אבטחת מערכות ויישומים ברשת - שיעור 6#

נושא היום: Web App. Threat Analysis and Intro to Web App. Security:

בשיעור שעבר עשינו threat analysis על מערכת לדוגמא. כיום נמשיך בזה, אך קודם נרחיב בנושא עליו דיברנו: ניתוח הפגיעויות האפשריות בתשתיות עליהן רץ ה-web app.

Insecure Configuration Management:

ע"פ הדו"ח האחרון של OWASP חשיבות נושא זה גבוהה. ה-Web app. הוא חלק ממכלול תשתיות. התשתית הראשונה היא מערכת ההפעלה שעליה כל שאר המערכות מעליה רצות. כיום נושא זה מורכב כיוון שבמקרים רבים ישנם שרתים הרצים על VM ושם נושא מערכת ההפעלה מביא עמו סיבוכיות נוספת. כיום ישנן VM הרצות ישירות על החומרה ומעליה מערכת ההפעלה; ישנן גרסאות בהן יש מערכת הפעלה שמריצה VM שמריץ מערכת הפעלה – 3 רמות.

- מערכת ההפעלה מממשת access control על משאבי המערכת – שאלת הקצאת הרשאות משמעותית מבחינת אבטחה.
- מעל מערכת ההפעלה ישנו ה-web server (apache...). גם כאן ישנם אספקטים רבים של קוני שיש להם השלכה ישירה על אבטחת המערכת.
- מעל ישנו ה-app. Server וגם לו אספקטים רבים של קוני הנוגעים לאבטחה.
- מעל ישנו ה-framework בו מפתחים את האפליקציה.
- מעל ישנה האפליקציה, אך זו לא הרמה האחרונה. האפליקציה משתמשת הרבה בבסיסי נתונים – ה-data repository של האפליקציה, שהקונפיגורציה שלו כמובן גם כן משפיעה על אבטחת המערכת.

כל סביבת ה-web app היא סביבה המורכבת מהמון מרכיבים, ולכן ניתוח המערכת מהחינת אבטחת מידע לא יכולה להסתמך רק על בחינת הקוד האפליקטיבי, אלא יש צורך בהבנת הסיכונים של המרכיבים השונים. לרוב קוד אותם מרכיבים אינו באחריות מפתח האפליקציה, אך מתאפשרת שליטה על הקוני של אותם רכיבים המשפיעים על אבטחת המידע של המערכת.

Insecure Configuration Managrement:

טעות בקוני של web server יכולה לגרום לבעיית security מערכתית ללא תלות ברמה האפליקטיבית. המרכיבים של התשתית הן לרוב רכיבים מסחריים או public domain, ולכן מכירים אותם גם גורמים עויינים המחפשים דרכים לתקוף. ככל שתשתית שמשתמשים בה יותר פופולרית, כך היא יותר אטרקטיבית לגורמים המחפשים vulnerabilities כדי לתקוף. האחראי על התשתיות השונות שלפני האפליקציה עצמה בחברת פיתוח היא קבוצת התשתיות, והקשר ביניהם לבין קבוצת הפיתוח לרוב לוקה. חלק משמעותי בעבודה שצריכה לעשות קבוצת התשתיות להקשחת אבטחת המערכת קשור בתקשורת בין הקבוצות, ופעמים רבות נושא זה נופל בין הכסאות. ידובר עוד בהמשך.

<u>דוגמאות ברמת התשתית</u>:

- מוצרי תשתית המכילים פגיעויות ולא הותקנו עליהם patches שה-vendor שחרר לתיקון אותן פגיעויות: בכל מוצר יש באופן פוטנציאלי פגיעויות, וכל הזמן הם מתגלים ומשוחררים תיקונים. למי שתהיה גישה למערכת ההפעלה בגלל פגיעות שלא patched, אזי תהיה לו גישה למשאבים במערכת ההפעלה.
- קונפיגורציה לוקה של ה-web server: יש להגדיר לאילו משאבים יתאפשרו גישה ע"י ה-web server כלפי פנים.
- חשיפה מתוך backups: יש צורך לדאוג לאבטחת גיבויים, שכן חשיפתם שקולה לחשיפת המערכת או המידע במערכת. כמו כן ישנם רכיבי תוכנה כמו demos שלא מקפידים בהם על אבטחה. המפתח יתמקד ברצון שהרכיב יעבוד, ופחות דגש על אבטחה. תוקפים יכולים לחשוף בהן פגיעויות אותם ינצלו כדי לפגוע במערכת.
- מערכת ההפעלה מכילה שירותים רבים, ומומלץ לא לאפשר לאפליקציה שירותים שאינה צריכה. אם האפליקציה לא תגדיר את השירותים הנדרשים לה בצורה מסודרת, אדמיניסטרטור המערכת לא יוכל להגביל אותה, וכך להפחית את הסיכונים. תוקפים יכולים להגיע לממשקים אדמיניסטרטיביים מתוך חיבור כמשתמש רגיל, וזאת כיוון שאלו חשופים לweb ומהווים סיכון משמעותי.
- Default accounts: המערכת מגיעה עם משתמש דיפולטי וסיסמא דיפולטית. אם האדמין לא ידאג למחוק אותם ולהחליף את כל ה-default pass, תוקפים ינצלו זאת.
- סיכון אפליקטיבי: פונקציות המאפשרות לאדמיניסטרטור שימוש בכלים שונים, העלולים להחשף לתוקף.
- הודעות שגיאה אינפורמטיביות יתר על המידה.

- קונפיגורצית SSL חסרה או בעייתית: חשוב לקנפג את ה-server side בו יש להגדיר cipher-suite מינימלי שתחתיות לא יהיה ניתן לעבור, וזאת כדי למנוע תקשורת SSL שהיא למעשה לא מאובטחת.

Insecure Conf. Detection:

ישנם כלים היודעים לסרוק את הרשת מבפנים או מבחוץ ויודעים לבדוק שאין default accounts, התעבורה מאובטחת, access control חסום למשאבים חיוניים של המערכת, הכל up-to-date וכו'. אלו משמשים את ה-infrastructure engineers לאיתור בעיות, אך כלים אלו לא נמצאים רק בשימוש אלו, אלא גם בשימוש תוקפים. כלים אלו הם רובם אוטומטיים, ותוקפים פיתחו אותם מעבר לשימוש לצורכי התקפה.

How to protect – hardening:

ההקשחה צריכה להיות זהה עבור כל השרתים וכל ההתקנות של אותו מוצר תשתית באותו ארגון. הארגון צריך ליצור לעצמו guideline של הגנה מאובטחת לכל מוצרי התשתית והוא יהיה בסיס להתקנה אותו מוצר תשתית באותו הארגון.

- Server Hardening: לכל אחד ממוצרי התשתית יש מנגנוני אבטחת מיגע built-in בתוכו. בשלב הראשון יש לקנפג אותם כדי לקבל מוצר תשתית מאובטח. אלו תלויים מערכ ת הפעלה ושרת.
- Access control: מגדירים roles ולכל role את ההרשאות שלו, וכך מגדירים AC ארגוני. לאחר מכן יש לשייך משתמש לבעל תפקיד. לפעמים למשתמש יכולים להיות כמה roles והרשאותיו יהיו איחוד הרשאות שני ה-roles שלו.
- מחיקת default account או החלפת סמאותם מסמאותם ברירת המחדל.
- Logging and alerts.

[אין סיכום מאמצע השעה השניה עד תחילת השלישית]

Unused or shared User Accounts:

- Sharing user accounts: אם אדמינים משתמשים יחד באותו משתמש מתוך מחשבה שאלו אותן הרשאות גם כך, רמת הזהירות של המשתמשים נמוכה יותר. כאשר אין trace על משתמש ספציפי במקום על קבוצת משתמשים, אנשים נוטים להיות פחות זהירים. אם לכל משתמש יש את החשבון שלו ויש מעקב אחר פעולות והמבצעים שלהם, יש סיכוי טוב יותר שהמשתמשים יהיו זהירים ולא ינצלו את ה"מסכה" של ריבוי משתמשים. הפתרון הוא פשוט הגדרת role עם הראשות מסויימות ושיוך כל המשתמשים לאותו role.
- מחיקת חשבונות רדומים: משתמשים לא אקטיביים צריכים להמחק או להחסם. זאת כיוון שתוקפים יכולים לנצל משתמשים אלו, שלעומת חשבונות אקטיביים הם יותר קשים לזיהוי פריצה. משתמש אקטיבי, הרואה למשל את זמני הכניסה האחרונה שלו לחשבון, יכול לזהות כניסה של תוקף דרך החשבון שלו – משתמש רדום לא, כי אין עליו review.
- Public account review: צריך לבצע review על כל המשתמשים כל כמה זמן כדי לזהות משתמשים לא אקטיביים, או לזהות משתמשים לא ידועים – תוקפים יכולים להוסיף לעצמם account למערכת (אם הצליח לחדור לאדמין), והוא יוכל לבצע כל מה שהוא רוצה בלי שמישהו ישים לב. כדי לזהות זאת יש לבצע ניטור כל פרק זמן שהוא. דבר נוסף הוא בדיקת שיוך משתמש ל-role - יתכן משתמש שעובר תפקיד, וה-role שהיה לו לא ימחק אלא רק יתווסף לו ה-role החדש, ויחזיק כוח רב בידי משתמש.

Separation of duties:

- Least priviliges: יש לתת לכל משתמש בדיוק את ההרשאות הדרושות לו ולא יותר. היתרונות: אם בתהליך כלשהו יש פגיעות כלשהי, הנזק הפוטנציאלי שיכול להגרם על ידו מוגבל בהרשאות של אותו תהליך. למשל, תהליך המוגבל לקריאה בלבד, תוקף שהצליח להשתלט על התהליך לא יוכל להגיע לבצע כתיבה.
- Separation of duties: יש להגדיר בעלי תפקידים ולכל אחד לתת את ההרשאות שלו. ע"י חלוקת המשתמשים לבעלי תפקידים מונעים unauthorized / unintentional info disclosure, וניתן למנוע טעויות של משתמשים.

בארגון המשמעות היא אדמין בעל הרשאות שונות לכל שרת והרשאות שונות בין אדמין מערכת ההפעלה למסד הנתונים.

המשך הדוגמא מההרצאה הקודמת: לא יסוכם.