MODULE *STPWebModel*

EXTENDS *Integers*, *TLC*, *Sequences*

CONSTANT *HonestKey*, *AtkKey*, *HonestSessionId*

```
--algorithm STPWebModel
variables
    MessageLog = {} ;
    HonestBrowser = [source ↦ "honest", Cookies ↦ {}, Headers ↦ {}] ;
    HonestServerState = [SessionIds ↦ {}, Tokens ↦ {}] ;
    AttackerServerState = [SessionIds ↦ {}, Tokens ↦ {}] ;
    HonestSecretKey = [secret ↦ {HonestKey}] ;
    AttackerSecretKey = [secret ↦ {AtkKey}] ;
    AttackerState = [SessionIds ↦ {}] ;
    AttackerBrowser = [source ↦ "attacker", Cookies ↦ {}, Headers ↦ {}]

define

    HonestAccepted ≜
        ∀ msg ∈ MessageLog :
            msg.request.source = "honest" ⇒ msg.response.status = "success"
    MaliciousRejected ≜
        ∀ msg ∈ MessageLog :
            msg.request.source = "attacker" ⇒ msg.response.status = "error"


    HasValidSessionId(req, srv) ≜ ∃ SessionId ∈ srv.SessionIds : ∃ cookie ∈ req.cookies : SessionId = cookie
    HasValidToken(req, srv) ≜ ∃ Token ∈ srv.Tokens: ∃ Header ∈ req.headers: Token = Header
    HasValidToken(req, srv) ≜ (srv.Tokens = {HonestKey}) = (req.cookies = {HonestSessionId})
    ServerRequest(src) ≜ [source ↦ src.source, cookies ↦ src.Cookies, headers ↦ src.Headers]
    Response(req, srv) ≜ IF (HasValidToken(req, srv)) = (HasValidSessionId(req, srv)) THEN [destination ↦

end define ;


procedure LogMessagePair(req, resp)begin
    Log:
    MessageLog := MessageLog ∪ {[request ↦ req, response ↦ resp]} ;
    print ⟨MessageLog⟩ ;
    return ;
end procedure ;

process SiteRequest = 1
begin
    SSR:
    print ⟨"Same Site Req"⟩ ;

    Session:
```

```
    HonestBrowser.Cookies := {HonestSessionId} ;
    HonestServerState.SessionIds := HonestBrowser.Cookies ;

    Tokens:
    HonestServerState.Tokens := HonestSecretKey.secret ;
    HonestBrowser.Headers := HonestServerState.Tokens ;
    call LogMessagePair(ServerRequest(HonestBrowser), Response(ServerRequest(HonestBrowser), HonestSe

end process ;

process CrossSiteRequest = 2
begin
    CSR:
    print ⟨ "Cross Site Req" ⟩ ;

    HonestToAttacker:
    HonestBrowser.Cookies := {HonestSessionId} ;
    AttackerState.SessionIds := HonestBrowser.Cookies ;
    AttackerBrowser.Cookies := AttackerState.SessionIds ;

    Session:
    AttackerServerState.SessionIds := AttackerBrowser.Cookies ;

    Tokens:
    AttackerServerState.Tokens := AttackerSecretKey.secret ;
    AttackerBrowser.Headers := AttackerServerState.Tokens ;
    call LogMessagePair(ServerRequest(AttackerBrowser), Response(ServerRequest(AttackerBrowser), Hones

end process ;

end algorithm ;
```

```
\*
BEGIN TRANSLATION (chksum(pcal) = "1544d951" ∧ chksum(tla) = "fc531894")
Label Session of process SiteRequest at line 50 col 5 changed to Session_
Label Tokens of process SiteRequest at line 54 col 5 changed to Tokens_
CONSTANT defaultInitValue
VARIABLES MessageLog, HonestBrowser, HonestServerState, AttackerServerState,
          HonestSecretKey, AttackerSecretKey, AttackerState, AttackerBrowser,
          pc, stack

define statement
HonestAccepted ≜
    ∀ msg ∈ MessageLog :
        msg.request.source = "honest" ⇒ msg.response.status = "success"
MaliciousRejected ≜
    ∀ msg ∈ MessageLog :
        msg.request.source = "attacker" ⇒ msg.response.status = "error"
```

$HasValidSessionId(req,\ srv) \triangleq \exists\, SessionId \in srv.SessionIds : \exists\, cookie \in req.cookies : SessionId = cookie$

$HasValidToken(req,\ srv) \triangleq (srv.Tokens = \{HonestKey\}) = (req.cookies = \{HonestSessionId\})$
$ServerRequest(src) \triangleq [source \mapsto src.source,\ cookies \mapsto src.Cookies,\ headers \mapsto src.Headers]$
$Response(req,\ srv) \triangleq \text{IF } (HasValidToken(req,\ srv)) = (HasValidSessionId(req,\ srv)) \text{ THEN } [destination \mapsto re$

VARIABLES $req,\ resp$

$$vars \triangleq \langle MessageLog,\ HonestBrowser,\ HonestServerState,\ AttackerServerState,\ HonestSecretKey,\ AttackerSecretKey,\ AttackerState,\ AttackerBrowser,\ pc,\ stack,\ req,\ resp\rangle$$

$$ProcSet \triangleq \{1\} \cup \{2\}$$

$$\begin{aligned}
Init \triangleq\ & \text{Global variables} \\
& \land MessageLog = \{\} \\
& \land HonestBrowser = [source \mapsto \text{“honest”},\ Cookies \mapsto \{\},\ Headers \mapsto \{\}] \\
& \land HonestServerState = [SessionIds \mapsto \{\},\ Tokens \mapsto \{\}] \\
& \land AttackerServerState = [SessionIds \mapsto \{\},\ Tokens \mapsto \{\}] \\
& \land HonestSecretKey = [secret \mapsto \{HonestKey\}] \\
& \land AttackerSecretKey = [secret \mapsto \{AtkKey\}] \\
& \land AttackerState = [SessionIds \mapsto \{\}] \\
& \land AttackerBrowser = [source \mapsto \text{“attacker”},\ Cookies \mapsto \{\},\ Headers \mapsto \{\}] \\
& \text{Procedure } LogMessagePair \\
& \land req = [self \in ProcSet \mapsto defaultInitValue] \\
& \land resp = [self \in ProcSet \mapsto defaultInitValue] \\
& \land stack = [self \in ProcSet \mapsto \langle\rangle] \\
& \land pc = [self \in ProcSet \mapsto \text{CASE } self = 1 \to \text{“SSR”} \\
& \qquad\qquad\qquad\qquad\quad \square\ \ self = 2 \to \text{“CSR”}]
\end{aligned}$$

$$\begin{aligned}
Log(self) \triangleq\ & \land pc[self] = \text{“Log”} \\
& \land MessageLog' = (MessageLog \cup \{[request \mapsto req[self],\ response \mapsto resp[self]]\}) \\
& \land PrintT(\langle MessageLog'\rangle) \\
& \land pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc] \\
& \land req' = [req \text{ EXCEPT } ![self] = Head(stack[self]).req] \\
& \land resp' = [resp \text{ EXCEPT } ![self] = Head(stack[self]).resp] \\
& \land stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])] \\
& \land \text{UNCHANGED } \langle HonestBrowser,\ HonestServerState, \\
& \qquad\qquad AttackerServerState,\ HonestSecretKey, \\
& \qquad\qquad AttackerSecretKey,\ AttackerState,\ AttackerBrowser\rangle
\end{aligned}$$

$$LogMessagePair(self) \triangleq Log(self)$$

$$\begin{aligned}
SSR \triangleq\ & \land pc[1] = \text{“SSR”} \\
& \land PrintT(\langle\text{“Same Site Req”}\rangle) \\
& \land pc' = [pc \text{ EXCEPT } ![1] = \text{“Session_”}] \\
& \land \text{UNCHANGED } \langle MessageLog,\ HonestBrowser,\ HonestServerState,
\end{aligned}$$

$AttackerServerState$, $HonestSecretKey$, $AttackerSecretKey$,
$AttackerState$, $AttackerBrowser$, $stack$, req, $resp\rangle$

$Session\_ \triangleq$ $\land\ pc[1] = \text{“Session\_”}$
$\land\ HonestBrowser' = [HonestBrowser \text{ EXCEPT } !.Cookies = \{HonestSessionId\}]$
$\land\ HonestServerState' = [HonestServerState \text{ EXCEPT } !.SessionIds = HonestBrowser'.Cookies]$
$\land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Tokens\_”}]$
$\land$ UNCHANGED $\langle MessageLog$, $AttackerServerState$, $HonestSecretKey$,
$AttackerSecretKey$, $AttackerState$, $AttackerBrowser$,
$stack$, req, $resp\rangle$

$Tokens\_ \triangleq$ $\land\ pc[1] = \text{“Tokens\_”}$
$\land\ HonestServerState' = [HonestServerState \text{ EXCEPT } !.Tokens = HonestSecretKey.secret]$
$\land\ HonestBrowser' = [HonestBrowser \text{ EXCEPT } !.Headers = HonestServerState'.Tokens]$
$\land\ \land\ req' = [req \text{ EXCEPT } ![1] = ServerRequest(HonestBrowser')]$
$\land\ resp' = [resp \text{ EXCEPT } ![1] = Response(ServerRequest(HonestBrowser'), HonestServerState')]$
$\land\ stack' = [stack \text{ EXCEPT } ![1] = \langle [procedure \mapsto \text{“LogMessagePair”},$
$pc \mapsto \text{“Done”},$
$req \mapsto req[1],$
$resp \mapsto resp[1]]\rangle$
$\circ\ stack[1]]$
$\land\ pc' = [pc \text{ EXCEPT } ![1] = \text{“Log”}]$
$\land$ UNCHANGED $\langle MessageLog$, $AttackerServerState$, $HonestSecretKey$,
$AttackerSecretKey$, $AttackerState$, $AttackerBrowser\rangle$

$SiteRequest \triangleq SSR \lor Session\_ \lor Tokens\_$

$CSR \triangleq$ $\land\ pc[2] = \text{“CSR”}$
$\land\ PrintT(\langle \text{“Cross Site Req”} \rangle)$
$\land\ pc' = [pc \text{ EXCEPT } ![2] = \text{“HonestToAttacker”}]$
$\land$ UNCHANGED $\langle MessageLog$, $HonestBrowser$, $HonestServerState$,
$AttackerServerState$, $HonestSecretKey$, $AttackerSecretKey$,
$AttackerState$, $AttackerBrowser$, $stack$, req, $resp\rangle$

$HonestToAttacker \triangleq$ $\land\ pc[2] = \text{“HonestToAttacker”}$
$\land\ HonestBrowser' = [HonestBrowser \text{ EXCEPT } !.Cookies = \{HonestSessionId\}]$
$\land\ AttackerState' = [AttackerState \text{ EXCEPT } !.SessionIds = HonestBrowser'.Cookies]$
$\land\ AttackerBrowser' = [AttackerBrowser \text{ EXCEPT } !.Cookies = AttackerState'.SessionIds]$
$\land\ pc' = [pc \text{ EXCEPT } ![2] = \text{“Session”}]$
$\land$ UNCHANGED $\langle MessageLog$, $HonestServerState$,
$AttackerServerState$, $HonestSecretKey$,
$AttackerSecretKey$, $stack$, req, $resp\rangle$

$Session \triangleq$ $\land\ pc[2] = \text{“Session”}$
$\land\ AttackerServerState' = [AttackerServerState \text{ EXCEPT } !.SessionIds = AttackerBrowser.Cookies]$
$\land\ pc' = [pc \text{ EXCEPT } ![2] = \text{“Tokens”}]$
$\land$ UNCHANGED $\langle MessageLog$, $HonestBrowser$, $HonestServerState$,

$$HonestSecretKey, AttackerSecretKey, AttackerState, AttackerBrowser, stack, req, resp\rangle$$

$$
\begin{aligned}
Tokens \triangleq\ & \land pc[2] = \text{"Tokens"} \\
& \land AttackerServerState' = [AttackerServerState \text{ EXCEPT } !.Tokens = AttackerSecretKey.secret] \\
& \land AttackerBrowser' = [AttackerBrowser \text{ EXCEPT } !.Headers = AttackerServerState'.Tokens] \\
& \land \ \land req' = [req \text{ EXCEPT } ![2] = ServerRequest(AttackerBrowser')] \\
& \quad \land resp' = [resp \text{ EXCEPT } ![2] = Response(ServerRequest(AttackerBrowser'), HonestServerState)] \\
& \quad \land stack' = [stack \text{ EXCEPT } ![2] = \langle[procedure \mapsto \text{"LogMessagePair"}, \\
& \qquad\qquad pc \mapsto \text{"Done"}, \\
& \qquad\qquad req \mapsto req[2], \\
& \qquad\qquad resp \mapsto resp[2]]\rangle \\
& \qquad\qquad \circ\, stack[2]] \\
& \land pc' = [pc \text{ EXCEPT } ![2] = \text{"Log"}] \\
& \land \text{UNCHANGED } \langle MessageLog, HonestBrowser, HonestServerState, \\
& \qquad HonestSecretKey, AttackerSecretKey, AttackerState\rangle
\end{aligned}
$$

$$CrossSiteRequest \triangleq CSR \lor HonestToAttacker \lor Session \lor Tokens$$

Allow infinite stuttering to prevent deadlock on termination.

$$
\begin{aligned}
Terminating \triangleq\ & \land \forall self \in ProcSet : pc[self] = \text{"Done"} \\
& \land \text{UNCHANGED } vars
\end{aligned}
$$

$$
\begin{aligned}
Next \triangleq\ & SiteRequest \lor CrossSiteRequest \\
& \lor (\exists self \in ProcSet : LogMessagePair(self)) \\
& \lor Terminating
\end{aligned}
$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

$$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{"Done"})$$

END TRANSLATION

* Modification History
* Last modified Sun *Apr* 23 12:13:36 *EDT* 2023 by *Cade Chabra*
* Last modified *Fri Apr* 21 11:11:49 *EDT* 2023 by *andyking*
* Created Sat *Mar* 25 15:05:39 *EDT* 2023 by *andyking*