אבטחת מערכות ויישומים ברשת - שיעור 10#

נושא היום: Web SSO (singled sign-on), SAML and OpenID:

SAML:

בארגון יש לרוב מספר מערכות שצריך לעבוד איתן. ברור שלא נוח לדרוש הזדהות וניהול סיסמא מול כל מערכת בנפרד, אך אם רוצים לעבוד באופן נכון יש צורך בסיסמא מול כל אחת מהמערכות הללו. אם נשתמש בסיסמא אחת עבור כל המערכות מולן עובדים, אז בשינוי סיסמא צריך לשנות בכל המערכות. דבר בעייתי נוסף הוא שהסיסמא נמצאת בכמה מערכות. מספיק שתהיה בעיית אבטחה במערכת אחת והסיסמא תחשף – ואז אותם credentials ניתנות לשימוש בשאר המערכות להתחזות. כאשר עובדים בפרט עם סיסמאות סטטיות, שימוש במערכת אחת מולה מזדהים כך ששאר המערכות יוכלו לקבל מאותה אחת אינפורמציה על המשתמש היא צורך קיים בכל ארגון. נושא ה-singel sign-on קיים בעולם ה-information security כבר שנים רבות.

כאשר עוברים לעולם ה-web, לרוב יש לפחות 5 אתרים / מערכות שונות הדורשות הזדהות (דואר אלקטרוני וכו'). בעולם ה-web הבעיה אפילו עוד יותר חריפה. אמנם ניתן לנהל סיסמאות שונות לכל האתרים ולהחזיק קובץ סיסמאות על המחשב (לא הכי בטוח אך אפשרי). הרוב נוטים לא לעבוד כך, אלא משתמשים באותה סיסמא לכל האתרים. הבעיה כאן היא ששיתוף סיסמאות בין אתרים, אז מעבר לבעיית ה-phishing קל מאוד למצוא את ה-user/pass בצורה פשוטה: מעלים אתר עם תכנים אטרקטיביים חינמיים שדורשים הזדהות – ואז כבעלי האתר מקבלים בצורה חוקית user/pass בעלי סבירות רבה שמשתמשים בהם באפליקציות נוספות. אם אלו ממילא user/pass לאפליקציות לא חשובות, אך אם הן משמשות לאפליקציות רגישות כמו דואר אלקטרוני, e-banking וכו', בעלי האתרים ה"רעים" יכולים להיכנס לחשבונות שלהם.

היינו רוצים מצב בו המשתמש מזדהה פעם אחת מול אתר מרכזי המנהל את סיסמאות המשתמשים ועבור כל אחת מהמערכות הלוויניות הקשורות לאותו אתר, כאשר המשתמש עובר אליהם הוא רוצה שהזהות שלו תעבור ברקע כך שהאתר אליו עובר ידע מי הוא, ואותנטיקציה תתבצע מאחורי הקלעים ע"י האתר הראשי מולו הזדהה המשתמש (לדוגמא: הזדהות מול google לשירותים שונים – gmail, google-docs, calendar וכו').

<u>Back office transaction use-case</u>:

אם חברה מעבירה למשל שירות כלשהו לאתר חיצוני כלשהו, נרצה שאותו שירות חיצוני יכיר את זהות משתמשי החברה המרכזית. למשל, שירות לעובדים המאפשר הזמנת ציוד משרדי מאתר חיצוני כלשהו. רוצים להיות מסוגלים להעביר בצורה שקופה למשתמש נתונים על המשתמש מהאתר המרכזי המנהל את אותם attributes (כמו שם, כתובת, כתובת אימייל וכו') אל אתר נותן השירות, שעל בסיס נתונים אלו ינתן השירות. דוגמא נוספת, כאשר נכנסים למערכת כמו Gmail, אם אין session פתוח מי שמבצע את ההזדהות הוא לא gmail אלא אתר הזדהות מרכזי של google. כאשר הוא מזהה את המשתמש הוא מעביר את המשתמש עם האינפורמציה על ההזדהות אל gmail יחד עם attribute של כתובת הדואר הלאקטרוני של המשתמש. דרישה נוספת היא שכאשר האתר המספק את השירות החיצוני רוצה לדעת האם מותר לא לאשר פעולה כזו או אחרת יוכל לפנות לאתר המרכזי שהוא יאשר או לא יאשר את הפעולה. כלומר זוהי דרך שכל מערכת לווינית פונה למערכת המרכזית שתהווה access control בבחינת אישור פעולות כאלו או אחרות.

<u>What is SAML</u>:

Security Assertion Markup Language:

Assertion: הצהרה על עובדה כלשהי על נושא כלשהו – יישות כלשהי שלרוב היא אנושית או תהליך. ביחס לאותו נושא יוצרים או מחליפים assertions – הצהרות על אותו נושא. למשל: האימייל של john smith הוא כך וכך – הצהרת attribute על המשתמש john smith.

ה-identity provider הוא מייצר ה-assertions וה-service providers צורכים את אותם הצהרות ופועלים לפיהם. ה-SAML מגדיר את היכולת לייצר assertion ביחס ל:

- זהות של הנושא.
- ה- attribute של הנושא.
- הרשאות הנושא ביחס ליישויות אחרות.

את ה-assertions הללו ניתן להעביר למערכות אחרות. ה-SAML מבוסס XML ומשתמש בו כדי לתאר כיצד מקודדים assertion – כיצד אינפו' מועברת כך שהמקבל יבין מה המייצר התכוון להעביר לו, ורוצים לייצר סטנדרט המאפשר לשתי מערכות שפותחו בנפרד לתקשר אחד עם השני באופן יחיד. כמו כן הוא מגדיר חוקים כיצד מטפלים ב-assertions, שולחים ברשת וכדי.

SAML asserting party (identity provider):

המנהל של המשתמשים במערכת, ה-identity management במערכת, ולכן יודע את האינפורמציה על המשתמשים, יכול לזהותם ולבצע עליהם authentication ולכן יוכל לייצר assertion למשל לגבי איך המשתמש הזדהה בפניו. כמו כן הוא מנהל את האטריביוטים שלו ולכן יוכל לענות על שאלות כאלו או אחרות לגבי פעילויות המותרות או אסורות על משתמש מסויים. תתכן הצהרה הכוללת בתוכה כמה פרטים, למשל attrib של משתמש ואופן ההזדהות שלו כלפי המערכת.

SAML relying party (service provider):

המערכת המתבססת על ה-assertions כדי לספק שירות למשתמש / יישות. השירות מחליט על בסיס ה-assertions האם הוא נותן או לא נותן שירות והאם סומך על אותו משתמש / יישות – זו סמכותו של ספק השירות ולא ספק ה-assertions. יתכן שנותן השירות יקבל אינפורמציה על משתמש ו-attrib שלו, אך לא בהכרח יאשר לו מתן שירות.

SAML Assertions:

ישנם שלושה סוגים של assertions. ישנו סוג בסיסי יותר שהוא statement, וישנם 3 סוגי statements:

- Authentication statement: מבנה בו ה-ID provider נותן ל-service provider מידע: הנושא הזדהה בזמן מסויים באמצעי הזדהות מסויים. למשל: המשתמש ג'ון סמית' הזדהה בשעה X באמצעות שם משתמש וסיסמא.
- Attribute statement: נותנת מידע ספציפי לגבי הנושא. ישנן הגדרות לגבי פורמט ה-XML כיצד להעביר את אותם נתונים. Certificate הוא אמנם לא XML אך הוא כן סוג של הצהרה של attribute – נותן מידע על ה-public key של אתר מסויים. זו הצהרה של CA שמפתח מסויים שייך ליישות מסויימת.
- Authorization decision statement: מתארים מה הנושא מורשה לעשות בתנאים מסויימים, בד"כ כתשובה לשאילתא.

SAML protocol:

ה-statement תמיד צריך להיות מוכל בתוך assertion, ובתוך assertion אחד יכולים להיות כמה statements ולא בהכרח מאותו סוג. בקשת SAML:

- בקשת assertion מסויים.
- העברת user ובקשה להחזיר authentication statements על אותו משתמש, או attrib מסויים.

ה-response תמיד יחזיר assertion עם statements בתוכן. ה-XML מגדיר את מבנה ה-statement, assertion וגם ה-request, response. כיצד מעבירים אותם מאחד לשני?

Bindings: מגדירים כיצד מעבירים את המידע מאחד לשני, וישנה יותר מאופציה אחת להעברת האינפורמציה. הרבה פעמים מקובל להעביר כ-SOAP message על גבי HTTP, אך ישנן אפשרויות נוספות.

Profiles: כיוון שהמנגנון מבוסס על XML, ניתן להרחיב את כל הנ"ל (למשל הרחבת ה-request). וכאשר מאפשרים יותר גמישות מתעוררות בעיות תאימות. על כן מגדירים profiles. אלו מגדירים use cases מסויימים המתארים דיאלוג גנרי בין serive provider ל-id provider, החל מה-assertions הנכללים בבקשות וכלה באמצעי התקשורת. כאשר אלו מוגדרים, ניתן לבצע מבחני compliance לבדיקת תמיכת מוצרים מסויימים ב-profiles מסויימים. מוצרים התואמים לפרופילים מסויימים, כאשר אחד לפרופיל id provider מסויים והשני ל-service provider מסויים, אם אלו בהגדרתם תואמים לאותו פרופיל, מובטח ששני המוצרים יוכלו לעבור אחד עם השני בפרוטוקול SAML.

SAML for web single sign-on (SSO):

ה-SAML משמש כיום המערכות רבות להעברת אינפורמציה ממערכת מרכזית למערכות לוויניות, למשל Google. נכיר חלק מהפרופילים שהוגדרו ל-SSO. כאשר מערכת סומכת על assertion מסויים, זה שקול לכך המשתמש מזדהה בפני המערכת. יתרונות:

- Platform neutrality: ניתן לחבר בין מוצרים של vendors שונים ולחבר ביניהם באופן תואם כיוון שהם תואמים לפרופיל מסויים.
- Loose coupling of directories: מקור יחיד המחזיק את האינפורציה שכולם ניגשים אליו לקבל attrib מסויים, וכך לא צריך לסנכרן בין המידע על המשתמש בין directories שונים.
- Improved online experience for end users: ה-SAML מאפשר SSO ע"י כך שמאפשר למשתמשים להזדהות מול id provider ואז לגשת לשירותים שונים ללא צורך בהזדהות נוספת – נוחות למשתמש. כך מקבלים תמיכה ל-SSO בעולם ה-web.

- Reduced administrative costs for servie providers: הנוחות היא גם לצד ספקי השירות, שנחסך מהם האחריות והאדמיניסטרציה של ניהול משתמשים. לכן, אם ניתן להעביר את האחריות לגורם אחר שמטפל בזהות, אזי מורידים את העלויות הקשורות לניהול האדמיניסטרציה וגם הסיכון מועבר ל-id provider, האחריות עוברת לגורם אחר.
- Risk transference: מעבר האחריות ל-id provider.

Web Browser Profiles in SAML 1.1:

בגרסה זו הגדירו 3-2 פרופילים מרכזיים, אותם ננתח. הפרופילים מתחלקים לשתי משפחות מרכזיות:

- pull model – Browser/artifact profile: ה-assertion לא עובר דרך הדפדפן של המשתמש, ה-idp שולח אל ה-sp איזשהו artifact – פיסת מידע המכילה את ה-uid של ה-assertion שייצר ה-idp ואז ה-sp שקיבל את ה-artifact פונה אל ה-idp ומבקש ממנו את ה-assertion שמתאים ל-artifact הזה.
- push model – Browser/POST profile: ה-assertion, בעיקר זה של ה-authentication נדחף מה-id provider ל-service provider, שלא כתוצאה מבקשה של ה-service provider אלא נדחף אליו – וזאת נעשה מעל HTTP request.

The browser/artifact profile:

ננתח שתי סיטואציות:

- Source-site-first: פרופיל אחד תחת ה-pull.
- Destination-site-first: המשתמש פנה ישירות ל-destination site, והוא שולח אותו ל-idp, הוא יחזיר לו artifact ואז הוא יבקש ממנו שוב assertion כלשהו.

Source site first:

מתאים למשל לסיטואציה של פורטל אירגוני ממנו ניתן להגיע להרבה אפליקציות. משתמש פונה לפורטל הארגוני שלא מכיר את המשתמש כרגע. מערכת ה-access control מזהה שאין session פתוח ולכן מערכת ה-login מבקש למשל user/pass. ה-form עם הנתונים נשלח ל-source site, והגיע ל-authentication authority, והוא מבצע אותנטיקציה ומועבר חזרה לאפליקציה שאליה רצה המשתמש להגיע (redirect). בדף זה יש לינקים לאפליקציות אפשריות שיכול הפורטל הארגוני להעביר את המשתמש. לינקים אלו לא מצביעים ישירות לאותם אפליקציות אלא ל- inter site transfer service – העברה מאובטחת אל ה-destination site. כל פניה ללינק כזה מעבירה את ה-dest site אליה רוצה המשתמש להגיע.

האחרון מייצר assertion לאותו משתמש ויוצר artifact עם uid של ה-assertion וכתובת ה-dest. הוא מבצע redirect למשתמש ומעביר את ה-artifact ל-reciever service. כאשר זה מגיע אליו, הוא מחפש את הפרמטר artifact, יודע לבצע עליו פענוח (כי אלו מקודדים), ומבצע SAML req ממנו אל הרכיב המתאים ב-source site ומבקש את ה-assertion התואם ל-uid ב-artifact. זה נשלח ל-responder, שהוא ב-source site אחראי על קבלת הבקשות ושליחת התשובות. הוא מקבל את ה-assertion ג-SAML resp. הוא יוצר כתגובה למשתמש session cookie ומבצע לו redirect לבקשה עם ה-cookie. כעת יש כבר פניה ל-remote app עם הקוקי המתאים והמשתמש הגיע ליעדו ל-remote app מבלי להזדהות שוב.

Destination site first:

מה קורה כאשר המשתמש לא עובר תחילה דרך ה-source siteאלא פנה ישירות ל-dest site. הוא הגיע אליו בלי session cookie, ולכן הוא מבצע לו תחילה redirectאל ה-Source site. נניח כי אין לו Session cookie ל-source site, אז זה מקפיץ לו מסך login, המשתמש מכניס נתונים והנתונים נשלחים לבדיקה ב-authentication authority. מכאן התהליך חוזר על עצמו כמו קודם עם יצירת ה-artifact וכו'.

אם המשתמש יפנה שוב ל-dest אחר, התהליך יחזור על עצמו למעט כמה שלבים – ה-source לא יבצע תהליך הזדהות שוב כי נפתח כבר cookie session אצל ה-source. כל הפניות הבאות שיעבירו את המשתמש ל-source, יחסך מהן תהליך ה-login.

The Browser/POST profile:

תסריט ה-push (source site first):

אין קשר ישיר בין ה-source site ל-dest site, בניגוד למה שיש ב-pull. כל השלבים הראשונים דומים, עד שיתן הדף המרכזי של האפליקציה. כעת ה-assertion מקודד כ-hidden ב-HTTP form, והוא מועבר לדפדפן של המשתמש. כעת המשתמש נדרש לבצע submit ב-post אל ה-assertion consumer, וכעת הוא מקבל את ה-assertion במלואו ולכן לא יפנה ל-ivp. הוא ייבדוק את ה-assertion ואם הוא תקין הוא ייצור cookie ויפנה את המשתמש ל-remote app. לא ניתן לעבוד כאן ב-redirect אלא ב-post request בגלל מגבלות גודל – יש מגבלות על גודל query string ב-redirect ו-XML הוא מבנה לא קטן מספיק ולרוב יחרוג מאילוצי הגודל. לכן מחוייב להיות מושם כפרמטר של POST. על כן צריך ליצור דף HTML שיכלול form עם hidden parameter הו יושם ה-assertion. ב-artifact יש רק uid של assertion ולכן צריך לבקש מה-idp את האסרשן וזה ב-pull. כאן דוחפים מה-ivp את ה-assertion אל ה-sp.

פרופיל נוסף מ-SAML 2.0 של push: SP initiated: POST->POST binding:

כאן המשתמש מגיע תחילה ל-sp וכל ה-binding מבוסס על POST. כאן אין תקשורת ישירה בין ה-sp ל-idp. המשתמש מגיע ל-sp ללא session cookie. הוא מייצר authentication request ל-idp, ומעביר לו את זה כ-hidden parameter בתוך form - מייצר דך HTML שנשלח לדפדפן עם form. חלק מה-request response protocols. הדפדפן עושה POST ומגיע ל-SSO. הוא מנתח את ה-request, מוצא בו את ה-SAML request מסוג authentication request. אם למשתמש אין session cookie מול ה-idp, הוא נדרש להזדהות. כאשר הזדהה, ה-idp יוצר גם הוא דף HTML עם hidden parameter ב-form שנשלח דרך הדפדפן ל-sp. בלה בלה בלה... אם הכל תקין נפתח session cookie למשתמש. בדוגמא זו ה-binding הוא POST/POST – גם העברת ה-SAML req מה-sp ל-idp ו-SAML resp מה-idp ל-sp.

SAML Security Analysis:

מה המשמעות של ההבדל בין הפרופילים push, pull בנוגע לאבטחת מידע? אם עובדים במודל ה-push זה אולי יותר יעיל מבחינת זה שנחסכת התקשורת בין ה-sp ל-idp אך נכנס פה גורם סיכון משמעותי שהוא ה-end user. הוא יכול לתפוס את ה-assertions העוברים דרך הדפדפן, והוא יכול לבצע מניפולציה. כדי להגן על ה-assertion צריך לחתום דיגיטלית את ה-assertion כי קידוד לא מספיק. אין צורך בהצפנה אלא מספיק לחתום בחתימה דיגיטלית. ה-SAML הוא XML ובעולם זה יש הגדרה ברורה כיצד חותמים דיגיטלית על תכני XML, והפרופיל מגדיר שאם משתמשים ב-push חייבים לחתום את ה-assertion תוך שימוש בסטנדרט חתימה דיגיטלית XMLית (תומכת בסימטריה ואסימטריה).

אם מדובר ב-pull, אין חובה לחתום דיגיטלית, כיוון ששני הצדדים במתקשרים בוטחים אחד בשני. אבל, אם מישהו מאזין לרשת הוא יכול למשל לשנות את ה-response (גם את הבקשה, אך זה פחות חשוב). דבר נוסף חשוב הוא mutual authentication – ה-artifact receiver צריך לדעת שהוא אכן מדבר עם ה-source ולא man in the middle. כיוון שאלו הן HTTP req/resp – יש להשתמש ב-SSL המבטיח mutual authentication על בסיס certificate, וכך מקבלים זיהוי ו-data integrity – וכך מגנים מפני הסיכונים בסיניריו זה.

סיכון נוסף: man in the middle בין ה-sp ל-idp למשתמש. צריך להשתמש ב-artifact חד פעמי, כך שעם קבלתו יינתן assertion ומיד יימחק. כך נמנע מצב של שימוש חוזר ב-artifact שהוקלט ע"י תוקף. אבל, תוקף יכול להקליט ולשבש קצת את ה-artifact. כעת הוא יגיע אל ה-artifact receiver שינסה להשתמש בו וייכשל, אך אז התוקף ישלח אותו ויצליח. כלומר התוקף גונב artifact, משבש את המקורי, גורם למקורי להיכשל ואז התוקף יצליח (מבחינת ה-idp הוא בקשה ראשונית שלו). לכן הפתרון כאן חייב להיות הגנת SSL גם על ערוץ מה-inter site לדפדפן של המשתמש וממנו אל ה-dest site.

ה-SSL מגן רק עד שהדף מגיע לדפדפן, ולכן יש צורך גם בחתימה דיגיטלית כדי למוע מהמשתמש לשנות אותו כשהוא בדפדפן. לכן חתימה דיגיטלית XMLית על ה-assertion תשמור עליו לכל אורך חיי ה-assertion.

כדי למנוע שמשתמש ייקח assertion שקיבל כאשר ביקש לעבור לאתר X וישתמש בו כדי להגיע לאתר Y, הוכנס מנגנון target audience בו מייצר ה-assertion יכול להצהיר על האתרים שעבורם יוצר ה-assertion. אם אתר Y אינו חלק מה-target audience הוא מתעלם מה-assertion.

סיכום:

- כאשר עובדים ב-push חייבים חתימה דיגיטלית על ה-assertion. כאשר עובדים ב-pull צריך ערוץ SSL בין ה-sp ל-idp.

- בנוגע ל-reply, נגביל את ה-assertion בזמן. כדי למנוע שימוש חוזר, ב-pull נמחוק את ה-artifact אחרי בקשה ראשונהף וב-push צריך לשמור היסטוריית artifacts כדי לדעת מה כבר שומש. אלו צריכים להיות שמורים במשך אותו חלון זמן בו ה-assertions תקפים. שימוש ב- target audience.
- עבור הגנה מפני man in the middle צריך להגן על הערוץ בין ה-sp למשתמש ובין המשתמש ל-idp ע"י SSL.
- <u>איום נוסף: DoS</u>: כאשר יש idp ברור כי הוא ה-single point of failure – אם מצליחים להפילו או להפכו ללא זמין, אז משתמשים לא יכולים להיכנס למערכת – לכל האתרים התלויים בו. לכן צריך להשתמש ב-signed request כדי להקשות על התוקף.

<u>SAML security summary</u>: במצגת.

<u>מה SAML לא מגדיר</u>:

- אופן זיהוי המשתמש: ה-SAML מגדיר שה-assertion יתאר איך זוהה המשתמש אך לא מגדיר את האופן בו יש לזהות את המשתמש. כלומר sp צריך לבדוק את הצורה בה משתמש הזדהה.
- הביטחון שיש בהצהרה שנתן ה-idp הוא כביטחון שיש ב-idp עצמו, ולכן אם תוקף יוצר assertions מזוייפים מתוך idp זו בעיה חמורה, אחרת כל מודל הביטחון בכלל המערכת נשבר. דבר נוסף שיש לזכור הוא שאין לשיטה על SAML assertion לאחר היווצרותו לאן יועבר.