Hardware, Languages, and Architectures for Defense Against Hostile Operating Systems

Vikram Adve, Krste Asanović, David Evans, Sam King, Greg Morrisett, R. Sekar, Dawn Song, David Wagner (PI)

Vikram Adve
(UIUC)

Krste Asanović
(UC Berkeley)

David Evans
(U Virginia)

Sam King
(UIUC)

Greg Morrisett
(Harvard)

R. Sekar
(Stony Brook)

Dawn Song
(UC Berkeley)

David Wagner
(UC Berkeley)

My Goals for Today

- Brief you on our planned approach
- Solicit your feedback on our plans
- Understand your needs, problem domain, and threat model better
- Build personal connections

Agenda

9:00- 9:30	Welcome + Overview (Wagner)
9:30- 9:45	Secure Virtual Architecture (Adve)
9:45-10:00	Architectures for secure systems (Evans)
10:00-10:15	Binary analysis (McCamant)
10:45-11:00	Binary translation (Sekar)
11:00-11:15	Formal verification (Chlipala)
11:15-11:30	Secure browsers, hardware (King)
11:30-11:45	Secure hardware (Asanović)
11:45-12:15	Discussion and feedback
12:15- 1:15	Lunch

Problem Statement

Defend against a compromised, hostile, or malicious operating system.

Today: If the OS is malicious, all is lost.

Desired end state: We can survive a malicious OS, perhaps with degraded functionality or availability.

Problem Statement

Defend against a compromised, hostile, or malicious operating system.

We want to:

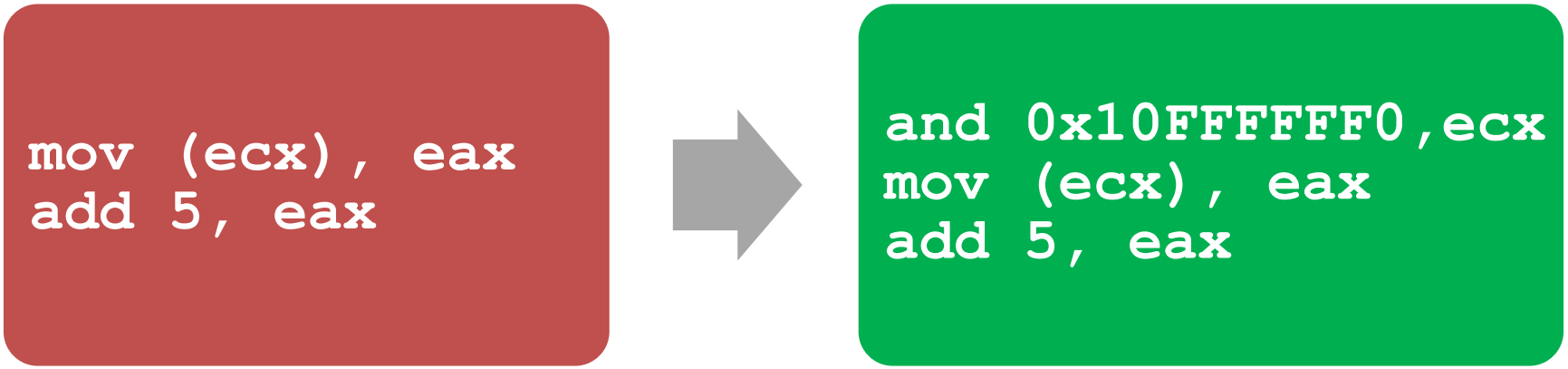
- Protect legacy software
- Facilitate building secure systems of the future

Theme 1: Restrict the OS

We'll enforce end-to-end security policies by instrumenting the OS, using:

- Binary translation:
  when we only have code in binary form
- Secure Virtual Architecture:
  when we have source code

Binary Translation

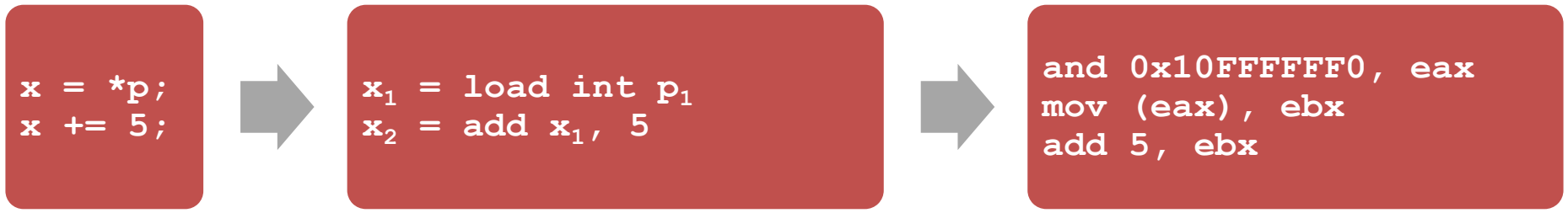


x86 code
(potentially hostile)

x86 code
(neutered)

Instrument binary OS code to enforce security policies (e.g., end-to-end information flow properties; memory safety; isolation).

Secure Virtual Architecture (SVA)

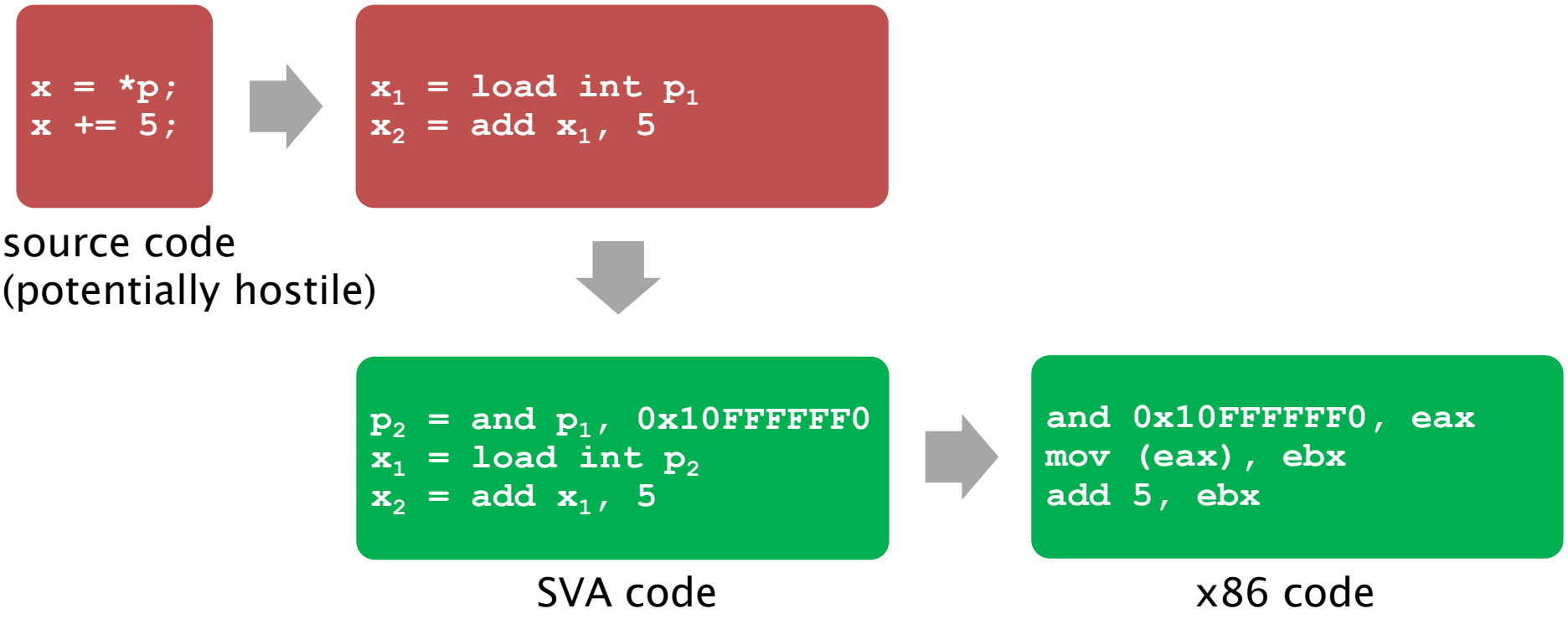


source code
(potentially hostile)

SVA code
(potentially hostile)

x86 code
(potentially hostile)

Secure Virtual Architecture (SVA)

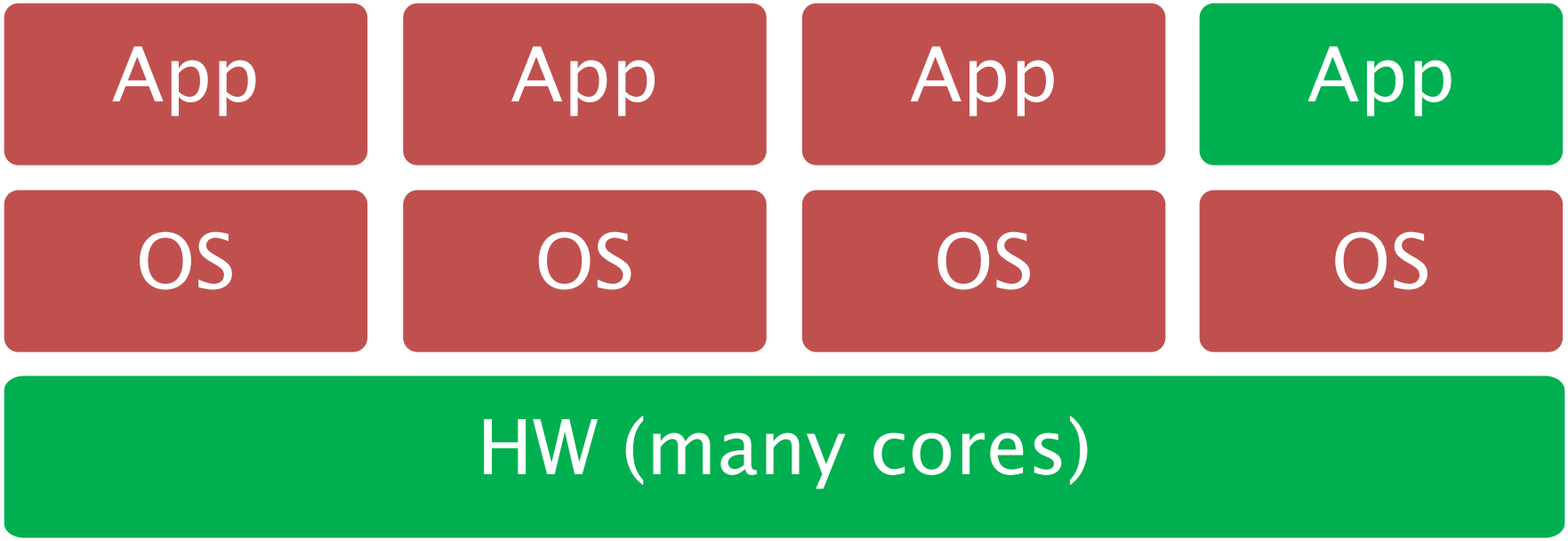


SVA facilitates instrumentation, static analysis , rewriting, virtualization, memory safety, …

Theme 2: Minimize Reliance upon the OS

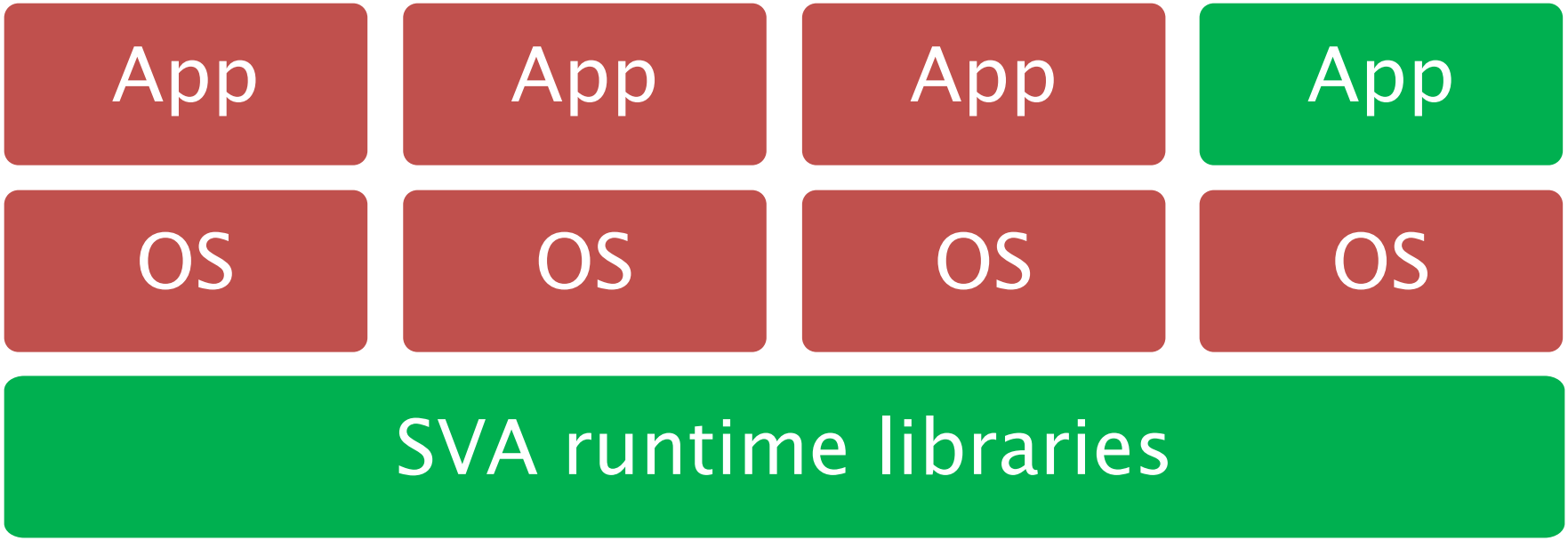
- Isolate OS instances to contain the impact of OS compromise, using:
  - Secure hardware, or
  - Virtualization (Secure Virtual Architecture)
- Build single-purpose "appliances"
  - Don't need a general-purpose OS
    $\Rightarrow$ eliminates a class of risks

Secure Hardware



Partition untrusted OS instances across multiple cores, with hardware isolation.

Virtualization

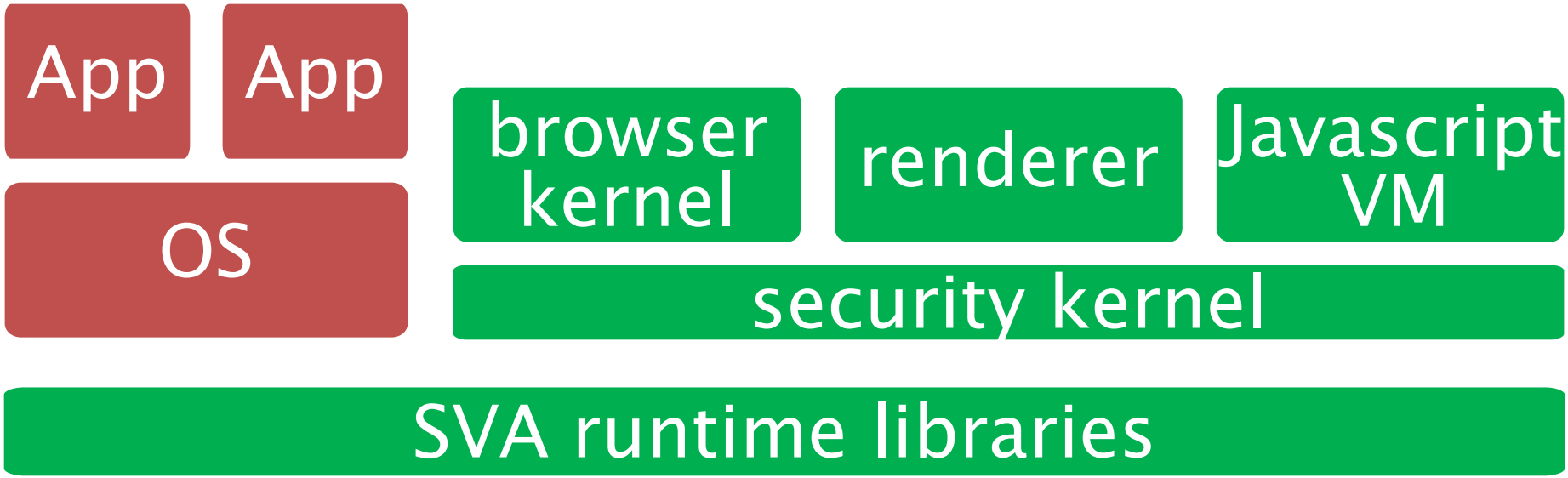


Use SVA to virtualize and isolate multiple OS instances.

Secure Browser Appliance

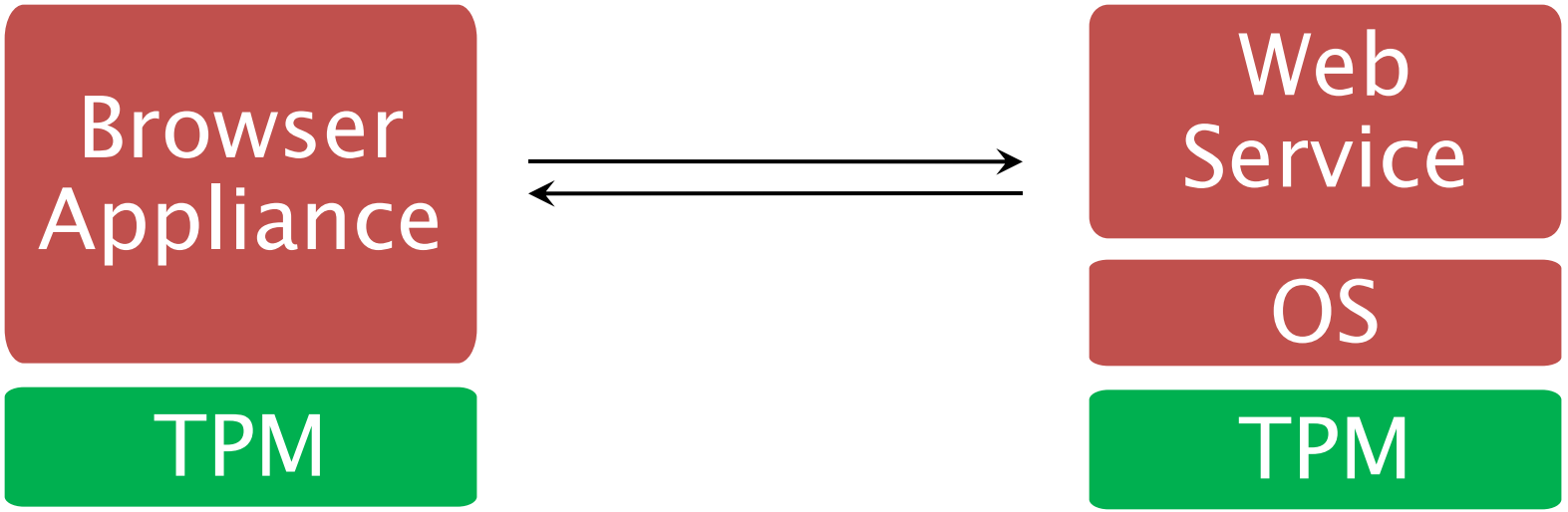
Build a single-purpose appliance, for secure web browsing, with no OS.

Secure Browser Appliance



"Virtual browser appliance" can co-exist securely with insecure OS environment.

Secure Web Systems



New security architecture:

- UI code runs on client (HTML/CSS/Javascript).
- Application logic runs on server, protected with SVA/hardware/safe languages.

Data-Oriented Computing

A new security architecture:

- Users may attach a security policy to each discrete piece of data.
- Goal: Ensure that entire system respects this policy, and that policy is inseparable from data.
  - Mechanism: binary translation, SVA, or other tools

Be Aware

- We are exploring many approaches to the problem. We do not know which will prove most effective. Some may fail.
- We hope some of our ideas will have applications to other security problems outside of the hostile OS problem.

Agenda

9:00- 9:30	Welcome + Overview (Wagner)
9:30- 9:45	Secure Virtual Architecture (Adve)
9:45-10:00	Architectures for secure systems (Evans)
10:00-10:15	Binary analysis (McCamant)
10:45-11:00	Binary translation (Sekar)
11:00-11:15	Formal verification (Chlipala)
11:15-11:30	Secure browsers, hardware (King)
11:30-11:45	Secure hardware (Asanović)
11:45-12:15	Discussion and feedback
12:15- 1:15	Lunch