DARTMOUTH

Work in Progress: On Session Languages

Prashant Anantharaman, *Sean W. Smith*
Dartmouth College, NH, USA

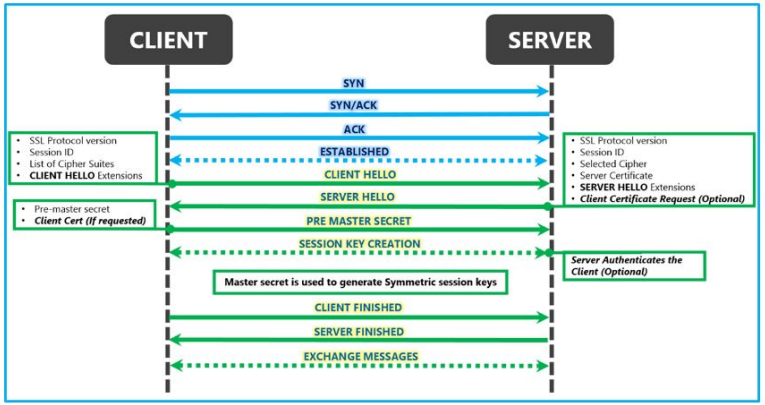
pa@cs.dartmouth.edu / *https://prashant.at*

What are Session Languages?

- The spirit of LangSec is to use formal tools to tighten up input validation.
- Formal grammars are often static, and what constitutes “well-formed” input can change over the execution lifetime for network protocols.
- What are the right tools to tackle these languages that change, i.e., Session Languages?

Why Session Languages?

- Communication protocols support various messages: and the sender and the receiver usually need to keep a finite-state machine.
- Most protocols specify what the correct flow is, but do not specify what happens when certain arbitrary sequence of messages appear.



Source: www.bogotobogo.com

Outline

In this talk we'll discuss some approaches to tackle this problem:

- Session Types,
- Register Automata,
- Sequences of Languages,
- and, Three-tiered Grammars.

Session Types

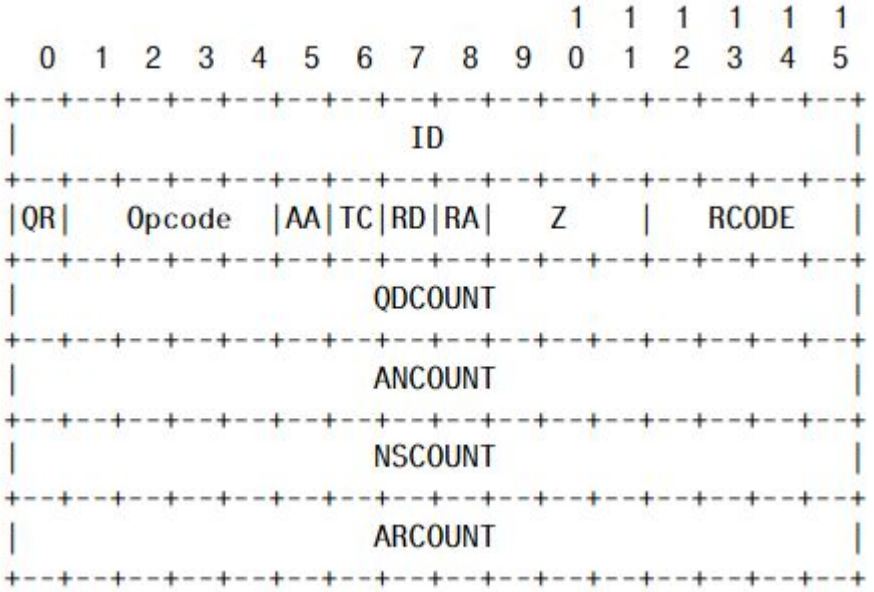
- Session types use typed π-calculi, and are basically “types for communication protocols.”
- They provide a clear sequence of messages for sender and receiver along with the types to match for the sender and receiver.
- However, when a party receives the message, what if they need to alter their grammar based on what they received?

```
global protocol OnlineWallet
(role S, role C, role A) {
        login(id:string , pw:string) from C to A;
        choice at A {
                login_ok () from A to C, S;
                rec LOOP {
                account(balance:int ,overdraft:int) from S to C;
                choice at C {
                @<amount <= balance+overdraft >
                pay(payee:string , amount:int) from C to S;
                continue LOOP;
                } or {
                quit() from C to S; }}
                } or {
                login_fail(error:string)from A to C, S;
}}
```

Source: Neykova et al. “SPY: Local Verification of Global Protocols”

DNS Example

- Sender needs to ensure that the answers are related to the number of questions they asked.
- The type to match the DNS response, depends on certain values in the DNS request.



Register Automata

A register automata (RA) is represented as a 6-tuple (R, Q, q_0, v_0, F , Δ).

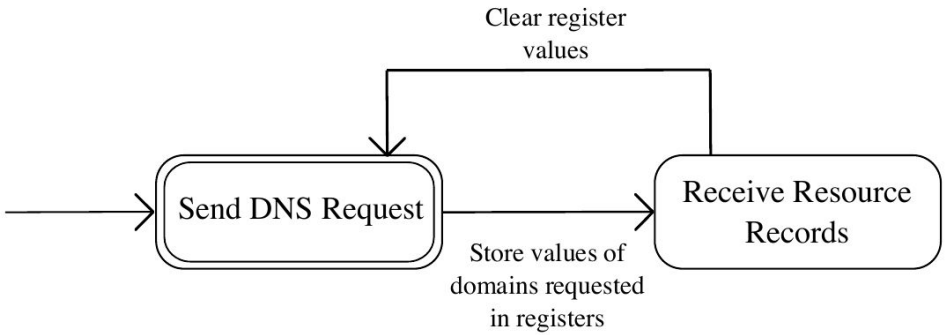
- R is a finite set of registers,
- Q is a finite set of states,
- $q_0 \subseteq Q$ is the start state,
- v_0 is the initial assignment of the registers in R.
  - They use the registers allows you to check for equality and inequality conditions.

Symbolic Register Automata

- Transitions are defined on first-order predicates, instead of specifying individual symbols.
- Reduces the number of states needed drastically.

DNS Example with Register Automata

```
<transitions>
        <transition from="send_dns_request"
params="src_ip, dst_ip,domain_list"
symbol="dns_req" to="receive_resource_records">
        <guard>
        dst_ip==dnsserver && src_ip==client
        </guard>
        <assignments>
        <assign to="requested_domain_values">
                domain_list
        </assign>
        </assignments>
  </transition>
```



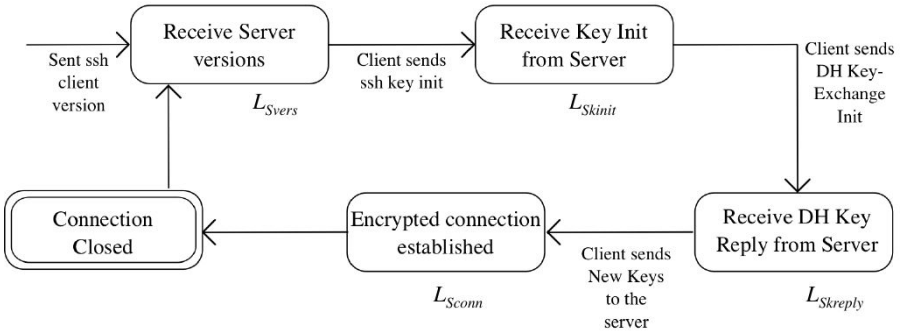
http://automata.cs.ru.nl/Syntax/Register#Registerautomatamodel provides an XML syntax to describe register automata.

Sequences of Languages

- Let C_1 and C_2 be classes of languages.
- We define C_1 / C_2 to be the class of session languages

$$\{L_{seq} / L_{ext} : L_{seq} \in C1 \text{ and } L_{ext} \subseteq C2\}$$

- The session language on the right could be represented as the sequence of the following languages $L_{Sver}\ L_{Skinit}\ L_{Skreply}\ L^*_{Sconn}$



Three Layered Grammars: Languages with Internal Actions

- We can use such a formalism to argue about session languages that need to check some values such as register values in DNS.
- L_{int} describes the internal actions of the software.
- Each string in L_{int} can influence what L_{ext} is going to be.

$$\{L_{seq}/L_{ext}/L_{int} : L_{seq} \in C_1$$
$$\text{and } L_{ext} \subseteq C_2$$
$$\text{and } L_{int} \subseteq C_3\}$$

Ongoing work

- Exploring π-calculi-based languages to describe protocols including semantic actions. How do we use the work done in the session type domain to support the concept of layered grammars?

- How do we use session types to describe protocols as symbolic register automata?

Questions?

Prashant Anantharaman, pa@cs.dartmouth.edu
Sean W. Smith, sws@cs.dartmouth.edu

Tech Report Available here:
https://www.cs.dartmouth.edu/~sws/pubs/TR2020-881.pdf