C Algorithms

1 Algorithm idp

Input: $\langle a, b, m\rangle, s$

1: **let** $s' := s$
2: **let** $n, method, path, parameters, headers, body$ **such that** $\langle \texttt{HTTPReq}, n, method, path, parameters, headers, body\rangle \equiv m$ **if** *possible*; **otherwise** stop $\langle\rangle, s'$
3: **if** $path \equiv /script$ **then**
4: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{IdPScript}\rangle$
5: **stop** $\langle b, a, m'\rangle, s'$
6: **else if** $path \equiv /authentication$ **then**
7: **let** $cookie := headers[Cookie]$
8: **let** $session := s'.SessionList[cookie]$
9: **let** $uname := body[uname]$
10: **let** $passwd := body[passwd]$
11: **if** $passwd \not\equiv \texttt{PasswdOfUser}(\texttt{uname})$ **then**
12: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{LoginFailure}\rangle$
13: **stop** $\langle b, a, m'\rangle, s'$
14: **end if**
15: **let** $session[uid] := \texttt{UIDOfUser}(uname)$
16: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{LoginSucess}\rangle$
17: **stop** $\langle b, a, m'\rangle, s'$
18: **else if** $path \equiv /reqToken$ **then**
19: **let** $cookie := headers[Cookie]$
20: **let** $session := s'.SessionList[cookie]$
21: **let** $IDTokens := session[IDTokens]$
22: **if** $IDTokens[body[PID_{RP}]] \not\equiv \texttt{null}$ **then**
23: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, IDTokens[body[PID_{RP}]]\rangle$
24: **stop** $\langle b, a, m'\rangle, s'$
25: **end if**
26: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Unauthenticated}\rangle$
27: **stop** $\langle b, a, m'\rangle, s'$
28: **else if** $path \equiv /authorize$ **then**
29: **let** $cookie := headers[Cookie]$
30: **let** $session := s'.SessionList[cookie]$
31: **let** $uid := session[uid]$
32: **if** $uid \equiv \texttt{null}$ **then**
33: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
34: **stop** $\langle b, a, m'\rangle, s'$
35: **end if**
36: **let** $PID_{RP} := parameters[PID_{RP}]$
37: **if** $\texttt{IsValid}(PID_{RP}) \equiv \texttt{FALSE}$ **then**
38: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail}\rangle$
39: **stop** $\langle b, a, m'\rangle, s'$

40: **end if**
41: **if** $\texttt{IsInScope}(uid, body[Attr]) \equiv \texttt{FALSE}$ **then**
42: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail} \rangle$
43: **stop** $\langle b, a, m' \rangle, s'$
44: **end if**
45: **let** $ID_U := session[uid]$
46: **let** $PID_U := \texttt{Multiply}(PID_{RP}, ID_U)$
47: **let** $Validity := \texttt{CurrentTime}() + s'.Validity$
48: **let** $Content := \langle PID_{RP}, PID_U, s'.Issuer, Validity \rangle$
49: **let** $Sig := \texttt{SigSign}(Content, s'.SK)$
50: **let** $IDToken := \langle Content, Sig \rangle$
51: **let** $session[IDTokens] := session[IDTokens] +^{\langle\rangle} \langle PID_{RP}, IDToken \rangle$
52: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, Token \rangle$
53: **stop** $\langle b, a, m' \rangle, s'$
54: **end if**
55: **stop** $\langle\rangle, s'$

---

2 Algorithm rp

Input: $\langle a, b, m \rangle, s$
1: **let** $s' := s$
2: **let** $n, method, path, parameters, headers, body$ **such that**
$\langle \texttt{HTTPReq}, n, method, path, parameters, headers, body \rangle \equiv m$
if *possible*; **otherwise** stop $\langle\rangle, s'$
3: **if** $path \equiv /script$ **then**
4: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{RPScript} \rangle$
5: **stop** $\langle b, a, m' \rangle, s'$
6: **else if** $path \equiv /loginSSO$ **then**
7: **let** $m' := \langle \texttt{HTTPResp}, n, 302, \langle \langle \texttt{Location}, s'.IdP.ScriptUrl \rangle \rangle, \langle\rangle \rangle$
8: **stop** $\langle b, a, m' \rangle, s'$
9: **else if** $path \equiv /startNegotiation$ **then**
10: **let** $cookie := headers[Cookie]$
11: **let** $session := s'.SessionList[cookie]$
12: **let** $t := body[t]$
13: **let** $t^{-1} := \texttt{Inverse}(t)$
14: **let** $session[t^{-1}] := t^{-1}$
15: **let** $session[state] := expectToken$
16: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \langle \texttt{Cert}_{\texttt{RP}}, s'.Cert_{RP} \rangle \rangle$
17: **stop** $\langle b, a, m' \rangle, s'$
18: **else if** $path \equiv /uploadToken$ **then**
19: **let** $cookie := headers[Cookie]$
20: **let** $session := s'.SessionList[cookie]$
21: **if** $session[state] \not\equiv expectToken$ **then**
22: **let** $m' := \langle \texttt{HTTPResp}, n, 200, \langle\rangle, \texttt{Fail} \rangle$
23: **stop** $\langle b, a, m' \rangle, s'$
24: **end if**

```
25:    let IDToken := body[IDToken]
26:    if checksig(IDToken.Content, IDToken.Sig, s'.IdP.PK) ≡ FALSE then
27:       let m' := ⟨HTTPResp, n, 200, ⟨⟩, Fail⟩
28:       stop ⟨b, a, m'⟩, s'
29:    end if
30:    let Time := CurrentTime()
31:    let Content := Token.Content
32:    if Time > Content.Validity then
33:       let m' := ⟨HTTPResp, n, 200, ⟨⟩, Fail⟩
34:       stop ⟨b, a, m'⟩, s'
35:    end if
36:    let PID_U := Content.PID_U
37:    let t^-1 := session[t^-1]
38:    let Acct := Multiply(PID_U, t^-1)
39:    if Acct ∉ ListOfUser() then
40:       let AddUser(Acct)
41:    end if
42:    let session[user] := Acct
43:    let s'.serviceTokens := s'.serviceTokens +^⟨⟩ ⟨IDToken, Acct⟩
44:    let m' := ⟨HTTPResp, n, 200, ⟨⟩, LoginSuccess⟩
45:    stop ⟨b, a, m'⟩, s'
46: end if
47: stop ⟨⟩, s'
```

3 Algorithm script_idp

Input: $\langle tree, docID, scriptstate, scriptinputs, cookies, ids, secret\rangle$

```
 1: let s' := scriptstate
 2: let command := ⟨⟩
 3: let target := PARENTWINDOW(tree, docID)
 4: let IdPDomain := s'.IdPDomain
 5: switch s'.phsae do
 6:    case start:
 7:       let t := Random()
 8:       let command := ⟨POSTMESSAGE, target, ⟨t, t⟩, null⟩
 9:       let s'.Parameters[t] := t
10:       let s'.phase := expectCert
11:    case expectCert:
12:       let pattern := ⟨POSTMESSAGE, target, *, ⟨Cert_RP, *⟩⟩
13:       let input := CHOOSEINPUT(scriptinputs, pattern)
14:       if input ≠ null then
15:          let Cert_RP := π2(π4(input))
16:          if checksig(Cert.Content, Cert.Sig, s'.PubKey) ≡ null then
17:             let stop ⟨⟩
18:          end if
```

```
        let s'.Parameters[Cert] := Cert_RP
        let t := s'.Parameters[t]
        let PID_RP := Multiply(Cert_RP.ID_RP, t)
        let s'.Parameters[PID_RP] := PID_RP
        let Url := ⟨URL, S, IdPDomain, /reqToken, ⟨⟨PID_RP, PID_RP⟩⟩⟩
        let command := ⟨XMLHTTPREQUEST, Url, GET, ⟨⟩, s'.refXHR⟩
        let s'.phase := expectLoginState
      end if
    case expectReqToken:
      let pattern := ⟨XMLHTTPREQUEST, Body, s'.refXHR⟩
      let input := CHOOSEINPUT(scriptinputs, pattern)
      if input ≠ null then
        if π2(input) ≡ Unanthenticated then
          let user ∈ ids
          let Url := ⟨URL, S, IdPDomain, /authentication, ⟨⟩⟩
          let command := ⟨XMLHTTPREQUEST, Url, POST, ⟨⟨uname, uname⟩, ⟨passwd, passwd⟩⟩, s'.refXHR⟩
          let s'.phase := expectLoginResult
        end if
        let IDToken := π2(input)[IDToken]
        let RPOringin := ⟨s'.Parameters[Cert].Enpt, S⟩
        let command := ⟨POSTMESSAGE, target, ⟨IDToken, IDToken⟩, RPOrigin⟩
        let s.phase := stop
      end if
    case expectLoginResult:
      let pattern := ⟨XMLHTTPREQUEST, Body, s'.refXHR⟩
      let input := CHOOSEINPUT(scriptinputs, pattern)
      if input ≠ null then
        if π2(input) ≠ LoginSuccess then
          let stop ⟨⟩
        end if
        let PID_RP := s'.Parameters[PID_RP]
        let Url := ⟨URL, S, IdPDomain, /authorize, ⟨⟨PID_RP, PID_RP⟩, ⟨Attr, Attr⟩⟩⟩
        let command := ⟨XMLHTTPREQUEST, Url, GET, ⟨⟩, s'.refXHR⟩
        let s'.phase := expectToken
      end if
    case expectToken:
      let pattern := ⟨XMLHTTPREQUEST, Body, s'.refXHR⟩
      let input := CHOOSEINPUT(scriptinputs, pattern)
      if input ≠ null then
        let IDToken := π2(input)[IDToken]
        let RPOringin := ⟨s'.Parameters[Cert].Enpt, S⟩
        let command := ⟨POSTMESSAGE, target, ⟨IDToken, IDToken⟩, RPOrigin⟩
        let s.phase := stop
      end if
  end switch
  let stop ⟨s', cookies, localStorage, sessionStorage, command⟩
```

4 Algorithm script_rp

Input: $\langle tree, docID, scriptstate, scriptinputs, cookies, ids, secret\rangle$

1: **let** $s' := scriptstate$
2: **let** $command := \langle\rangle$
3: **let** $IdPWindow :=$ SUBWINDOW$(tree, docnonce).winID$
4: **let** $RPDomain := s'.RPDomain$
5: **let** $IdPOringin := \langle s'.IdPDomian, \mathtt{S}\rangle$
6: **switch** $s'.phase$ **do**
7: **case** *start*:
8: **let** $Url := \langle \mathtt{URL}, \mathtt{S}, RPDomain, /loginSSO, \langle\rangle\rangle$
9: **let** $command := \langle \mathtt{IFRAME}, Url, \_SELF\rangle$
10: **let** $s'.phase := expectt$
11: **case** *expectt*:
12: **let** $pattern := \langle \mathtt{POSTMESSAGE}, target, *, \langle \mathtt{t}, *\rangle\rangle$
13: **let** $input :=$ CHOOSEINPUT$(scriptinputs, pattern)$
14: **if** $input \not\equiv$ `null` **then**
15: **let** $t := \pi_2(\pi_4(input))[t]$
16: **let** $Url := \langle \mathtt{URL}, \mathtt{S}, RPDomain, /startNegotiation, \langle\rangle\rangle$
17: **let** $command := \langle \mathtt{XMLHTTPREQUEST}, Url, \mathtt{POST}, \langle\langle t, t\rangle\rangle, s'.refXHR\rangle$
18: **let** $s'.phase := expectCert$
19: **end if**
20: **case** *expectCert*:
21: **let** $pattern := \langle \mathtt{XMLHTTPREQUEST}, Body, s'.refXHR\rangle$
22: **let** $input :=$ CHOOSEINPUT$(scriptinputs, pattern)$
23: **if** $input \not\equiv$ `null` **then**
24: **let** $Cert_{RP} := \pi_2(input)[Cert_{RP}]$
25: **let** $command := \langle \mathtt{POSTMESSAGE}, IdPWindow, \langle\langle \mathtt{Cert}, Cert\rangle\rangle, IdPOringin\rangle$
26: **let** $s'.phase := expectToken$
27: **end if**
28: **case** *expectToken*:
29: **let** $pattern := \langle \mathtt{POSTMESSAGE}, target, *, \langle \mathtt{IDToken}, *\rangle\rangle$
30: **let** $input :=$ CHOOSEINPUT$(scriptinputs, pattern)$
31: **if** $input \not\equiv$ `null` **then**
32: **let** $IDToken := \pi_2(input)[IDToken]$
33: **let** $Url := \langle \mathtt{URL}, \mathtt{S}, RPDomain, /uploadToken, \langle\rangle\rangle$
34: **let** $command := \langle \mathtt{XMLHTTPREQUEST}, Url, \mathtt{POST}, \langle\langle \mathtt{IDToken}, IDToken\rangle\rangle, s'.refXHR\rangle$
35: **let** $s'.phase := expectLoginResult$
36: **end if**
37: **case** *expectLoginResult*:
38: **let** $pattern := \langle \mathtt{XMLHTTPREQUEST}, Body, s'.refXHR\rangle$
39: **let** $input :=$ CHOOSEINPUT$(scriptinputs, pattern)$
40: **if** $input \not\equiv$ `null` **then**
41: **if** $\pi_2(input) \equiv$ `LoginSuccess` **then**
42: **let** Load Homepage

43: **end if**
44: **end if**
45: **end switch**
46: **let stop** $\langle s', cookies, localStorage, sessionStorage, command \rangle$