Secured Routines:
Language-based Construction of Trusted Environments

Adrien Ghosn, James R. Larus, Edouard Bugnion
EPFL, Switzerland
The solution to the problem of trust in the Cloud.

Confidentiality & Integrity.

Intel SGX enclaves.

... SGX is pretty hard to use!
TEE-Support Challenges

- Performance
  - TCB size
  -.syscalls
  - Coop
  - Partition
- Security
  - secret leak
- Usability
  - legacy apps
  - crossings

Partition Coop
Guess what…

... A compiler can do that for you! *

*(given the right programming abstraction)*
Secured Routines

A language-level approach for TEEs.

A familiar programming abstraction.

Implemented **GOTEE**, our fork of the Go compiler.

Adding a single keyword, `gosecure`...
func main() {
    go foo()
    ...}

Go Runtime
func main() {
    gosecure foo()
    ...
}

Go Runtime

Trusted Runtime

Go channel
GOTEE

- **Compile-time**
  - Minimal TCB
  - Automatic code & data partitioning
  - Cross-domain channels

- **Run-time**
  - Memory Isolation
  - TCB size
  - Coop
  - Partition
  - Crossings

- **Security**
  - Memory leaks
  - Transparent syscalls

- **Performance**
  - Crossings

- **Usability**
  - Legacy apps

- **Transparent syscalls**
  - Runtime Cooperation
  - Minimal code changes
@Compile-time

Security
- memory leaks

Performance
- TCB size
- crossings

Usability
- Syscalls
- Coop
- Partition
- legacy apps

Minimal TCB
Automatic code & data partitioning

Minimal code changes
GOTEE compile-time overview

- App.go
- Parse
- Gosec
- Runtime ext.
- Compile/Link untrusted
- Compile/Link trusted
- Enclave_entry.go
- Untrusted code & data
- Package ELF
- Trusted code & data
- Gosecure calls
- /App
...

```go
package pkg

func foo(a, b, c) {
    fmt.Println("Called the enclave")
}
```

...
Final executable

```
./app
.text
.data
.rodata
.enclave
```

```
.text
.data
.rodata
.enclave
```
GOTEE run-time overview

Diagram showing the flow of untrusted and trusted applications, runtime, syscalls, and gosecure servers. Threads and goroutines are also depicted with channels for cross-domain communication.
These are Go typed & synchronized channels extended by GOTEET.

Automatic deep-copy for cross-domain communication.

Single point of interaction between the domains…

Get rid of crossings!
Deep-copy

No cross-domain memory references.

Similar to network marshalling.

Implemented via reflection.

Independent GCs & enhance memory isolation!
@Run-time(2)

Performance

Security

Usability

Memory Isolation

Transparent syscalls

Runtime Cooperation

memory leaks

Syscalls

TCB size

Coop

Partition

crossings

cross-domain channels

legacy apps

@Run-time(2)
Security
- memory leaks
- TCB size
- Syscalls
- Coop
- Partition
- legacy apps

Performance
- TCB size
- Syscalls
- Crossings

Usability
- Coop
- Partition
- legacy apps

Memory Isolation
- Cross-domain channels

Runtime Cooperation
- Transparent syscalls
In Vanilla Go

1. `chan A` write(read) 1 block
2. `read(write)` unblocks

`chan A blocked queue`

`scheduler queue`
In GOTEE

write(read)

chan A

blocks

1

1

read(write)

chan A blocked queue

s

polls

schedules

1

checks &
unblocks

scheduler queue

TrustedReadyQueue

(Trusted) scheduler queue
Evaluation

Micro-benchmarks.

Macro-benchmarks.

Code & Data sizes.
## Evaluation @micro-benchmarks

<table>
<thead>
<tr>
<th></th>
<th>Go</th>
<th>GOTEE</th>
<th>Intel SGX SDK</th>
<th>Ratios GOTEE/SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syscall (µs)</td>
<td>0.23</td>
<td>1.35</td>
<td>3.69</td>
<td><strong>2.7x</strong></td>
</tr>
<tr>
<td>Gosecure+block (µs)</td>
<td>0.30</td>
<td>1.5</td>
<td>3.50</td>
<td><strong>2.3x</strong></td>
</tr>
<tr>
<td>Throughput (KOPS/trusted thread)</td>
<td>6000</td>
<td>1460</td>
<td>281</td>
<td><strong>5.2x</strong></td>
</tr>
</tbody>
</table>
Trusted SSH-server

Everything in the enclave, no application-level code modification.

Fine-grained TLS

Private key only available inside the enclave.

9 LOC in tls and 35 LOC of new code in enclcert.

88% of native throughput (handshakes).

Go-ethereum trusted keystore

1 day, 500 lines of code.
### Evaluation @Code & Data sizes

<table>
<thead>
<tr>
<th></th>
<th>TCB (KB)</th>
<th>Pkg Deps</th>
<th>Application LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtime GOTEY</td>
<td>793</td>
<td>runtime</td>
<td>-</td>
</tr>
<tr>
<td>Hello World</td>
<td>884</td>
<td>++ fmt, syscall, io, unicode...</td>
<td>13</td>
</tr>
<tr>
<td>SSH-server</td>
<td>2437</td>
<td>++crypto, golang.org/crypto/*,</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>net, encoding...</td>
<td></td>
</tr>
<tr>
<td>Keystore</td>
<td>3936</td>
<td>++crypto/ecdsa, crypto/elliptic,</td>
<td>474</td>
</tr>
<tr>
<td></td>
<td></td>
<td>crypto/aes...</td>
<td></td>
</tr>
<tr>
<td>SGX SDK runtime</td>
<td>75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SGX SDK Hello</td>
<td>166</td>
<td>-</td>
<td>355</td>
</tr>
</tbody>
</table>
This was GOTEE. Merci Beaucoup!