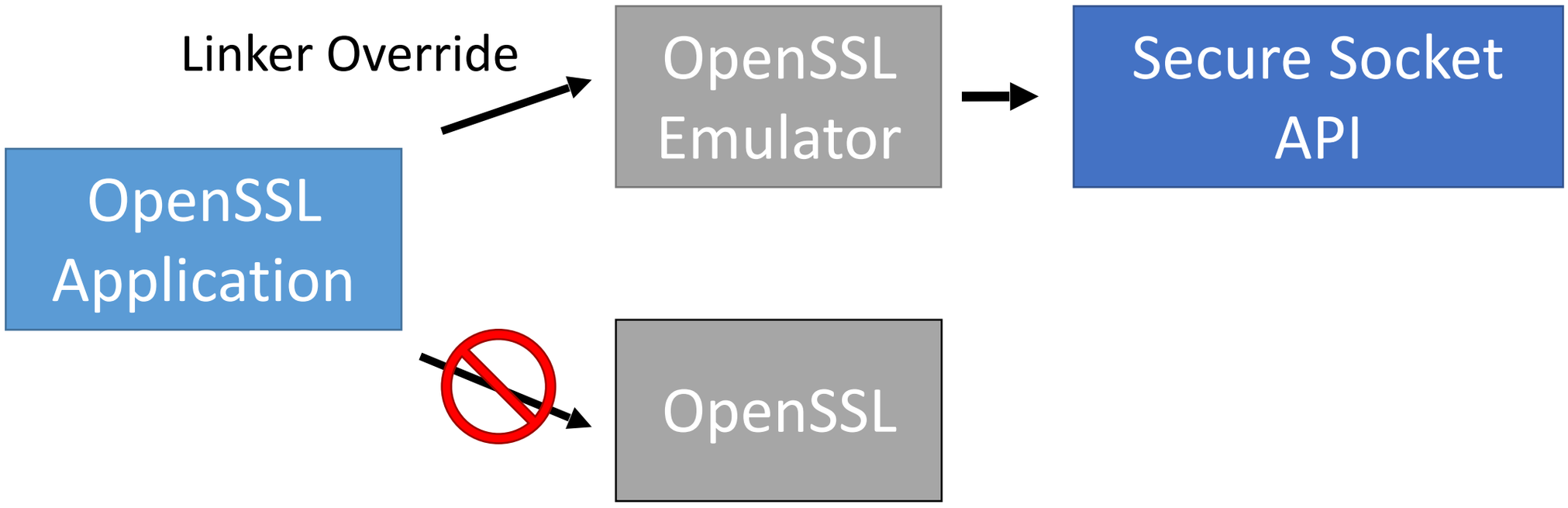
- any language that uses the network uses network system calls (directly or indirectly)
- the SSA is implemented behind the system call layer
- adding SSA support to a language is trivial
  - Go: < 50 lines of code (syscall wrappers)
  - Python: new constants only
  - PHP: new constants only
  - C/C++: new constants only

performance vs OpenSSL

- no discernable time overhead for 0 – 100 concurrent TLS-using processes



broadening coverage



dynamically ported ncat, wget, lighttpd, irssi

outcomes

- general benefits
  - TLS through a known API
  - admin control of TLS settings

- implementation benefits
  - easy language support
  - natural privilege separation
  - alternative implementations supported

the Secure Socket API:

enabling developers to **secure connections**

using a **known API**

in ways **you can control**

Papers

- Mark O'Neill, Scott Heidbrink, Scott Ruoti, Jordan Whitehead, Dan Bunker, Luke Dickinson, Travis Hendershot, Joshua Reynolds, Kent Seamons, and Daniel Zappala, TrustBase: An Architecture to Repair and Strengthen Certificate-based Authentication, *USENIX Security*, August 2017.

- Mark O'Neill, Scott Heidbrink, Jordan Whitehead, Tanner Perdue, Luke Dickinson, Torstein Collett, Nick Bonner, Kent Seamons, and Daniel Zappala, The Secure Socket API: TLS as an Operating System Service, *USENIX Security*, August 2018.

Source Code

- source code: https://github.com/markoneill/trustbase-linux, https://github.com/markoneill/trustbase-windows
- kernel module: https://github.com/markoneill/ssa
- encryption daemon: https://github.com/markoneill/ssa-daemon
- pull requests welcome!
- project website: https://owntrust.org
- contact Mark: mto@byu.edu
- thanks to our sponsors

Homeland Security

Next Steps

- Major tech transfer effort
- Libraries for various languages and OSes that will
  - check for kernel support and use that if available
  - check for an encryption daemon running and connect directly if available
  - worst case, do all the OpenSSL work needed for a secure connection
- Either way, developers use a simple POSIX API, and we try to use centralized policy/control where available