Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski socialni sklad

SETCCE

Namestitev SICAS PS

Navodila za namestitev SICAS ponudnika storitev

Identifikacijska oznaka dokumenta:	
Različica dokumenta:	1.5
Avtorji dokumenta:	Jurij Zelič
Status dokumenta:	Javni
Zadnja sprememba: dokumenta:	9.12.2016

VSEBINA IN PRAVICE

PODATKI O DOKUMENTU

Osnovni podatki o dokumentu

Dokument	
Identifikacijska oznaka	
Naslov	Namestitev SICAS PS
Različica	1.5
Tip	navodila
Avtorji	Jurij Zelič
Odgovorni	Svetlana Šaljić
Datum nastanka	22.5.2014
Datum zadnje spremembe	9.12.2016
Status	Javni
Datoteka	SETCCE_SICAS_MJU_NamestitevPS.docx
Projekt	SICAS

Nadzor sprememb

Verzija	Kratek opis sprememb	Avtorji	Datum
0.1	Začetna verzija	Jurij Zelič	22.5.2014
0.2	Spremenjeni OID na registrirano vrednost Popravljen OID na SETCCE OID Dodana navodila za inštalacijo na Ubuntu Linux	Jurij Zelič	29.7.2014
0.3	Urejanje dokumenta	Jurij Zelič	21.11.2014
1.0	Nov status atributa, osvežen dokument	Jurij Zelič	11.5.2015
1.1.	Nov atribut sices_id	Jurij Zelič	7.7.2015
1.2.	Nov atribut »celoten certifikat« Popravljeni OIDji za atribute ki zadevajo PI	Jurij Zelič	23.9.2015
1.3.	Dodan opcijski return URL po logout	Jurij Zelič	9.11.2015
1.4.	Dodana navodila za uporabo SICAS mockup	Jurij Zelič	20.11.2015
1.5.	Single logout, dodatni opisi	Jurij Zelič	9.12.2016

Odobritve dokumenta

Oseba	Aktivnost	Podpis	Datum
Aleš Pelan	Odobril		

POVZETEK IN NAMEN

Namen dokumenta

Dokument je namenjen administratorjem in programerjem ponudnikov storitev, ki želijo svojo aplikacijo nadgraditi z zanesljivo avtentikacijo uporabnika, ki do njihove storitve dostopa z brskalnikom, preko sistema SICAS.

Povzetek dokumenta

V Uvodu so opisani osnovni koncepti delovanja in naštete podprte platforme pri ponudniku storitev
Prvo poglavje opisuje postopek inštalacije na nekaterih podprtih platformah.
Drugo poglavje opisuje dostopanje do pridobljenih atributov v nekaterih od spletnih tehnologij.

KAZALO

KRATICE

Seznam uporabljenih kratic

Kratica	Pomen	Opis
AJP	Apache JServ Protocol	Protokol za posredovanje zahtevkov aplikacijskemu strežniku
CAS	Central Authentication Server	Centralni strežnik za avtentikacijo, protokol za dostop do CAS strežnika
CGI	Common Gateway Interface	Način generiranja dinamičnih vsebin za spletne strani
HTTP	Hypertext Transfer Protocol	
IDM	Identifikacijski mehanizem	
MIM	Men In the Middle attack	
OID	Object Identifier	
OS	Operacijski sistem	
PA	ponudnik atributov	
PEM	Privacy-enhanced Electronic Mail	
PI	ponudnik identitete	
PS	ponudnik storitve	
RHEL	RedHat Enterprise Linux	Linux distribucija firme RedHat
SAML 2.0	Security Assertion Markup Language ver. 2.0	Protokol za avtentikacijo in pridobivanje atributov (AA) OASIS organizacije (verzija 2.0)
SICAS	kratica za ime naročila/sistema	Centralni avtentikacijski sistem
SICES		Centralni podpisni strežnik
SOAP	Simple Object Access Protocol	Protokol za implementacijo spletnih storitev
URL	Uniform resource locator	string, ki opisuje spletni naslov
US	uporabnik storitve	
YUM	Yellowdog Update Modified	Odprtokodni sistem za upravljanje s paketi na RedHat baziranih Linux distribucijah

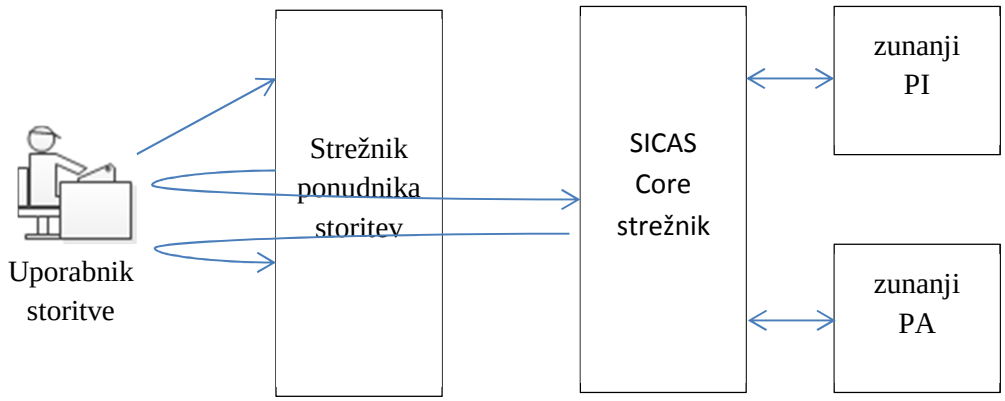
1. UVOD

SICAS je centralni avtentikacijki sistem, ki poleg zanesljive avtentikacije omogoča tudi pridobivanje atributov o avtenticiranem uporabniku storitve iz več virov.

1.1. Zgradba SICAS rešitve

V procesu avtentikacije uporabnika storitve, strežnik ponudnika storitev in SICAS Core strežnik nikoli ne komunicirata neposredno, ampak vedno preko preusmeritev uporabnikovega brskalnika. SAML 2.0 sporočila, ki nosijo podatke o uporabniku storitve in ponudniku storitev se prenašajo v parametrih http (običajno HTTP_POST) sporočil. Strežnik ponudnika storitev in SICAS Core strežnik neposredno izmenjujeta le metapodatke potrebne za delovanje sistema (na primer certifikat za podpisovanje SAML 2.0 sporočil), če je tak izbrani način izmenjave teh podatkov.

Proces avtentikacije se začne, ko uporabnik preko svojega brskalnika pride na stran ponudnika storitve oziroma pritisne »login« povezavo na strani ponudnika storitve, oziroma kako drugače dostopa do ščitene vsebine.



Slika 1– Zgradba sicas rešitve

HTTP zahtevek prestreže shibboleth SP daemon, sestavi SAMLAuthRequest sporočilo in ga preko preusmeritve uporabnikovega brskalnika prenese na SICAS Core strežnik.
SICAS Core strežnik avtenticira uporabnika in pridobi zahtevane atribute o uporabniku. Ob koncu procesa avtentikacije sestavi SAMLAuthResponse sporočilo, ki ga preko preusmeritve uporabnikovega brskalnika prenese na strežnik ponudnika storitev.
shibboleth SP daemon preveri podpis v SAMLAuthResponse sporočilu, izlušči atribute in posreduje zahtevek aplikaciji ponudnika storitve.

1.1.1. Zagotavljanje varnosti in konsistentnosti podatkov

Za preprečevanje MIM napada ali lažnega predstavljanja je med vsemi subjekti zahtevana uporaba varnih (https) protokolov. Vsa SAML 2.0 sporočila so podpisana, s čimer se zagotavlja istovetnost pošiljatelja sporočila. Za zagotavljanje varnosti občutljivih podatkov so atributi v SAML sporočilu kriptirani.

1.2. Podprte platforme pri ponudniku storitev

Avtentikacija preko SICAS uporablja SAML 2.0 protokol, tako da jo je možno implementirati na vsaki platformi. Z uporabo Shibboleth SP odprtokodnega projekta pa so podprte naslednje platforme:
Operacijski sistemi:

- Red Hat Enterprise in CentOS 5, 6
- Ubuntu 12.04 LTS in 14.04 LTS
- SUSE Linux Enterprise Server 10, 11, 11-SP1, 11-SP2, 11-SP3
- OpenSUSE Linux 12.1, 12.2, 12.3
- Windows XP SP2 in kasnejši
- Windows 2003 Server SP1 in kasnejši
- Windows 2008 Server
- Windows 2012 Server
- Na voljo so tudi inštalacije za MAC OS in Solaris

Web strežniki:

- Apache 2.X
- IIS 5 – 8

Pričujoči dokument opisuje predvsem inštalacijo na RHEL (CENTOS) 6 in Windows ter Apache web strežniku.

2. INŠTALACIJA IN KONFIGURIRANJE SHIBBOLETH SP

2.1. Inštalacija Shibboleth SP

Shibboleth SP je odprtokodni produkt, ki omogoča enostavno integracijo SAML 2.0 avtentikacije v že obstoječe spletne storitve. Podpira izredno širok nabor operacijskih sistemov, spletnih in aplikacijskih strežnikov ter spletnih tehnologij. Inegracija Shibboleth SP od osebja ponudnika storitev ne zahteva specialnih znanj, razen osnovnih konceptov delovanja SAML 2.0 protokola.

2.1.1. Windows strežnik

Ustrezno inštalacijsko datoteko si prenesemo iz spletne strani: http://shibboleth.net/downloads/service-provider/latest/win32/ za 32-bitni in iz http://shibboleth.net/downloads/service-provider/latest/win64/ 64-bitni OS. Na isti mapi dobimo tudi morebitne patche.

Inštalacija skreira Service z imenom »Shibboleth 2 Daemon (Default)«.

Shibboleth SP logi se privzeto zapisujejo v mapo `…\shibboleth-sp\var\log\shibboleth`

2.1.2. Linux YUM

Za distribucije, ki uporabljajo YUM (RHEL in CENTOS) se shibboleth inštalira s pomočjo tega orodja:

Najprej dodamo YUM repozitorij:

```
cd /etc/yum.repos.d
wget
http://download.opensuse.org/repositories/security://shibbo
leth/CentOS_CentOS-6/security:shibboleth.repo
```

Nato inštaliramo ustrezno shibboleth distribucijo

```
bash> yum install shibboleth
```

za 32-bitne in

```
bash> yum install shibboleth.x86_64
```

za 64-bitni OS.

Potrebno je le še pognati service shibd

Shibboleth SP logi se privzeto zapisujejo v mapo `/var/log/shibboleth`

2.1.3. Linux APT

Za distribucije, ki uporabljajo RPM (Ubuntu) se shibboleth inštalira s pomočjo tega orodja.
Najprej dodamo prenesemo ključ repozitorija:

```
bash> wget http://pkg.switch.ch/switchaai/SWITCHaai-
swdistrib.asc
```

Preverimo ključ

```
bash> gpg --with-fingerprint  SWITCHaai-swdistrib.asc
```

Rezultat preverjanja mora biti kluč »294E 37D1 5415 6E00 FB96 D7AA 26C3 C469 15B7

In ga dodamo v apt repozitorij

```
bash> apt-key add SWITCHaai-swdistrib.asc
```

Repozitorij dodamo v apt na Ubuntu 12.04 (verzijo preverimo z ukazom lsb_release –a):

```
bash> echo 'deb http://pkg.switch.ch/switchaai/ubuntu
precise main' | sudo tee /etc/apt/sources.list.d/SWITCHaai-
swdistrib.list > /dev/null
```

oziroma na Ubuntu 14.04:

```
echo 'deb http://pkg.switch.ch/switchaai/ubuntu trusty
main' | sudo tee /etc/apt/sources.list.d/SWITCHaai-
swdistrib.list > /dev/null
```

Osvežimo apt repozitorij:

```
bash> apt-get update
```

In naložimo shibboleth SP:

```
bash> apt-get install shibboleth
```

Potrebno je le še pognati service shibd

Shibboleth SP logi se privzeto zapisujejo v mapo `/var/log/shibboleth`

Inštalacijo preverimo z ukazom shibd –t. Včasih moramo ključe za podpisovanje ročno zegnerirati v mapo /etc/shibboleth:

```
bash> openssl req -x509 -nodes -days 3650 -newkey rsa:2048
-keyout /etc/shibboleth/sp-key.pem -out /etc/shibboleth/sp-
cert.pem
```

Oziroma namestimo svoje ključe.

Na ostale operacijske sisteme naložimo Shibboleth SP po navodilih na https://www.switch.ch/aai/docs/shibboleth/SWITCH/2.5/sp/deployment/.

2.2. Konfiguriranje

2.2.1. Apache WEB strežnik

Kot primeri za apache konfiguracijo so na mapi …./shibboleth-sp/etc/shibboleth pripravljeni primeri konfiguracijskih datotek (apache.config, apache2.config, apache22.config, apache24.config), ki jih lahko prirejene za svoje potrebe z Import uvozimo v glavno konfiguracijsko datoteko Apache strežnika (httpd.conf). Glavni elementi so:

- Naložiti je potrebno modul mod_shib:

  ```
  LoadModule mod_shib C:/opt/shibboleth-sp/lib/shibboleth/mod_shib_24.so
  ```

  Pri tem je potrebno paziti na pravo verzijo modula za pravo verzijo Apache (npr mod_shib_24.so za Apache 2.4 in mod_shib_20 za Apache 2.0)
- UseCanonicalName directiva mora biti postavljena na »On«
- Lokacija, ki jo želimo avtenticirati mora imeti nastavljene opcije

  ```
  <Location "/shibbolethSP/login/">
      AuthType shibboleth
      ShibRequestSetting requireSession 1
      Require valid-user
  </Location>
  ```

- Skonfiguriramo še lokacijo z Shibboleth napakovnimi stranmi:

  ```
  <IfModule mod_alias.c>
      <Location /shibboleth-sp>
          AuthType None
          Require all granted
      </Location>
      Alias /shibboleth-sp/main.css C:/opt/shibboleth-
  sp/doc/shibboleth/main.css
  </IfModule>

  <Location /Shibboleth.sso>
      AuthType None
      Require all granted
  </Location>
  ```

- Ob koncu je potrebno le še kostumizirati napakovne strani v mapi ….\shibboleth-sp\etc\shibboleth\dist

2.2.1.1. Javanski aplikacijski strežniki

Javanski aplikacijski strežniki (Tomcat, JBoss, Jetty) se običajno povezujejo z Apache http strežnikom preko AJP 1.3 protokola. Pri konfiguriranju AJP 1.3 konektorja na aplikacijskem strežniku je potrebno nastaviti:

- packetSize je potrebno nastaviti na največ kar omogoča AJP standard - 65536 (nastavitev je potrebna tudi v mod_jk oziroma mod_proxy_ajp konfiguraciji).
- pri uporabi mod_proxy_ajp je potrebno vsem atributom dodati prefiks »AJP_«. To naredimo tako da v shibboleth2.xml datoteki, katere vsebina je opisana v poglavju 2.2.3, v elementu »ApplicationDefaults« dodamo atribut attributePrefix="AJP_":

```
<ApplicationDefaults entityID="SICAS_TestSP-qaa2-par2" ….
attributePrefix="AJP_">
```

Prefiks je potrebno dodati tudi imenom parametrov v poglavju 2.2.3.4.

2.2.2. IIS WEB strežnik

IIS konfiguracijo za osnovno delovanje uredi že inštalacija Shibboleth SP (če se tako odločimo v procesu onštalacije). Potrebno le še kostumizirati napakovne strani v mapi ….\shibboleth-sp\etc\shibboleth\dist

2.2.3. Shibboleth SP

Glavna konfiguracijska datoteka shibboleth SP je …./shibboleth-sp/etc/shibboleth/shibboleth2.xml. V njej je potrebno skonfigurirati:

- Element »ApplicationDefaults« atribut »enityID« je potrebno nastaviti na entityID, dogovorjen ob registraciji ponudnika storitev pri SICAS. Na primer:

```
<ApplicationDefaults entityID="SICAS_TestSP-qaa2-par2" ….>
```

- Element »SSO« (znotraj elementa »ApplicationDefaults«) je potrebno nastaviti na:

```
<SSO entityID="SICAS">
    SAML2 SAML1
</SSO>
```

- Potrebno je narediti datoteke z metapodatki in zagotoviti varno izmenjavo teh datotek med SICAS Core strežnikom in strežnikom ponudnika storitev – glej poglavje 2.2.3.1.
- Potrebno je nastaviti ustrezno dolžino seje (v sekundah):

```
<Sessions lifetime="1200" timeout="1000" relayS…
```

  Parameter lifetime predstavlja dolžino seje (v sekundah). Po tem času se bu uporabnik ponovno avtenticiral. Parameter naj ne bo večji od dolžine seje na SICAS srežniku (30 min). Priporočljiva največja vrednost je 1500 (25 min). Parameter timeout predstavlja najdaljšo uporabnikovo neagtivnost (v sekundah). Vrednost 0 pomeni, da se neaktivnost uporabnika ne preverja.

2.2.3.1. Izmenjava metadata podatkov

Pred priklopom ponudnika storitev je potrebno zagotoviti izmenjavo metadata podatkov med ponudnikom storitev in SICAS Core stražnikom. Metadata podatke se izmenjuje v obliki XX_metadata.xml datotek. Datoteke se lahko izmenjujejo ciklično preko https protokola, ali na kakšen drug varen način, s tem da je potrebno prenesti datoteko ponudnika storitev na datotečni sistem SICAS Core strežnika in obratno.
Način izmenjave in pot do metadata datoteke SICAS Core strežnika skonfiguriramo znotraj »ApplicatinDefaults« elementa shibboleth2.xml datoteke v elementu »MetadataProvider«.

Primer če pridobivamo SICAS Core metadata datoteke ciklično preko https:

```
<MetadataProvider type="XML" reloadInterval="300"
backingFilePath="C:/opt/shibboleth-sp/etc/shibboleth/SICAS-
metadata.xml"
uri="https://sicas.si/idp-metadata.xml"/>
```

Primer če smo datoteko predhodno prenesli na datotečni sistem strežnika ponudnika storitev:

```
<MetadataProvider type="XML" file="C:/opt/shibboleth-
sp/etc/shibboleth/SICAS-metadata.xml "/>
```

Pomen artributov:

- reloadInterval: perioda prenašanja datoteke iz SICAS Core strežnika (v sekundah)
- backingFilePath lokalna datoteka v katero se shranjuje metadata datoteka iz SICAS Core strežnika
- uri naslov na katerem je dostopna metadata datoteka SICAS Core strežnika
- file metadata datoteka SICAS Core strežnika (če smo jo prenesli ročno)

2.2.3.2. Vsebina metadata datoteke

Datoteka …./shibboleth-sp/etc/shibboleth/example metadata je dobra osnova za izdelavo metadata datoteke ponudnika storitev (ki jo moramo kasneje na ena ali drug način prenesti na SICAS Core strežnik).
V datoteki je potrebno skonfigurirati:

- entityID je je potrebno nastaviti na vrednost, dogovorjeno ob registraciji ponudnika storitev pri SICAS
- X509Certificate je potrebno napolniti z javnim ključem s katerim za podpisovanje SAML sporočil (glej poglavje 2.2.3.3)
- Popravimo »Location« atribute vseh Service elementov, da kažejo na URL pri ponudniku storitev
- Izpolnimo element Organization

AttributeConsumingService elementa kot spisek zahtevanih atributov ne uporabljamo, ker je spisek atributov na podlagi tega elementa premalo zanesljiv.

2.2.3.3. Ključi za podpisovanje

Self signed ključi za podpisovanje SAML 2.0 sporočil se zgenerirajo ob inštalaciji Shibboleth SP komponente. Zgenerirani ključi so odloženi v mapi …./shibboleth-sp/etc/shibboleth v datotekah:

- sp-cert.pem – javni ključ
- sp-key.pem – zasebni ključ

Datoteki sta v PEM formatu (Unicode datoteka, ki vsebuje BASE64 kodiran ključ) in jih je mogoče odpreti s priljubljenim urejevalnikom teksta.
Če bi zaradi kateregakoli razloga želeli ključa nadomestiti s svojim je dovolj, da se nadomesti obe datoteki in javni ključ prenese tudi v metadata.xml datoteko, ki jo je potrebno prenesti na SICAS Core strežnik.

2.2.3.4. Mapiranje atributov

Atributom, do katerih smo upravičeni moramo zmapirati govoreče ime, kar naredimo v …./shibboleth-sp/etc/shibboleth/attribute-map.xml datoteki. Primer datoteke za atribute sicas_token, sicas_ime in sicas_priimek je:

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Attribute name="urn:oid:1.3.6.1.4.1.44044.1.1.1.7"
id="sicas_ime"/>
    <Attribute name="urn:oid:1.3.6.1.4.1.44044.1.1.1.8"
id="sicas_priimek"/>
    <Attribute name="urn:oid:1.3.6.1.4.1.44044.1.1.3.1"
id="sicas_token"/>

</Attributes>
```

Vsak atribut je določen z njegovim OID. OIDji vseh atributov imajo vrednost 1.3.6.1.4.1.44044.1.1.1.X, njihovih metadata podatkov pa 1.3.6.1.4.1.44044.1.1.2.X.
X je drugačen za vsakega od atributov:

Tabela 1 mapiranje atributov

X	Pomen
1	EMŠO številka
2	Davčna številka
3	Davčna številka organizacije
4	ZZZS številka
5	Država identifikacije
6	Ime
7	Priimek
8	Naziv
9	Naslov
10	Spol
11	Datum rojstva
12	Kraj rojstva
13	Državljanstvo
14	Naslov elektronske pošte
15	Telefonska številka
16	Organizacija
17	Organizacija zastopanja
18	Identifikator državnega organa?

Metapodatek atributa se pošilja za vse atribute, ki so vsebovani v izbrani politiki in ima lahko eno od vrednosti:

- T - vrednost atributa, ki pripada metapodatku je bila pridobljena iz zanesljivega vira
- F - vrednost atributa, ki pripada metapodatku je ročno vnesel uporabnik
- E – pri pridobivanju vrednosti atributa iz zanesljivega vira je prišlo do napake

Atributi, za enolično identifikacijo uporabnika imajo OID:

1.3.6.1.4.1.44044.1.1.3.1. Token (String ki je enoličen za uporabnika, ne glede na izbrani identifikacijski mehanizem)
1.3.6.1.4.1.44044.1.1.3.2. Identifikator uporabljenega identifikacijkega mehanizma za tega uporabnike (String ki je enoličen za uporabnika, izbrani identifikacijski mehanizem in izbrano identiteto pri ponudniku storitev)
1.3.6.1.4.1.44044.1.1.3.3. Uporabljeni identifikacijski mehanizem
1.3.6.1.4.1.44044.1.1.3.4. Identifikator uporabnika pri SICES – String, ki ga pridobimo pri SICAS-IDP in ga ne uporabljamo.
1.3.6.1.4.1.44044.1.1.3.5. Celoten certifikat (atribut je možno pridobiti le, če je uporabnik izbral avtentikacijo z X.509)

Tabela 2 identifikacijski mehanizmi

IDM	Pomen
SICAS-PWD	SICAS prijava z geslom
SICAS-SMS	SICAS prijava preko SMS
SICAS-KDP	SICAS prijava z digitalnim potrdilom
KDP-SIGOV	Prijava z digitalnim potrdilom - Overitelj SIGOV
KDP+PK-SIGOV	Prijava z digitalnim potrdilom - Overitelj SIGOV na pametni kartici
KDP-SI	Prijava z digitalnim potrdilom - Slovenski kvalificirani overitelji
KDP+PK-SI	Prijava z digitalnim potrdilom - Slovenski kvalificirani overitelji na pametni kartici
KDP-EU	Prijava z digitalnim potrdilom - Evropski kvalificirani overitelji
KDP+PK-EU	Prijava z digitalnim potrdilom - Evropski kvalificirani overitelji na pametni kartici
NDP	Prijava z digitalnim potrdilom - Poljubni overitelji priznani v brskalnikih
NDP ZZZS	Prijava z digitalnim potrdilom - Overitelj ZZZS
STORK-L1	STORK - Raven 1
STORK-L2	STORK - Raven 2
STORK-L3	STORK - Raven 3
STORK-L4	STORK - Raven 4
GOOGLE	Google
FACEBOOK	Facebook
SICAS-PWD	SICAS prijava z geslom

2.3. Uporaba več politik pri istem ponudniku storitev

Kadar pri istem punudniku storitev (na istem strežniku) želimo uporabiti različne politike (politika je seznam prijavnih mehanizmov + seznam povpraševanih atributov) moramo najprej aplikacije (ali dele aplikacij), ki se avtenticirajo z različnimi politikami razdeliti na različne lokacije (različne url-je). Na primer:
Implementacija z uporabo različnih navideznih strežnikov:
https://ps.si – del aplikacije, ki ne zahteva avtentikacije
https://ps1.si – del aplikacije, ki zahteva avtentikacijo s politiko 1
https://ps2.si – del aplikacije, ki zahteva avtentikacijo s politiko 2
ali
https://ps.si – del aplikacije, ki ne zahteva avtentikacije
https://ps.si:444 – del aplikacije, ki zahteva avtentikacijo s politiko 1

https://ps.si:445 – del aplikacije, ki zahteva avtentikacijo s politiko 2

Implementacija z uporabo različnih lokacij:
https://ps.si – del aplikacije, ki ne zahteva avtentikacije
https://ps.si/ps1 – del aplikacije, ki zahteva avtentikacijo s politiko 1
https://ps.si/ps2 – del aplikacije, ki zahteva avtentikacijo s politiko 2

! Za vsako politiko moramo zagotoviti svojo xx_metadata.xml datoteko z drugačnimi url naslovi (Location atribute).
! Datoteka attribute_map.xml naj vsebuje unijo mapinga atributov vseh politik, Za to, da v vsaka aplikacij dobi le atribute, ki ji pripadajo poskrbi SICAS Core.

2.3.1. Implementacija z uporabo ločenih navideznih strežnikov

Navidezni strežniki so web strežniki, ki tečejo na istem http strežniku. Med seboj se razlikujejo po vsej enem od:

- IP naslovu (IP based virtaul hosts)
- portu (port based virtaul hosts)
- URL naslovu (name based virtaul hosts)

2.3.1.1. Konfiguracija Apache web strežnika

Bistveni del konfiguracije navideznih strežnikov je definicija politike, s katero naj se lokacija, ki predstavlja določeni navidezni strežnik avtenticira:
Konfiguracija lokacije, ki se avtenticira (znotraj konfiguracije navideznega strežnika) z privzeto (default) politiko je enaka kot pri primeru z eno samo politiko (primer za uporabo različnih URL naslovov):

```
<VirtualHost *:443>
  SSLEngine on
  ServerName ps1.si:443
  DocumentRoot "${SRVROOT}/htdocssp1"

  ….

  <Location "/">
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    Require valid-user
  </Location>

  ….

</VirtualHost>
```

Konfiguracija lokacije, ki se avtenticira z dodatno politiko, pa vsebuje tudi definicijo te politike:

```
<VirtualHost *:443>
  SSLEngine on
  ServerName ps2.si:443
  DocumentRoot "${SRVROOT}/htdocssp2"

  ….

  <Location "/">
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    ShibRequestSetting applicationId POLICY2
    Require valid-user
  </Location>

  ….

</VirtualHost>
```

2.3.1.2. Konfiguracija Shibboleth SP

Za dodatno politiko definiramo razlike v konfiguraciji napram privzeti (default) politiki tako, da znotraj elementa ApplicationDefaults redifiniramo atribute politike2:

```
<ApplicationDefaults entityID="SICAS_TestSP-qaa2-par1" …. >

  ….

  <ApplicationOverride id="POLICY2" entityID="SICAS_TestSP-
qaa2-par2">
  <Sessions lifetime="1500" timeout="600"
relayState="ss:mem" checkAddress="false" handlerSSL="false"
cookieProps="http"
handlerURL="https://ps2.si/Shibboleth.sso"/>
  </ApplicationOverride>

</ApplicationDefaults>
```

2.3.2. Implementacija z uporabo ločenih lokacij

2.3.2.1. Konfiguracija Apache web strežnika

Pri tem načinu obe (vse) lokaciji definiramo znotraj istega navideznega strežnika (z uporabo direktive Location ali LocatinMatch«:

```
<VirtualHost *:443>
  SSLEngine on
  ServerName ps.si:443
  DocumentRoot "${SRVROOT}/htdocssp"

  ….

  <Location "/ps1/">
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    Require valid-user
  </Location >
  <Location "/ps2/">
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    ShibRequestSetting applicationId POLICY2
    Require valid-user
  </Location >


  ….

</VirtualHost>
```

V primeru z uporabo ločenih navideznih strežnikov je Shibboleth SP pričakoval odgovor od SICAS Core na lokaciji »/Shibboleth.sso«, ker se je lokacija za vsako od politik na vzven maifestirala v različne URL ni bila potrebna dodatna konfiguracija Apache strežnika (če se lokacija »/Shibboleth.sso« le ni prekrivala s kakšno drugo lokacijo). V Našem primeru pa je v konfiguraciji potrebno dodati:

```
<VirtualHost *:443>
  UseCanonicalName On
  SSLEngine on
  ServerName ps.si:443
  DocumentRoot "${SRVROOT}/htdocssp"

  ….
<Location /Shibboleth.sso>
  SetHandler shib
</Location>

<Location /ps2/Shibboleth2.sso>
  SetHandler shib
</Location>

  ….

</VirtualHost>
```

Privzeta politika bo pričakovala odgovor od SICAS Core na privzetem naslovu (https://ps.si/ Shibboleth.sso/…), politika 2 pa na dodatnem (https://ps.si/ps2/ Shibboleth.sso/…). Iste url je potrebno definirati tudi v xx-Metadata datoteki za vsako od obeh politik)

2.3.2.2. Konfiguracija Shibboleth SP

Enako kot v primeru z uporabo različnih navideznih strežnikov moramo za dodatno politiko definiramo razlike v konfiguraciji napram privzeti (default) politiki tako, da znotraj elementa ApplicationDefaults redifiniramo atribute politike2:

```
<ApplicationDefaults entityID="SICAS_TestSP-qaa2-par1" …. >

  ….

  <ApplicationOverride id="POLICY2" entityID="SICAS_TestSP-qaa2-par2">
  <Sessions lifetime="1500" timeout="600" relayState="ss:mem"
checkAddress="false" handlerSSL="false" cookieProps="http"
handlerURL="/ps2/Shibboleth.sso"/>
  </ApplicationOverride>

</ApplicationDefaults>
```

Dodatno pa moramo definirati tudi nov URL, na katerem bo poslušal ShibbolethSP za odgovorom od SICAS Core, kadar smo se avtenticirali s politiko2:

```
<RequestMapper type="Native">
  <RequestMap>
    <Host name="ps.si">
      <PathRegex regex="ps2" applicationId="POLICY2" authType="shibboleth"
requireSession="true"/>
    </Host>
  </RequestMap>
</RequestMapper>
```

3. ATRIBUTI

3.1. Nabor atributov

SICAS Core bo v procesu avtentikacije pridobil nabor atributov, do pridobivanja katerih je upravičen ponudnik storitev. Ti atributi so prenešeni na strežnik ponudnika storitev v SAMLAuthResponse sporočilu.

3.2. Pridobivanje atributov iz aplikacije ponudnika storitev

Shibboleth SP prenese atribute do aplikacija ponudnika storitve tako da jih prenese v spremenjivke okolice HTTP zahtevka (za vsak zahtevek zgenerira svoj nabor spremenljivk) z uporabo CGI mehanizma, kar nam omogoča dostopanje do atributov v praktično vseh obstoječih tehnologijah. Ali pa jih do aplikacije prenese preko glave http zahtevka.

Če izberemo prenašanje atributov od Shibboleth SP proti aplikaciji preko http glave je za izbrano lokacijo (v Apache konfiguraciji) potrebno dodati vrstico »ShibUseHeaders On«, na primer:

```
<Location "/bl-user-web/app/*">
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    Require valid-user
    JkMount ajp13_worker
    ShibUseHeaders On
    Order allow,deny
    Allow from all
</Location>
```

Če uporabljamo CGI mehanizem je v nekaterih tehnologijah kot ime spremenljivke okolice potrebno uporabiti same velike črke in namesto minus znaka (-) pa uporabimo podčrtaj (_)!

3.2.1. Perl skript aplikacije

V Perl hrani spremenlivke okolice v $ENV strukturi. Posamezno spremenljivko lahko preberemo z $ENV{'IME_SPREMENLJIVKE'}.

Primer Perl skripa ki izpiše nekatere atribute (uporaba CGI):

```
#!C:\strawberry\perl\bin\perl.exe

##  prinattributes -- demo CGI program which just prints
its environment

use strict;

use warnings;

print "Content-type: text/plain; charset=iso-8859-1\n\n";

my $emso =  $ENV{'SICAS_EMSO'};

my $ds =  $ENV{'SICAS_DS'};

my $ime = $ENV{'SICAS_IME'};

my $priimek =  $ENV{'SICAS_PRIIMEK'};

print "EMSO: ${emso}\n";

print "DS: ${ds}\n";

print "Ime: ${ime}\n";

print "Priimek: ${priimek}\n";
```

3.2.2. Java aplikacije

Primer je narejen za uporabo http glave za prenos spremenljivk od Shibboleth SP do aplikacije.

```
String ime = request.getHeader("sicas_ime");
```

Če kot privzeto kodno tabelo uporabljamo drugačno tabelo od UTF-8 je potrebno atribut prekodirati

```
String ime = request.getHeader("sicas_ime");
ime = new String( ime.getBytes("ISO-8859-1"), "UTF-8");
```

Pri dostopanju do spremenljivko okolice z uporabo Struts 2 pri neobstoječem atributu metoda getHeader pod določenimi pogoji vrne vrednsot (BigDecimal) 0 namesto null!

3.2.3. ASP aplikacije

Primer je narejen za uporabo http glave za prenos spremenljivk od Shibboleth SP do aplikacije.

```
Set ime = Request.Headers("sicas_ime")
```

3.2.4. .NET aplikacije

Primer je narejen za uporabo http glave za prenos spremenljivk od Shibboleth SP do aplikacije.

```
String[] ime=
 Request.Headers("sicas_ime");
```

3.2.5. Adobe ColdFusion

Vrednost atributov so kodirani z uporabo UTF-8 kodne tabele, vendar so v ColdFusion interpretirani z uporabo ISO-8859-1 tabele, zato jih je pred uporabo potrebno prekodirati:

```
<cfset ime =
charsetEncode(toBinary(toBase64(CGI.sicas_ime,"iso-8859-
1")),"utf-8")>
```

3.3. Odjava

Link za odjavo uporabnika (logout link) je <base url>/bl/logout (na primer https://sicas.setcce.si/bl/logout).
S klikom na ta link se uporabniku prikaže vse aktivne prijave pri vseh ponudnikih storitev in se mu ponudi, da se iz njih odjavi.
Če želimo, da se uporabnik odjavi iz naše aplikacije, ne da bi se mu to ponudilo v URL dodamo atribut eid=«naš entity id«. Na primer https://sicas.setcce.si/bl/logout?eid=POLICY1.
Po odjavi, če uporabnik pritisne gumb »Potrdi«, se uporabnikov brskalnik preusmeri na stran, kjer je pritisnil link do logout strani. Če želimo, da se browser vrne kam drugam to lahko naredimo z opcijskim ur atributom »ra«, ki ima vrednsot URLencoded(base64(returenURL)) (na primer https://sicas.setcce.si/bl/logout?ra=aHR0cDovL3d3dy5zZXRjY2Uuc2kv&eid=POLICY1 argument je URLencoded(base64(»http://www.setcce.si/).

4. NASTAVITVE

4.1. Čas trajanja seje

Trajanje seje na SICAS Core je na politiko natančno mogoče nastaviti na 30 minut ali na nič minut. To pomeni, da če bo ponovni zahtevek za avtentikacijo iz strani ponudnika storitev na CAS Core prišel v času, ki je krajši od časa trajanja seje, SICAS Core vrne avtentikacijske podatke in atribute pridobljene v prejšnji avtentikaciji, ne da bi uporabniku prikazal kakšno zaslonsko masko.

Trajanje seje pri ponudniku storitev lahko nastavimo v datoteki shibboleth2.xml na strežniku ponudnika storitev. Nastavljamo lahko parametra lifetime in timeout (oba sta atributa Sessions elementa in ju spet lahko nastavljamo na politiko natančno):
Atribut lifetime definira najdaljši čas seje (v sekundah), atribut timeout pa najdaljši čas neaktivnosti uporabnika (v sekundah). Po izteku prvega izmed teh dveh bo strežnik ponudnika storitev uporabnikov brskalnik spet preusmeril na stran SICAS Core na ponovno avtentikacijo (kjer se bo ponovno avtenticiral, če se je iztekla seja na CAS Core).

Glede na to so smiselne naslednje kombinacije:

1. Seja na SICAS Core je 0 minut, lifetime je dolg več ur, timeout je dolg nekaj minut. V taki kombinaciji se uporabniku ne bo potrebno ponovno avtenticirati, dokler bo aktiven.
2. Seja na SICAS Core vseeno kakšna, lifetime je dolg več ur, timeout je pravtako dolg več ur. V taki kombinaciji se bo uporabnik moral avtenticirati le vsakih nekaj ur.
3. Seja na SICAS Core je 30 minut, lifetime je dolg nekaj minut, timeout je dolg nekaj minut. V taki kombinaciji uporabnikov brskalnik vsakih nekaj minut preusmerjen na SICAS Core brez zaslonskih mask, po približno 30 minut neaktivnosti pa se bo moral uporabnik na CAS Core tudi ponovno avtenticirati

4.2. Izbira ustreznega bindinga

Binding je mapiranje SAML 2.0 sporočil na standardni komunikacijski protokol. Večina bindingov ne zahteva mrežne povezljivosti med strežnikom ponudnika storitev in strežnikom SICAS, ampak SOAP sporočila prenaša preko uporabnikovega brskalnika preko URL atributa (v primeru uporabe HTTP GET metode) ali preko payloada (v primeru uporabe HTTP POST metde). Bindingov, ki zahtevajo mrežno povezljivost med strežnikom ponudnika storitev in strežnikom SICAS se izogibamo, saj te povezljivosti ne moremo vedno zagotoviti, še slabše taka rešitev včasih deluje v šolskem in testnem okolju, ne pa tudi v produkcijskem!

Bindingi, ki jih podpira nek strežnik in URL naslovi vsakega od njih so našteti v metadata datoteki tega strežnika (elementi SingleSignOnService v metadata datoteki PI oziroma AssertionConsumerService v metadata datoteki PS).

Najpogosteje uporabljeni bindigi so:

- HTTP-POST: SAML 2.0 sporočila se prenašajo preko uporabnikovega brskalnika v payloadu POST http sporočil.
- HTTP-POST-SimpleSign: Enako kot http-POST (razlika je v podpisovanju SAML sporočil).
- HTTP-Redirect: Pri tem bindingu vsa komunikacija med strežnikom ponudnika storitev in CAS Core poteka s pomočjo redirect HTTP sporočil. SAML 2.0 sporočila se prenašajo v payloadu ali kot URL atribut, nikoli pa se ne zamenja http metoda (GET v POST ali obratno).
- HTTP-Artefact: Tudi pri tem bindingu se preusmerja uporabnikov brskalnik izključno preko redirect sporočil, posredovanje atributov pa poteka neposredno med strežnikoma ponudnika storitev in SICAS, zato se njena uporaba za priklapljanje na SICAS ne priporoča.
- SOAP in Reverse SAOP (PAOS): Pri tem bindingu se SAML 2.0 sporočila prenašajo znotraj SOAP ovojnice. Bindinga nista primerna za avtentikacijo uporabnika ki dela preko brskalnika zato ju ne uporabljamo.

5. SICAS MOCKUP

SICAS mockup je jar knjižnica namenjena razvijalcem PS aplikacij. Namenjena je simulaciji SICAS v razvojnem okolju, ko ni možen priklop na šolski SICAS strežnik. Uporabiti jo je možno pri razvoju z Java EE (ali na njej temelječih tehnologij, na primer spring, seam, …).

5.1. Integracija SICAS mockup v SP aplikacijo

SICAS mockup je web filter, zato ga definiramo v web.xml datoteki aplikacije:

```
<web-app xmlns="http://java.s...

...

    <filter>

        <filter-name>SicasMockup</filter-name>

        <filter-class>com.setcce.sicas.mockup.SicasMockupFilter</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>SicasMockup</filter-name>

        <url-pattern>/*</url-pattern>

    </filter-mapping>

...

</web-app>
```

Če gradimo aplikacijo s pomočjo maven dodamo odvisnost do SICAS mockupa v pom.xml, sicer pa dodamo jar v končni build na način, kot ga predvideva naše razvojno okolje:

```
  <dependencies>

...

    <dependency>

        <groupId>com.setcce.sicas</groupId>

        <artifactId>mockup</artifactId>

        <version>1.0-SNAPSHOT</version>

    </dependency>

...

</dependencies>
```

Atribute in njihove vrednosti, ki jih želimo dobivati od SICAS mockup konfiguriramo v datoteki sicasMockupConfiguration.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<sicasMockup>

  <attribute
name="sicas_Token">bDk4U3F6NHhhYXdVd3hYcmtPRjdyaFNIdWNraTVBczM2T1Q5QzNuVXR0NGVzV0g4SV
kvVDV0QTJuRkloYTBFZg==</attribute>

  <attribute name="sicas_ime">SmFuZXo=</attribute>

  <attribute name="sicas_priimek">Tm92YWs=</attribute>

</sicasMockup>
```

Za vsak atribut, ki ga želimo prejemati od SICAS mockup naredimo element »attribute«. V name atribut napišemo ime atributa, ki mora biti usklajeno z attribute_map.xml datoteko, kakršno bomo uporabili na produkcijkskem sistemu. Vrednost elementa je base64 kodiran niz, ki ga bomo v aplikaciji prebrali kot vrednost argumenta.

Siacs mockup že vsebuje minimalno privzeto konfiguracijo atributov. Svojo konfiguracijo, ki bo zamenjala privzeto odložimo na mapo, kjer jo bo našel class loader aplikacijskega strežnika:

- Na tomcat je to skupna lib mapa
- Na JBoss moramo narediti nov modul
- Lahko pa ga damo na WEB-INF/classes mapo našega projekta

6. ZAKLJUČEK

6.1. Problemi

Ni znanih problemov.

6.2. Nadaljnje delo

Ne predvideva se dodatnega dela.

7. REFERENCE

Shibboleth: https://shibboleth.net/ stran avtorja Shibboleth SP komponente programom in navodili za inštalacijo in konfiguriranje.

Aktualna in ažurirana navodila za inštalacijo shibboleth SP:
https://wiki.shibboleth.net/confluence/display/SHIB2/Installation