From Multiple Credentials to Browser-based Single Sign-On: Are We More Secure?

Alessandro Armando Roberto Carbone
Luca Compagna Jorge Cuellar
Giancarlo Pellegrino Alessandro Sorniotti

Artificial Intelligence Laboratory (AI-Lab)
DIST, University of Genova
Genova

Security & Trust Research Unit
FBK-IRST
Trento

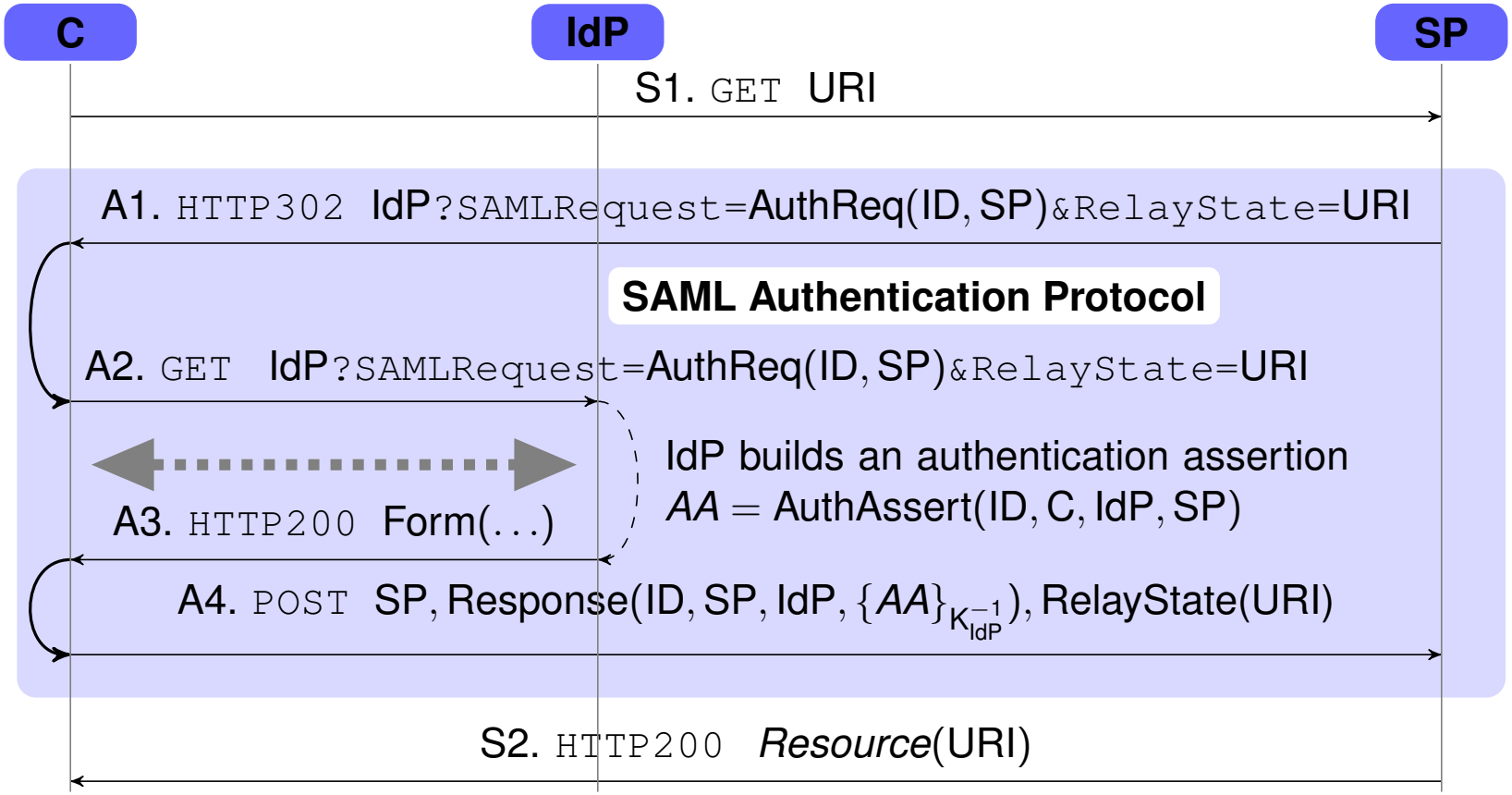
From Multiple Credentials to Browser-based SSO

- Improved user experience
- Security?
  - Pros: SSO mitigates risks of password guessing attacks
  - Cons: any? Yes, there are. :-(
- Plan of the talk:
  - model and analysis of SAML 2.0 Web Browser SSO Profile
  - authentication vulnerability in the prototypical SAML SSO use case
  - exploitations in actual deployments
  - some mitigations and solutions

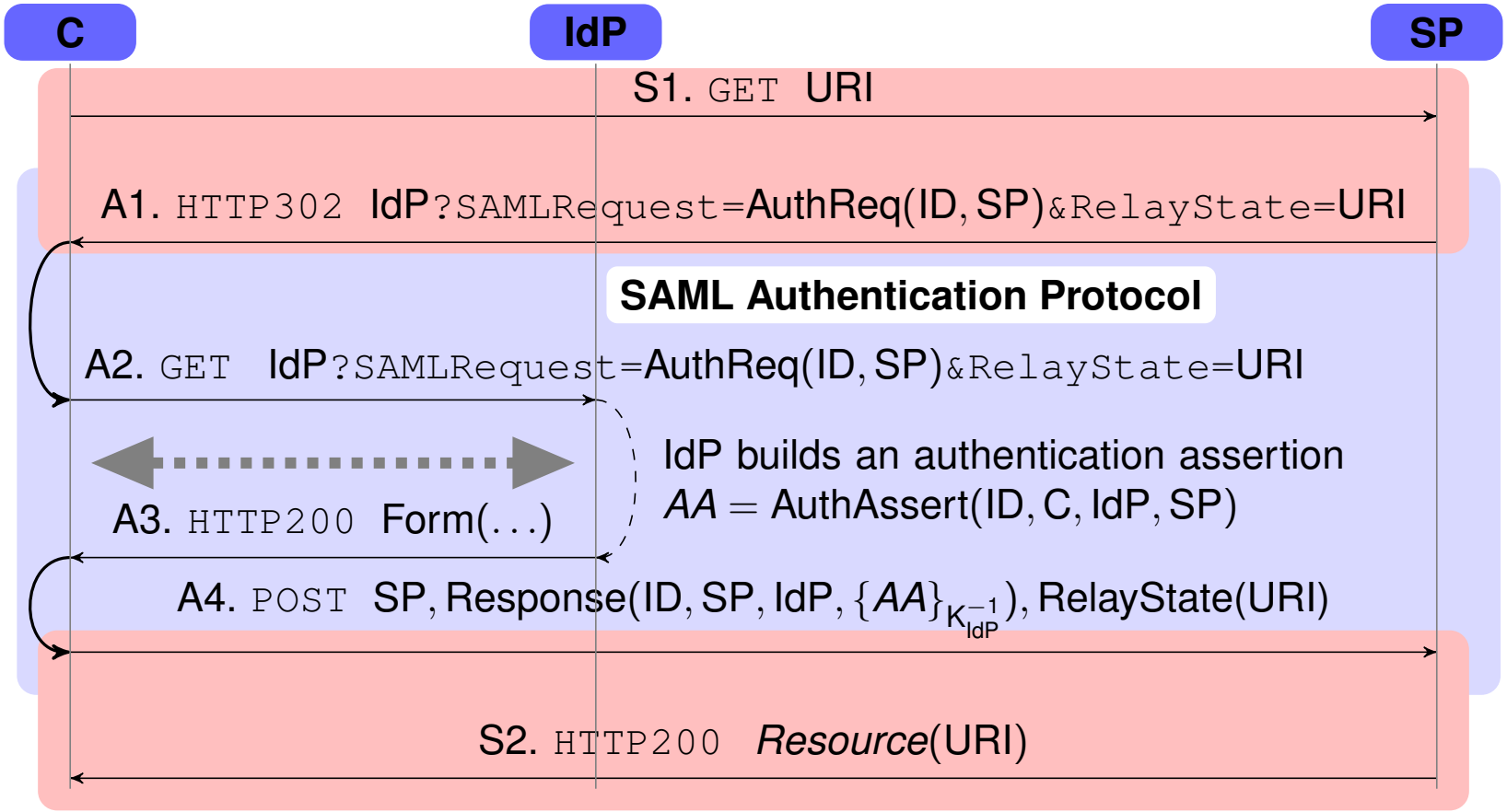
The SAML 2.0 Web Browser SSO Profile (SAML SSO)

- The OASIS *Security Assertion Markup Language* (SAML) 2.0 Web Browser SSO Profile (SAML SSO) is an emerging standard:
  - it defines an XML-based format for security assertions and
  - protocols and bindings prescribe how assertions must be exchanged in a variety of deployment scenarios.
- Supported by the vast majority of Identity and Access Management Systems, e.g. IBM Tivoli Access Manager, Novell Access Manager, Shibboleth, SAP NG SSO.
- Used by prominent software companies, e.g. the SAML-based SSO for Google Apps.

The SAML 2.0 Web Browser SSO Profile (SAML SSO)



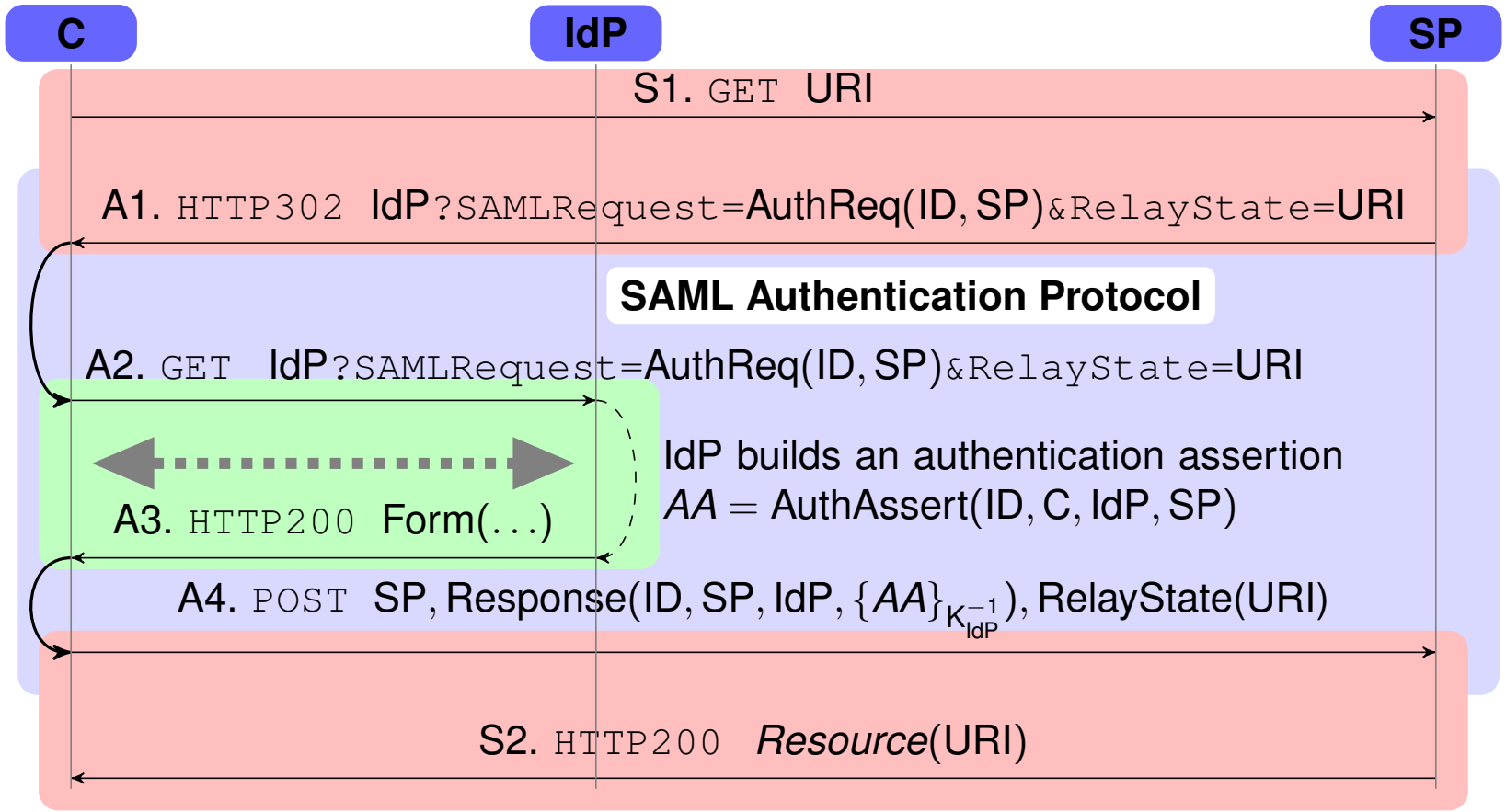
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Assumption on Transport Protocols (TP1)

Communication between C and SP is carried over a unilateral SSL/TLS channel.

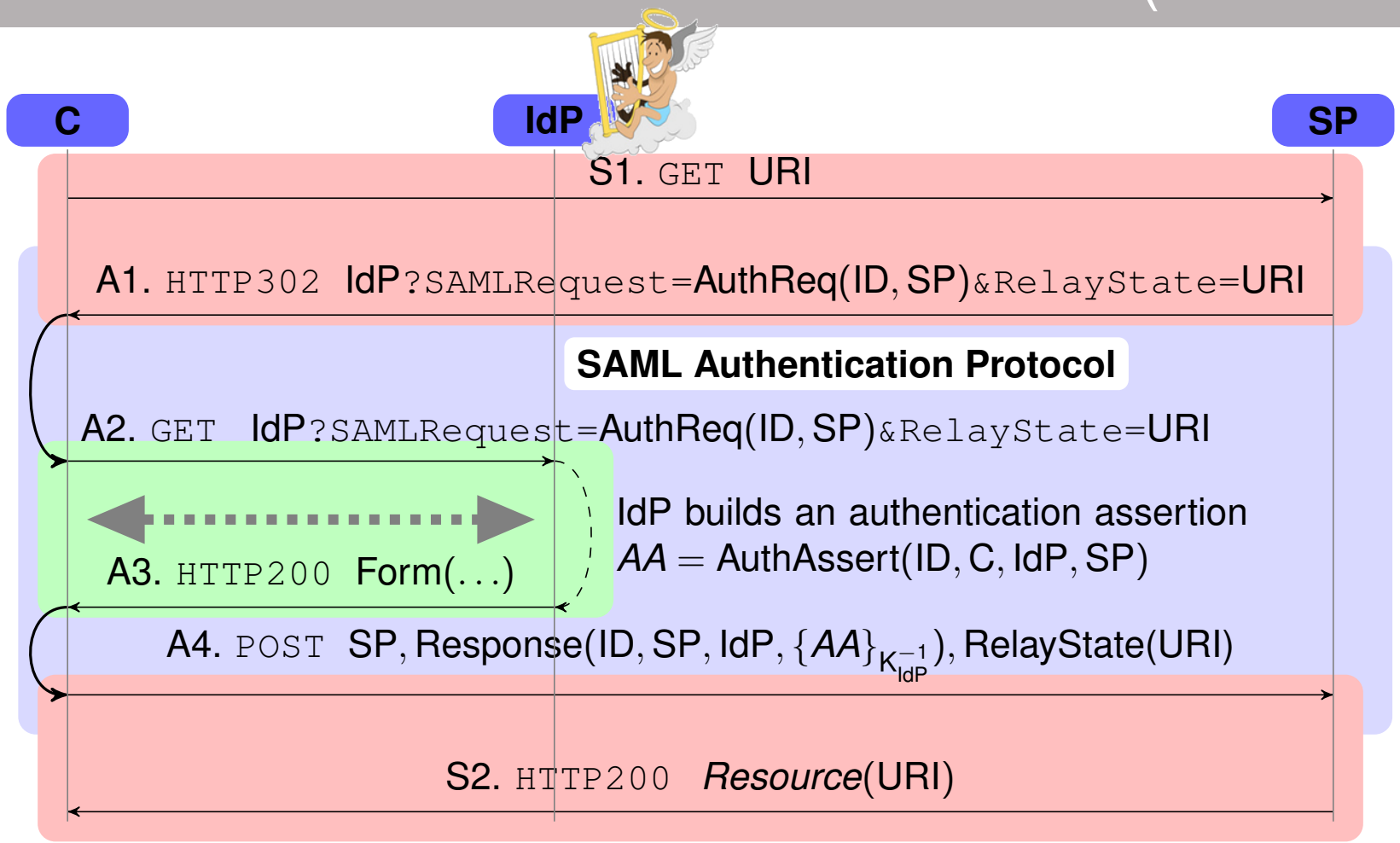
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Assumption on Transport Protocols (TP2)

Communication between C and IdP is carried over a unilateral SSL/TLS channel that becomes bilateral once C authenticates on IdP.

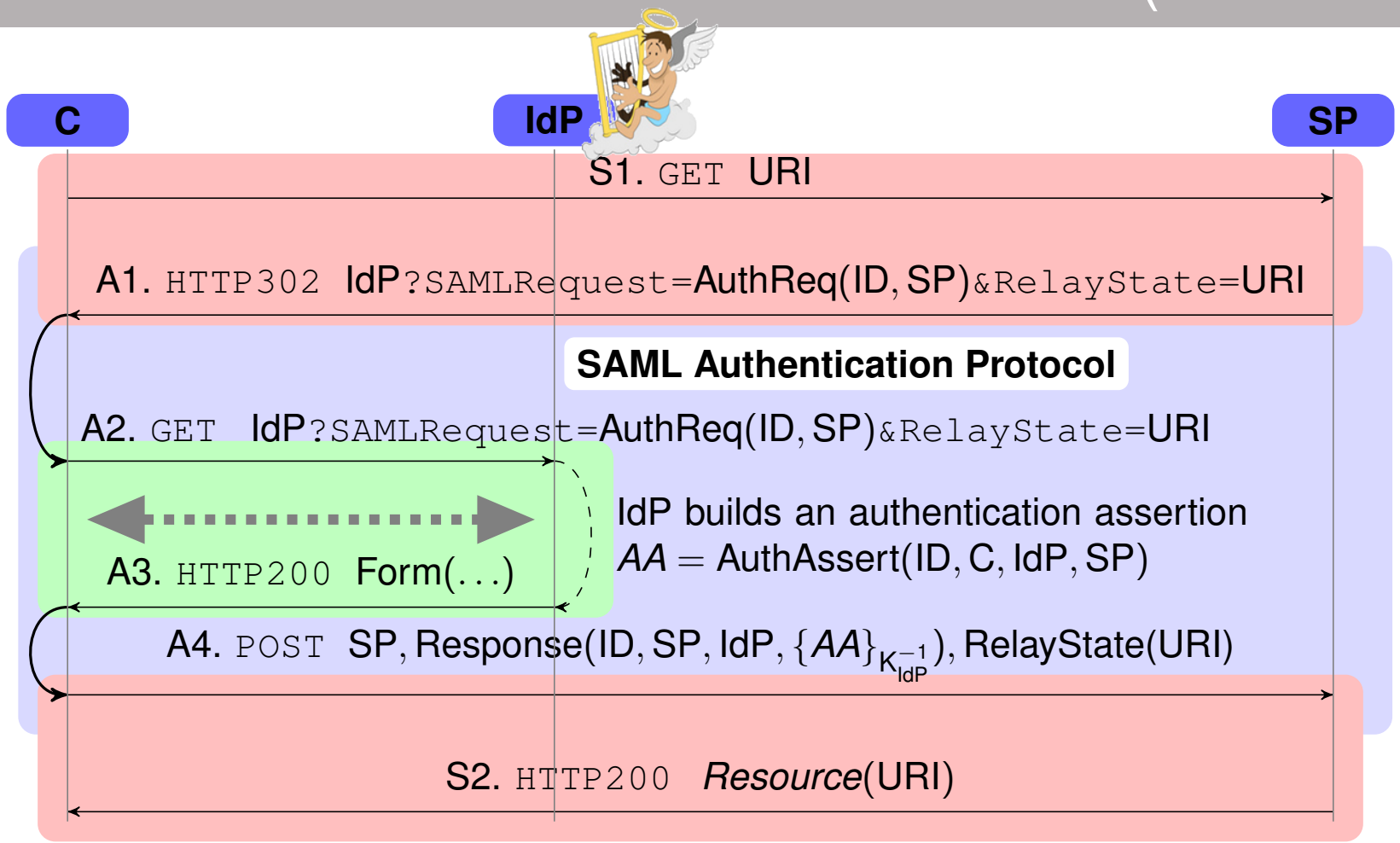
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Trust Assumption (TA1)

IdP is not compromised, i.e. it is not under the control of an intruder and it abides by the rules of the protocol.

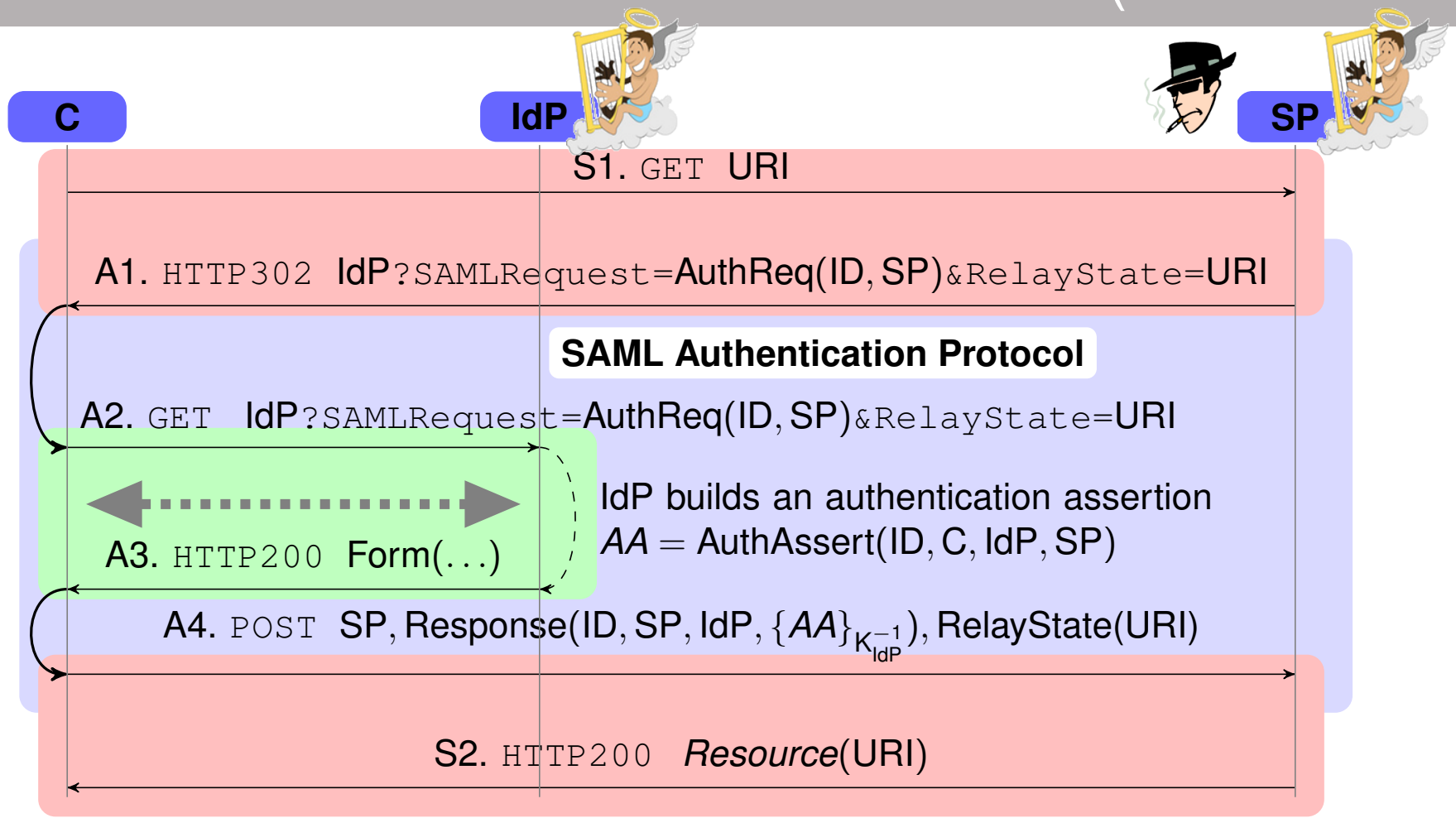
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Trust Assumption (TA2)

IdP is trusted by SP to generate authentication assertions about C.

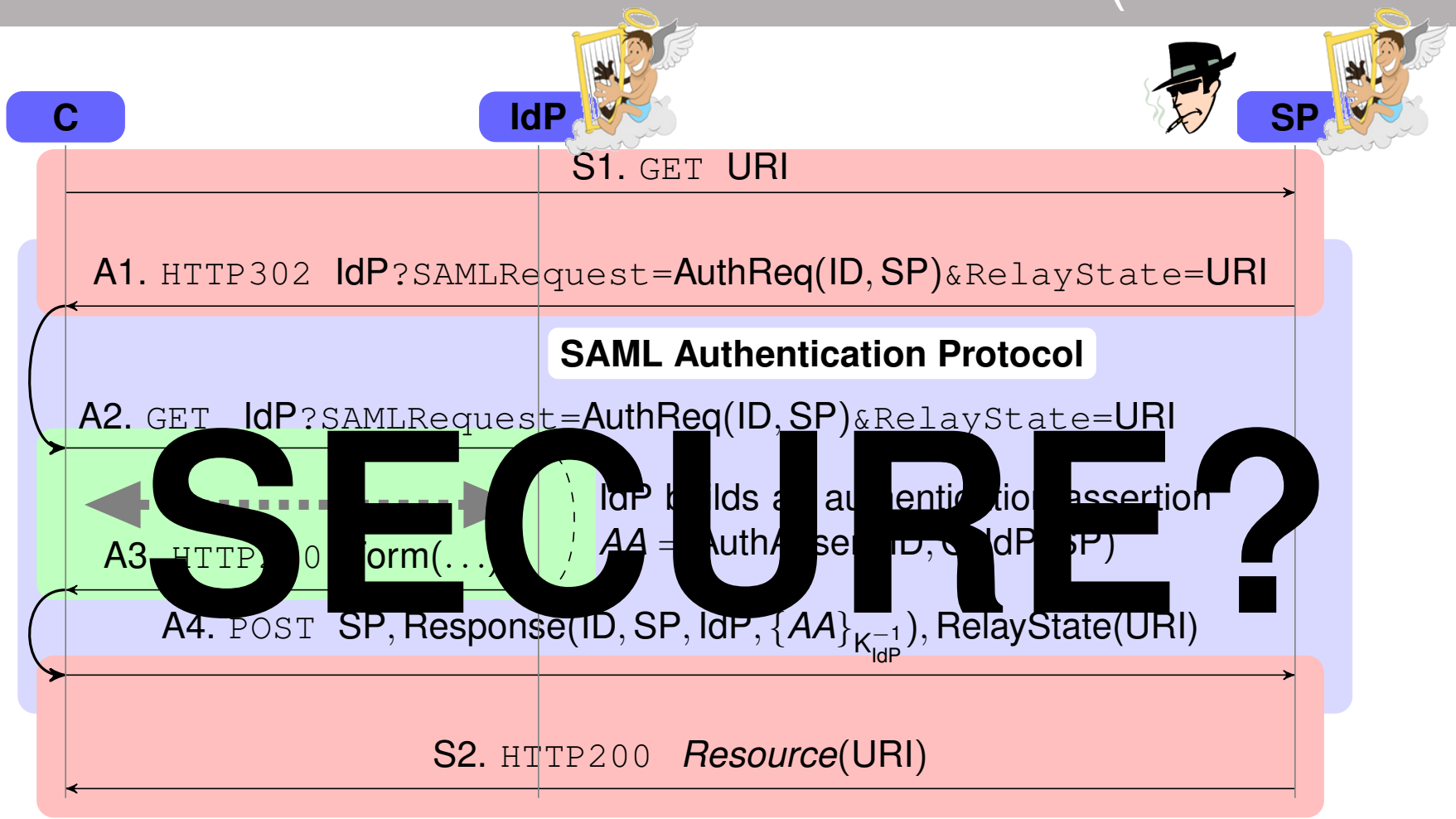
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



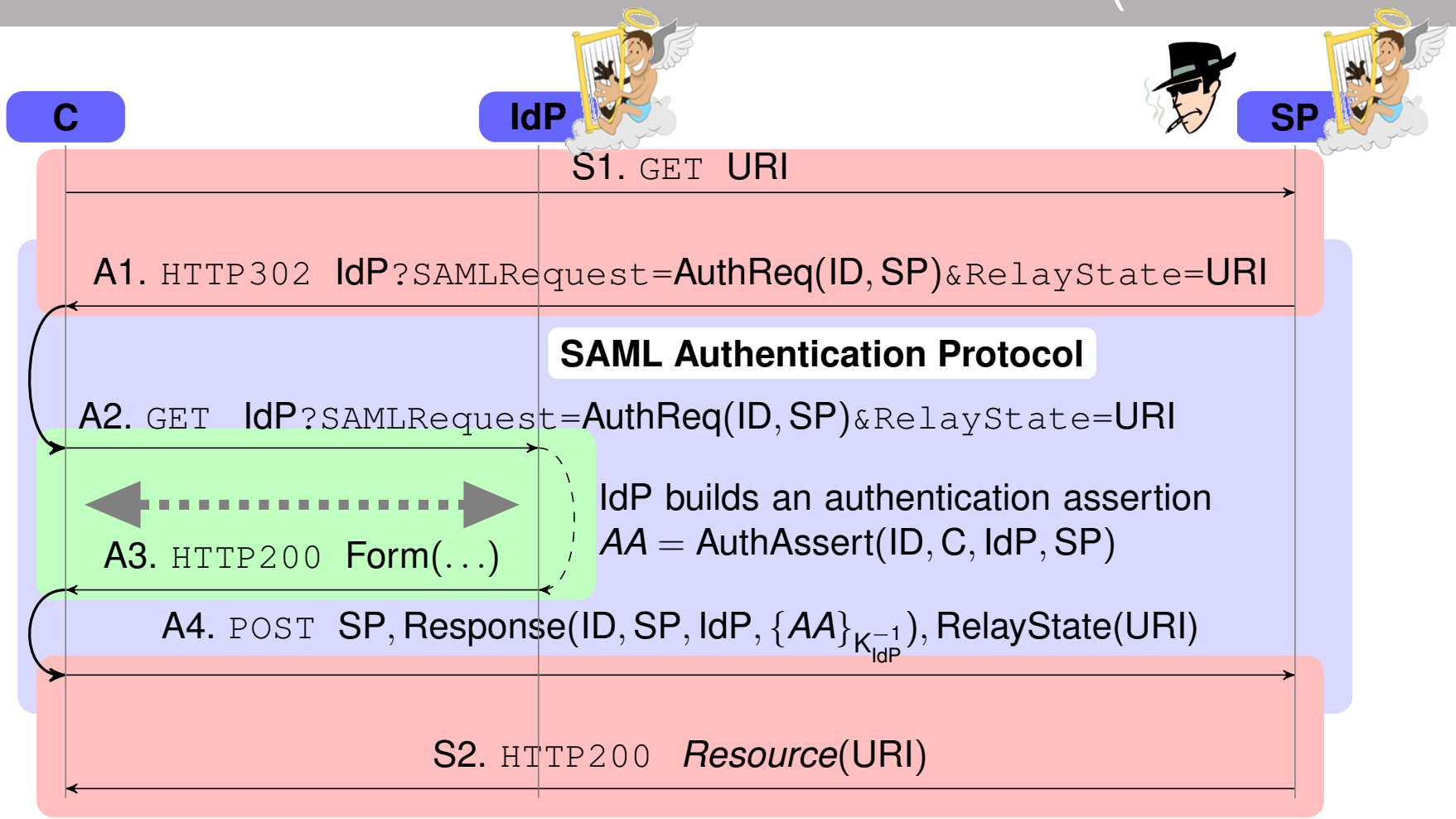
Important

We do not assume that all SPs whom C may play the protocol with are uncompromised.

The SAML 2.0 Web Browser SSO Profile (SAML SSO)



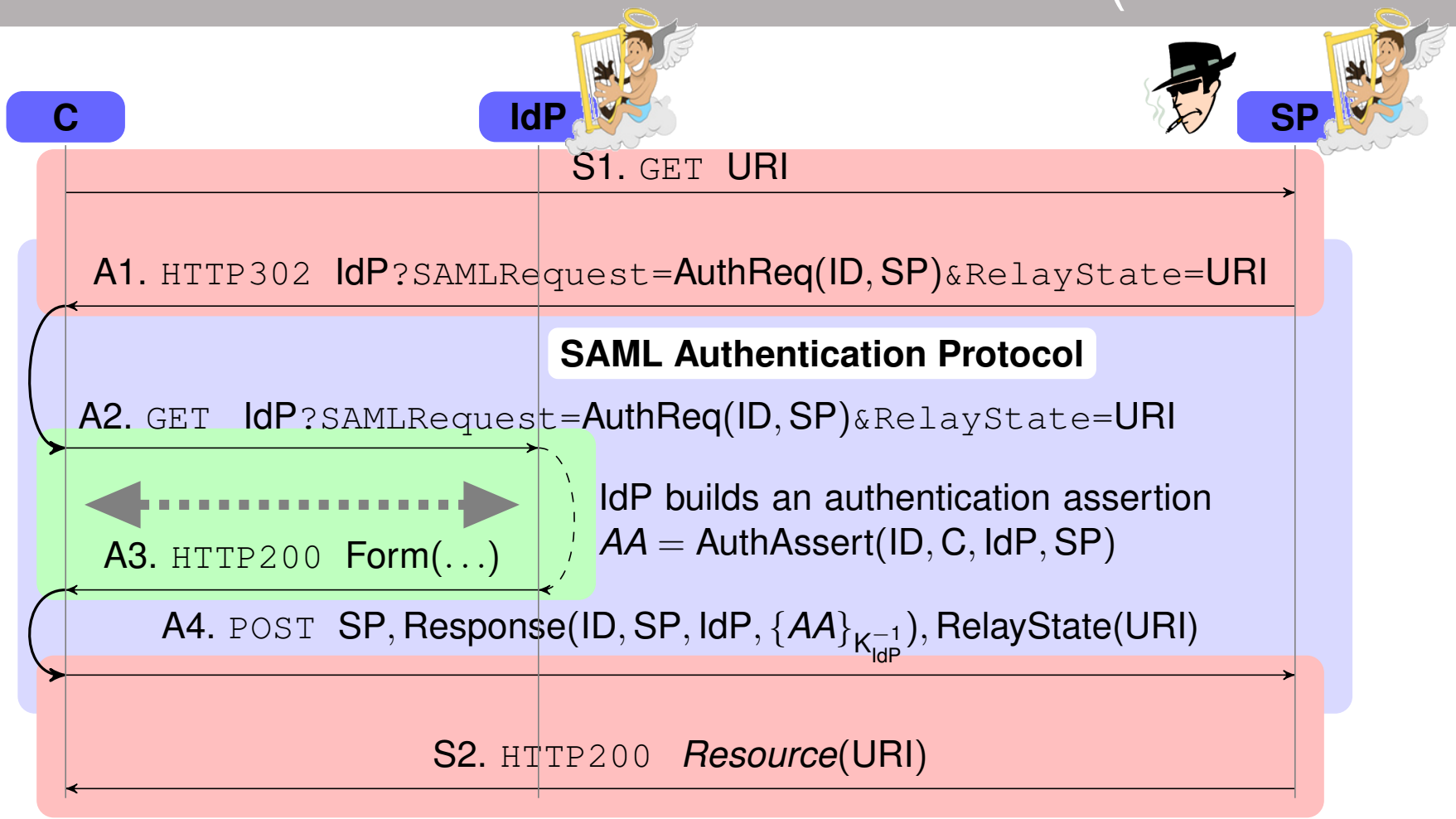
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Security Goal (SAML SSO)

SP and C mutually authenticate and agree on URI

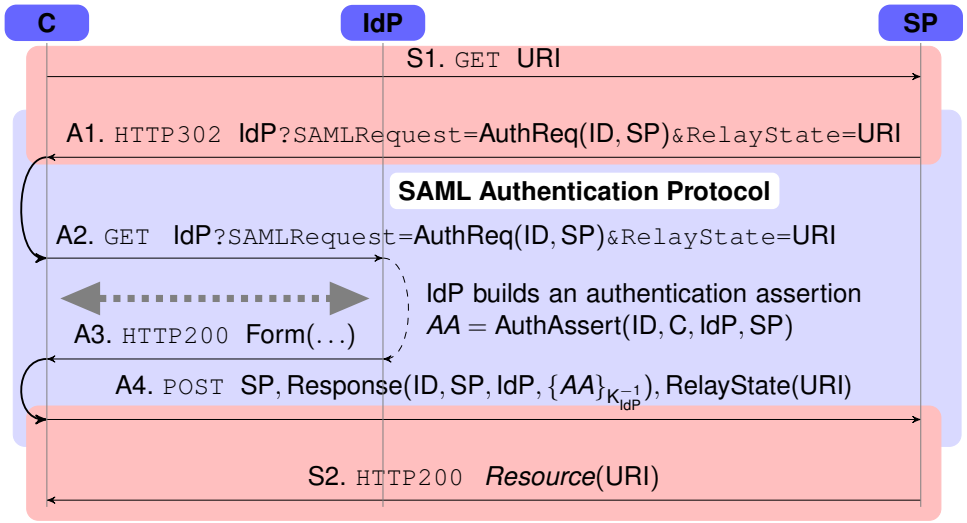
The SAML 2.0 Web Browser SSO Profile (SAML SSO)



Security Goal (SAML Authentication Prococol)

SP authenticates C

An Authentication Flaw in SAML SSO



The standard does not specify whether the messages at steps S1 and A4 must be transported over the same SSL/TLS channel.

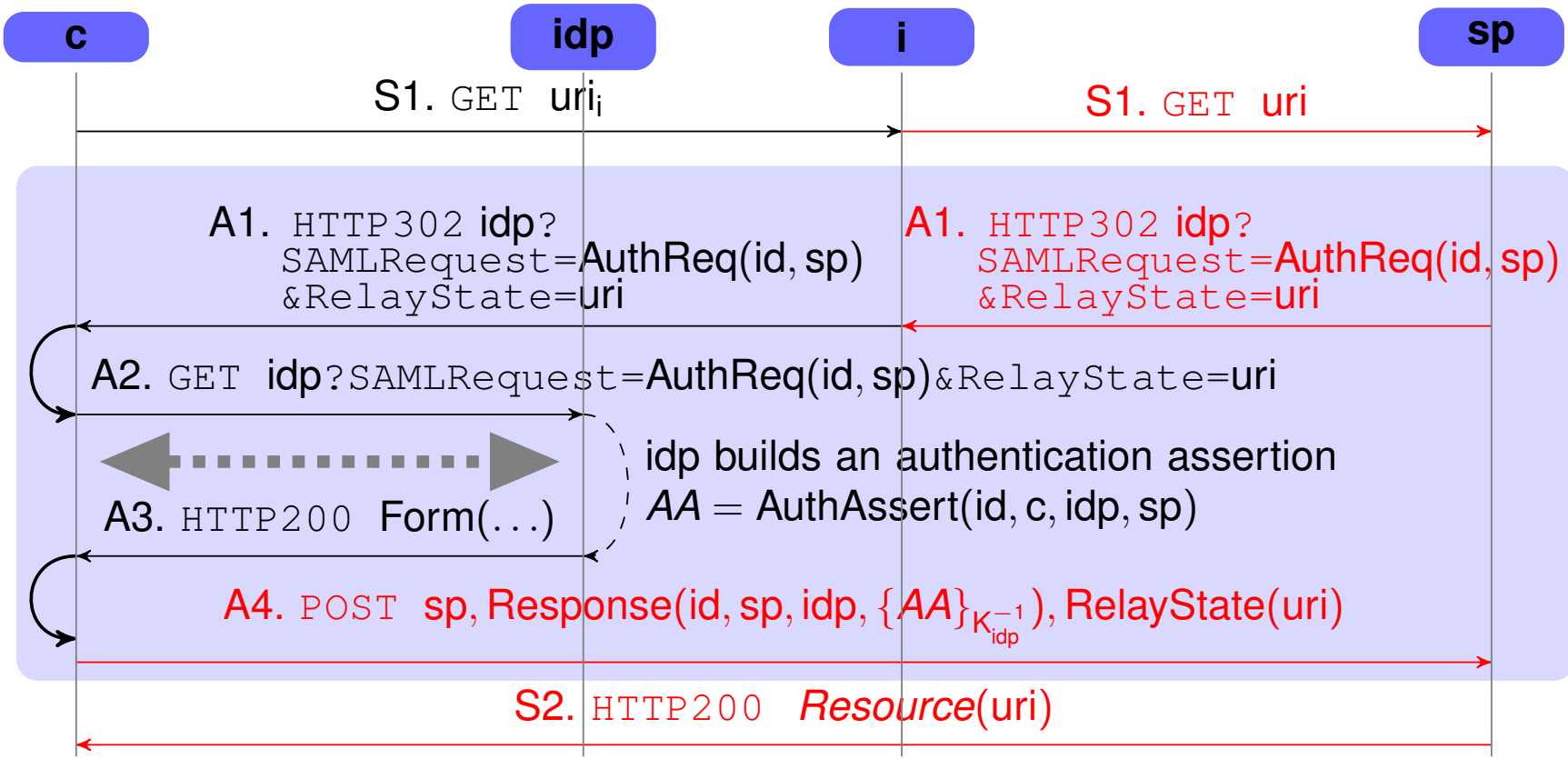
Reuse of the SSL/TLS channel apparently the most natural option, but difficult to achieve:

- **Resuming SSL/TLS sessions.**
  - the underlying TCP connection might be terminated,
  - an SSL server could not resume a previous session, or
  - the browser may very renegotiates the SSL session.
- **Software modularity.** The SW module that handles SAML messages may not have access to info of SSL/TLS.
- **Distributed SPs.** The SAML SP may be distributed over multiple machines, e.g., for work-balancing reasons.

An Attack on SAML SSO

We have built a formal model of the protocol and fed it to SATMC (`www.ai-lab.it/satmc`), a model-checker for security protocols.

SATMC found the following attack:



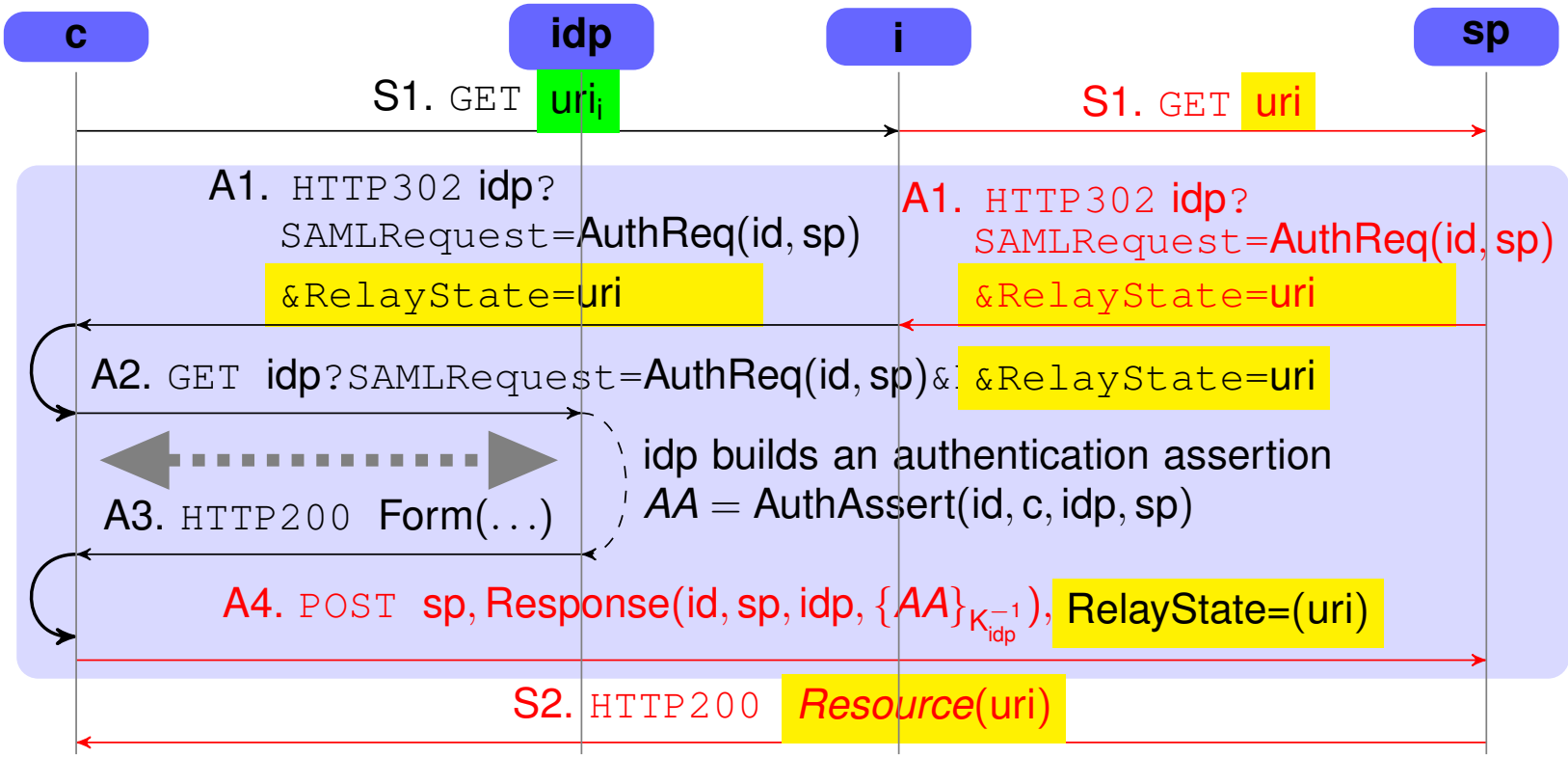
An Attack on SAML SSO: Considerations

- The attack would be prevented if SP could enforce that the S1 and A4 are carried over the same secure channel (but we have seen that this is very difficult to achieve in practice).
- Requiring digitally signed AuthReq would not fix the vulnerability.
- The same holds if the value in the `RelayState` is integrity protected as recommended by the standard.
- The attack does not require a malicious SP to be mounted. Any malicious web server *I* would be able, upon a request from *C*, to mount the attack provided that
  - C is a client of SP and
  - C has an active authentication context with IdP.

From Multiple Credentials to Browser-based SSO: Are We More Secure?

- The attack is possible because the client (a browser) does not verify whether the AuthReq and AuthResp are related to the initial request.
- Standard username/password authentication does not suffer from this (provided that C's credentials are different for each SP).
- The advantage of domain-specific credentials is that the user knows for which SP the credentials are intended.

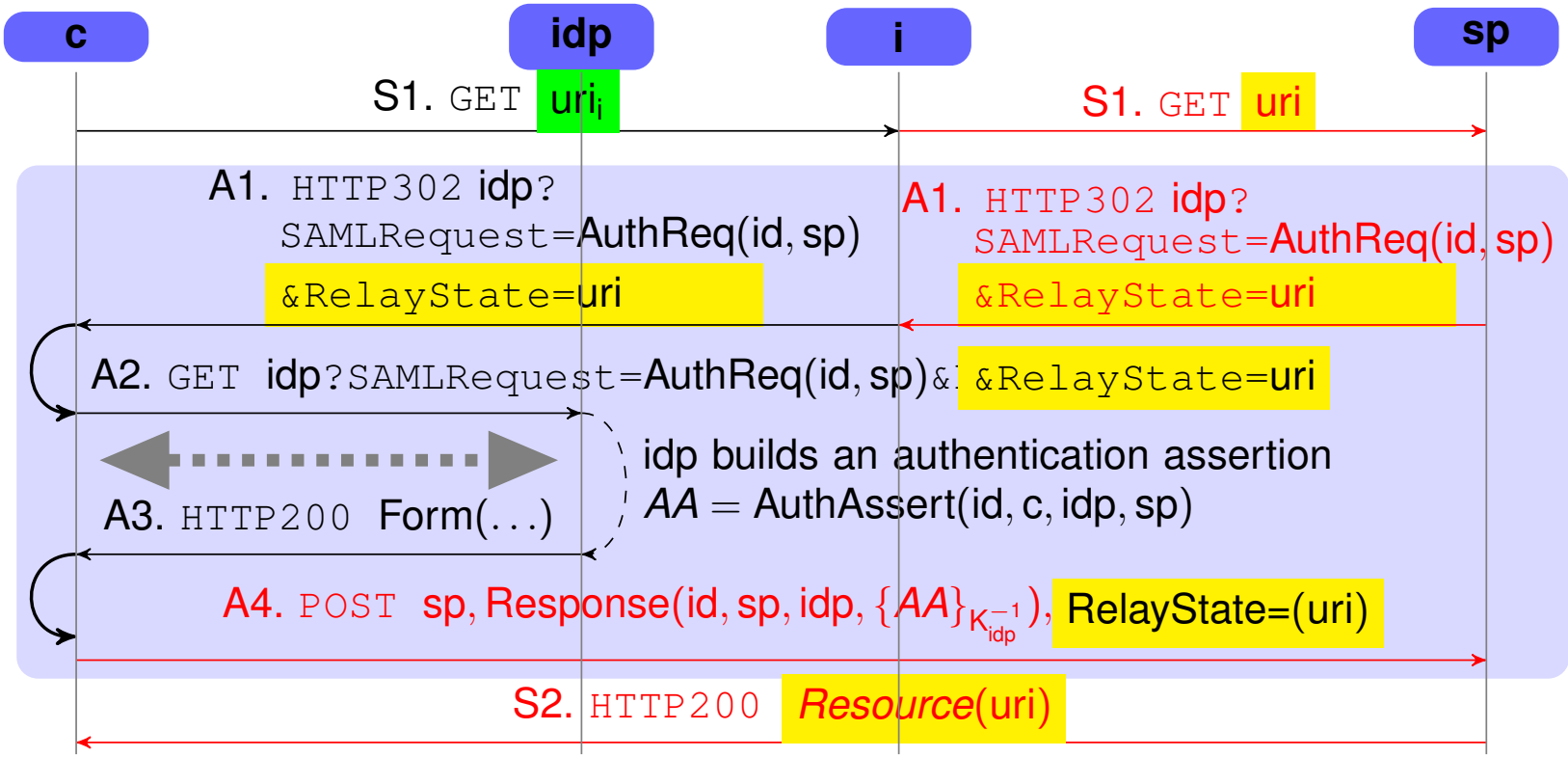
Exploiting the Vulnerability



Delivery of unrequested resource

Force C to receive a different resource from that initially requested.

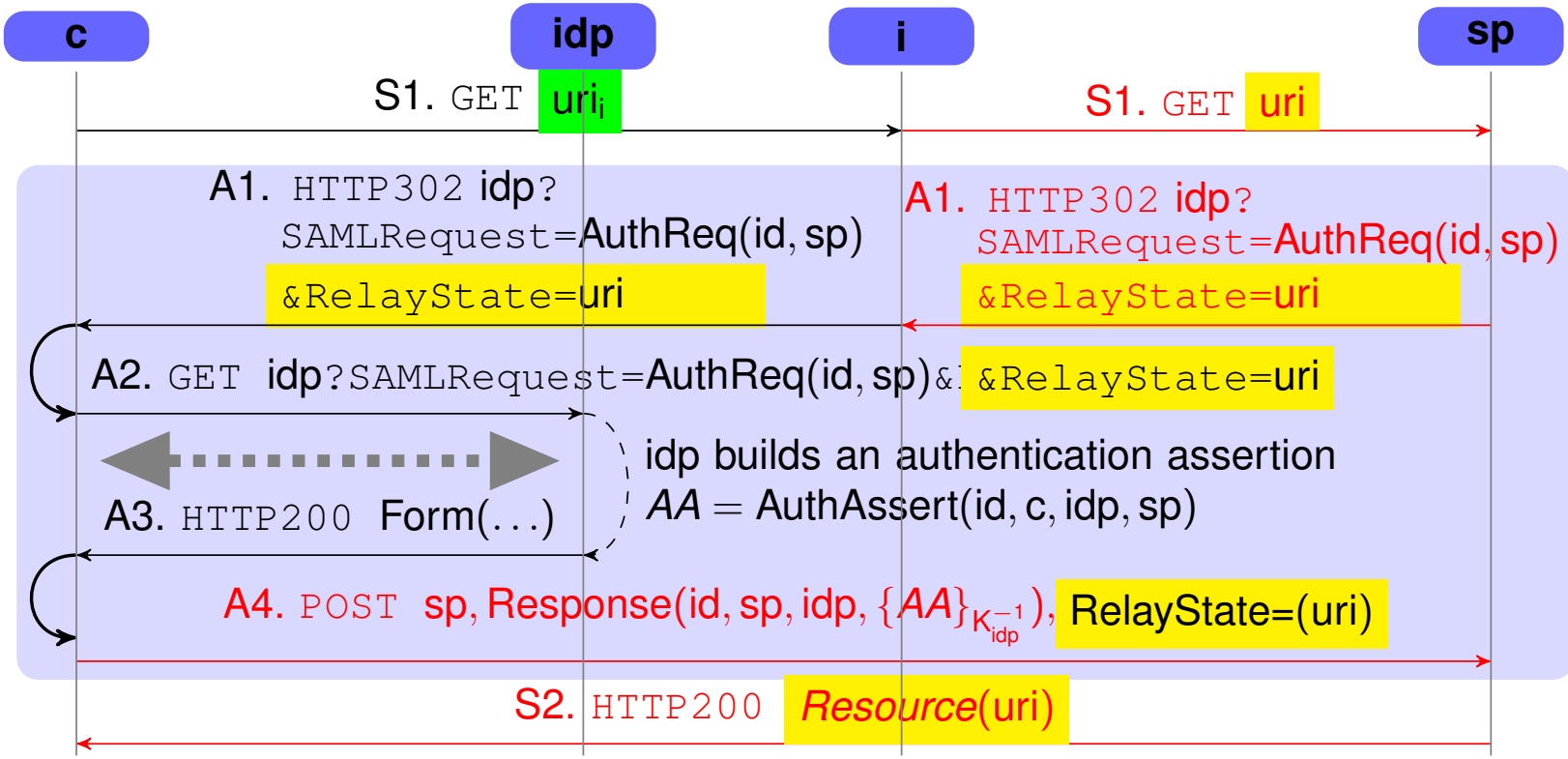
Exploiting the Vulnerability



Launching pad for XSRF

URI contains a URL-encoded command (e.g. a request to change of some settings). Even more pernicious than classic XSRF, because XSRF requires C to have an active session with SP, which is not the case here.

Exploiting the Vulnerability



Launching pad for XSS

`RelayState` exposed to injection of malicious code. Although the standard recommends to protect the integrity of this field, this often is not the case.

Experimenting with Deployed SSO Solutions

We have analyzed various SAML-based SSO solutions

- the SAML-based SSO for Google Apps,
- SimpleSAMLphp,
- Novell Access Manager 3.1.

All SPs considered accept SAML responses carried over different SSL/TLS channels.

Impact of the vulnerability on the Google Apps

Our analysis of the SAML-based SSO for Google Apps showed that:

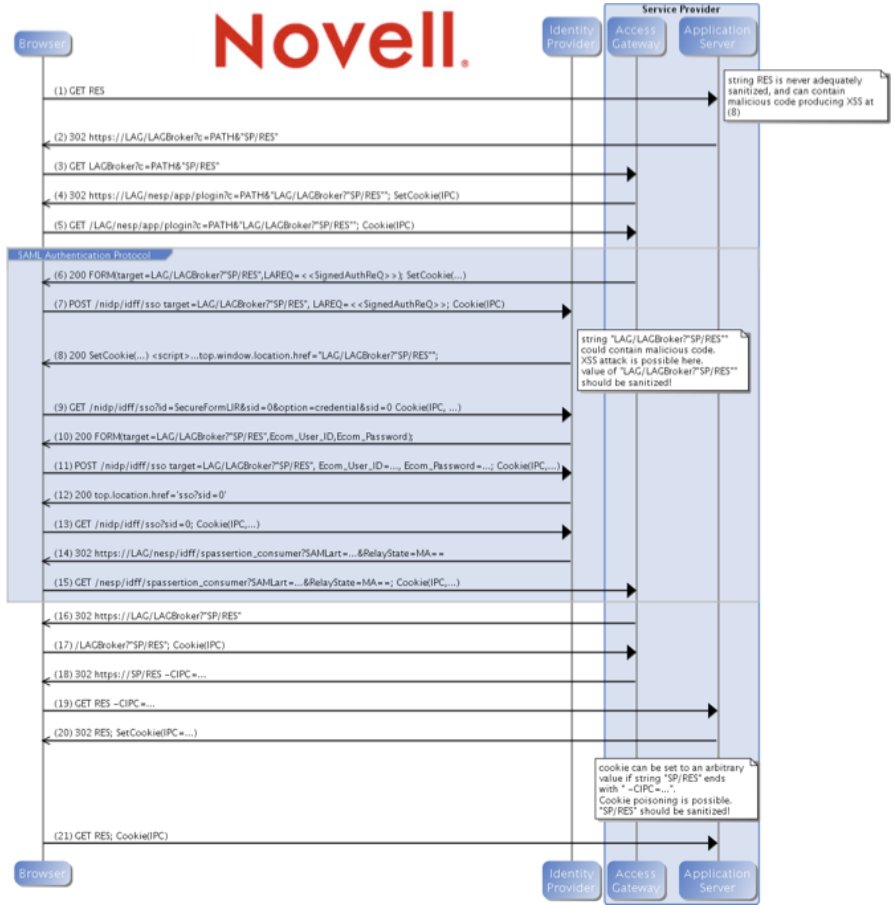
- `RelayState` was not sanitized and SAML SSO served as a launching pad for XSS.
- A malicious SP could force C to consume a resource from Google, for instance, visiting any page of the gmail service.
- A malicious SP could steal the cookies for the Google domain through XSS and could impersonate C on any Google application.

Impact of the vulnerability on simpleSAMLphp

- The SimpleSAMLphp stores the initially requested URI into the URL parameter `ReturnTo`.
- Although this field is not sanitized, no XSS could be mounted.
- The SP running SimpleSAMLphp use cookies that block the exploitation.

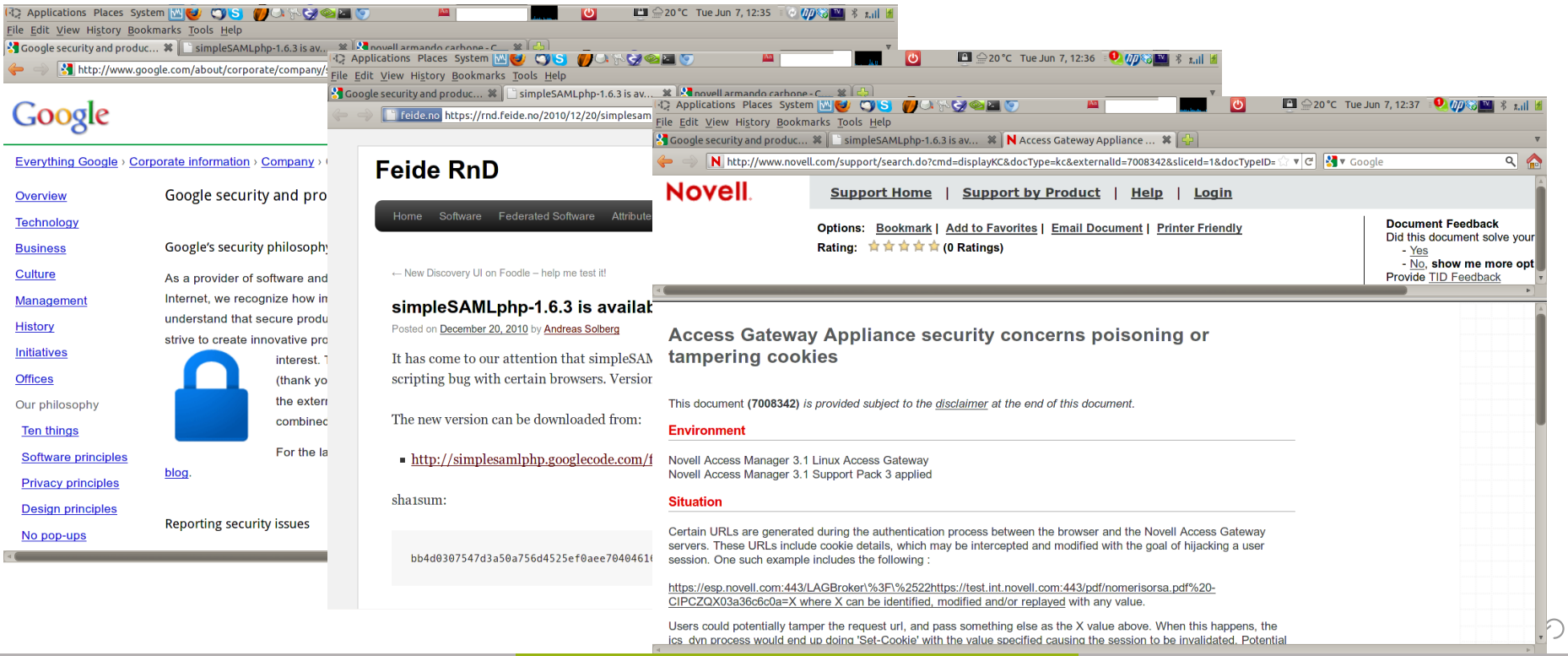
Impact of the vulnerability on Novell Access Manager

- URI not associated with the `RelayState` field as mandated by the standard, but passed as URL-encoded parameter which was not sanitized by the SP.
- XSS attack was possible.



Impact of the Vulnerability

- We proptly informed Google, Feide, and Novell
- who have since fixed the problems and
- credited us for finding and responsibly disclosing the vulnerabilities.



Fixing the Vulnerability

- The root of the problem of the authentication flaw lies in the following two factors:
  1. Clients are not able to link the AuthReq received in step A1 with their initial requests issued in step S1;
  2. SP is not able to enforce that the messages exchanged with C are carried over the same channel.
- If one of the two causes is removed the vulnerability is no longer exploitable.
- **Goal:** fix the problem so that existing solutions can be secured without radical modifications to the software components or to the standard.

Fixing the Vulnerability: Cookies

- By setting a session cookie in step A1 and expecting to receive it back on message A4, SP could check that the communication has occurred with the same client.
- Albeit sufficient in many scenarios, cookies only provide means to mitigate the problem and do not represent a complete countermeasure.
- Remind that cookies are difficult to steal but it is not as hard to set them.

Fixing the Vulnerability: Feedback from User

- IdP informs the user about the attempt to access URI on SP and asks for an explicit consent before issuing the authentication assertion to SP.
- In this way, the user may realise that the authentication is going to be sent to a different SP than expected and may be given the possibility to stop the protocol.
- Drawbacks:
  - it relies on a security decision taken by the user who may not have the expertise to tell apart legitimate redirections from malicious ones.
  - it breaks the seamlessness of SSO.

Self-signed Client Certificates

1. During the first SSL session C is asked to present the certificate.
2. SP generates an AuthReq with ID set to $n \,||\, \texttt{HMAC}_{K||n}(\texttt{RSA modulus})$, where n is a nonce, K is a secret known only to SP, and `RSA modulus` is the RSA modulus of the public key contained in the client's certificate.
3. SP deletes all state information and sends the AuthReq to C.
4. During second SSL session, C delivers same certificate to SP.
5. SP decomposes the ID field in the AuthResp as n' and H' and then check whether $H' = \texttt{HMAC}_{K||n'}(\texttt{RSA modulus})$.

Note that

- the SP can offer the client to generate one on his behalf;
- the certificate is *not* expected to carry information about the identity of the user.

Conclusions

- Authentication protocols are notoriously difficult to get right.
- Browser-based authentication protocols are no exception.
- In our paper we have
  - presented an authentication flaw in the SAML SSO
  - shown how this flaw can be generally exploited
  - reported related security issues that we have detected in actual SAML-based SSO solutions
  - presented a number of possible solutions
- Future work: do other browser-based SSO protocols suffer from the same problem?