מסמך מדיניות

פיתוח מערכת מאובטחת

גרסה 5.5

מעקב גרסאות

מס"ד	תאריך	עודכן על ידי	תיאור השינויים
2.2	14-11-2013	איליה צ'רניאק	גרסה ראשונית
3.0	22-01-2015	דניאל יופה	עדכון מסמך
3.1	22-01-2015	אברהם זרוק	עדכון מסמך
3.2	12-03-2015	דניאל יופה	עדכון מסמך
3.3	22-07-2015	אודי ברוך	עדכון מסמך
3.4	23-07-2015	יוגב מזרחי	הוספת נושא ה-CAPTCHA
3.5	29-05-2016	רועי ממן	שימוש בספריות צד שלישי (סעיף 3.12)
3.6	27-07-2016	אודי ברוך	מדיניות סיסמאות
4.0	6-10-2016	אופיר יהב	הוספת פרק 1 כללי, הוספת פרק 2 תהליך פיתוח מאובטח, הוספת פרק 3 שלבי פיתוח מאובטח, הוספת נספח אמצעי הגנה על מרכיבי אבטחת המידע,
4.1	25-10-2016	אופיר יהב	הוספת נספח איומי אבטחת מידע, הוספת נספח אמצעי הגנה, אחידות בעימוד, גופן, גודל, צבע
4.2	16-11-2016	אופיר יהב	הרחבות למספר חלקים גדול במסמך.
4.3	27.2.2017	אופיר יהב	הרחבת החלק המתייחס לאפליקציות WEB
4.4	04.04.2017	דני יופה	תיקוף המסמך
4.5	18.5.2017	אופיר יהב	עדכון הראשית והסיומת של כל עמוד
4.6	14.9.2017	אופיר יהב	איסור שימוש בנתוני אמת בסביבת הפיתוח והבדיקות.
4.7	25.9.2017	אופיר יהב	בהעברה לייצור לוודא כי שירותים ופרוטוקולים מיותרים – שאינם נחוצים לפעולת המערכת – סגורים.
4.8	24.10.2017	אופיר יהב	הוספת חובת ביצוע סריקת מנועי אנטי וירוס לכל גרסאות התוכנה לפני העברתן לייצור. חובת ביצוע הקשחות יצרן בנוסף להקשחות הספציפיות של המערכת.
4.9	14.11.2017	אופיר יהב	הוספת חובת תהליך סיווג חומרת ממצאי בדיקות חדירות וסקרי קוד

5.0	22.2.2018	אופיר יהב	תיקון טעויות תחביר, הקלדה ואחרות לפי פירוט במסמך אקסל (ר' ספריית ארכיב של המסמך) אשר שלח אוהד מליחי, תחום בדיקות אפליקציה במערך ההגנה בסייבר.
5.1	1.7.2018	אופיר יהב	הוספת נושא ערפול קוד בפרק 4.6
5.2	17.7.2018	אופיר יהב	תיקון שגיאות הקלדה לאורך המסמך – אותיות חסרות או עודפות – ללא שינוי תכני. הוספת התייחסות למסמך הנחיות ליישום מדיניות של HTTPS-Only.
5.3	3.10.2018	אופיר יהב	מיקום קבצי חיווי (לוג). איסור מיקום בשרתים קדמיים.
5.4	10.12.2018	אופיר יהב	איסור שמירת מפתחות פומביים בשטחי אחסון מחוץ לממשל זמין.
5.5	31.1.2019	אופיר יהב	בהעברת מערכת לבדיקות חדירות יש לצרף מסמך אפיון הכולל לפחות את תיאור הפונקציונאליות של וממשקים חיצוניים של המערכת הנבדקת.

נתוני גרסת המסמך

גורם	תפקיד	שם מלא	תאריך	חתימה
נערכה ע"י	ראש תחום מתודולוגיות הגנה בסייבר	אופיר יהב	31.1.2019	(חתימה)
נבדקה ע"י	ראש תחום בדיקות אפליקציה	רועי ממן	31.1.2019	(חתימה)
אושרה ע"י	מנהל מערך הגנה בסייבר	אברהם זרוק	31.1.2019	(חתימה)

תוכן עניינים

1. כללי

1.1. מסמך זה מפרט את קווי המדיניות לפיתוח מערכות מאובטחות בממשל זמין והוא מיועד לעובדי ממשל זמין ולקוחותיו המארחים את מערכותיהם בתשתיות ממשל זמין.

1.2. מסמך זה כולל הנחיות לפיתוח תוכנה - בנוסח עשה ואל תעשה - שמטרתן פיתוח מערכות מאובטחות אשר תהיינה מוגנות מפני איומי אבטחת מידע ובכך תקיימנה את כל מרכיבי אבטחת המידע של השירות המונגש באמצעותן (זמינות, חסיון ושלמות המידע).

1.3. האחריות לאכוף את הנחיות מסמך מדיניות זה חלה על מפתחי המערכות, ראשי הצוותים ומנהלי הפיתוח.

1.4. הבקרה על אכיפת קווי מדיניות אלו חלה על מערך הגנה בסייבר בממשל זמין.

1.5. יש להתייחס לקווי מדיניות המפורטים במסמך זה כמחייבים. במידה ולא ניתן לתת מענה להנחיה זו או אחרת יש לקבל אישור לכך ממערך ההגנה בסייבר.

2. תהליך פיתוח מאובטח

2.1 מעורבות מערך ההגנה בסייבר

2.1.1. מערך סייבר ואבטחת המידע של ממשל זמין יהיה מעורב בכל שלבי מחזור החיים של פיתוח תוכנה החל משלבי הייזום ודרך אפיון והתקנה, עד לחשיפת האפליקציה לגישה פומבית של המשתמשים בעולם.

2.1.2. פרק זה בא לפרט את הצורך והן את התוצרים, אותם יספקו עובדי מערך הגנה בסייבר בכל שלבי החיים של האפליקציה בממשל זמין.

2.2 מחזור חיי המערכת

2.2.1. על מנת לפתח ולתחזק מערכות מאובטחות תוך הטמעה ויישום מנגנוני אבטחת מידע בדרך נכונה ויעילה, מומלץ לערב את מערך הגנה בסייבר בכל אחד משלבי מחזור החיים של המערכת המאובטחת:

2.2.1.1. **הגדרת דרישות למערכת.**

2.2.1.2. **אפיון ועיצוב המערכת.**

2.2.1.3. **יישום המערכת (כתיבת הקוד).**

2.2.1.4. **בדיקת המערכת והטמעתה.**

2.2.1.5. **העברת המערכת לסביבת היצור.**

2.2.1.6. **עדכון ו/או תחזוקה של המערכת.**

2.2.1.7. **הסרת / החלפת המערכת במערכת חדשה.**

2.2.2. כל אחד משלבים אלו צריך להתבצע באופן מאובטח. מסמך מדיניות זה מקיף את כולם.

2.3 עקרונות הפיתוח המאובטח

בכל שלבי מחזור חיי המערכת צריכים להשמר העקרונות הבאים

2.3.1. **עקרון החוליה החלשה** – הרכיב החלש ביותר אבטחתית קובע את חוזק מערכת ההגנה כולה. נקודת החולשה נקבעת גם על ידי מידת החשיפה לאיום, ולא רק מהאיום עצמו.

2.3.2. **הרשאה מינימאלית** – כל פעולה במערכת צריכה להתבצע בהרשאה מינימאלית הניתנת לפרק הזמן הקצר ביותר. הענקת הרשאות גבוהות מהנחוץ הופכת את המערכת המאובטחת לחשופה ופגיעה יותר.

2.3.3. **פשטות** – עקרון זה הופך את המערכת לצפויה יותר וקלה יותר לבדיקה. מערכת מורכבת קשה יותר לבדיקה, מבחינת אבטחת מידע, מאחר ואבטחת מידע עוסקת באירועים צפוים פחות.
פשטות עוזרת להבנת אופן השימוש, איתור בעיות פוטנציאליות, קלות בתחזוקה והפחתת מקומות פוטנציאליים לטעות.

2.3.4. **מצב ברירת מחדל בטוח** – על המערכת להגדיר מה מותר ולשלול כל פעולה אשר אינה צפויה או אינה עונה על תנאי מוגדר. עדיפה מערכת נעולה עם מידע אמין מאשר מערכת נגישה עם מידע חשוד.

2.3.5. **נקודות גישה מוגבלות ושמורות** – יש להגביל את מספר נקודות הגישה למערכת ולהגן על נקודות אלו.

2.3.6. **הגנת עומק** – על המערכת להבנות על מספר קווי הגנה ולהגן גם על קווי הגנה אלו זאת על מנת למנוע תלות בנקודה קריטית, אפקט הדומינו והתפשטות הנזק. על כל תהליך להיות בנוי משלבים אשר בכל אחד מהם מוטמע מנגנון אבטחה (אימות, מניעת זליגה, רישום לקובץ לוג וכו').

2.3.7. **שימוש ברכיבים קיימים** – הסתמכות על מנגנוני אבטחה ידועים הופכת את המערכת לבטוחה ואמינה יותר. 'המצאת גלגלים' הינה מתכון למערכת חלשה.

2.3.8. **הגבלת החלטות משתמשים** – קבלת החלטות בנושאי אבטחת מידע אינה נתונה לשיקול דעתו של המשתמש.

2.3.9. **השארת ראיות** – על המערכת להשאיר ראיות להתקפה – ראיות אשר תאפשרנה לשקם את הנזק ולאתר את מקור הפגיעה הנחוץ למניעה עתידית.

2.3.10. **הצפנה** – יש לשמור הן על סודיות אלגוריתם ההצפנה והן על סודיות המפתח. יש להקפיד על תכנון אופן שמירת המפתחות והגנת הגישה אליהם. חל איסור על שמירת מפתחות פומביים באתרי אחסון משותפים ושטחי אחסון אחרים אשר נמצאים מחוץ לממשל זמין וזאת מחשש שהמפתחות יהיו נגישים גם בפני גורמים בלתי מורשים.

2.3.11. **פרטיות וחשאיות** – יש למנוע דליפת מידע אודות אמצעי האבטחה המיושמים במערכת ובכך לגרום לתוקף לנוע באיזור לא מוכר.

2.3.12. **הפרדת רשויות** – מערכת הבקרה ומערכות אבטחת המידע צריכות להיות מחוץ להישג ידו של הגורם המבוקר. עקרון זה תקף לגבי קבצי חיווי ואמצעי גילוי ומניעה שונים.

3. שלבי פיתוח מערכת מאובטחת

3.1 הגדרת הדרישות למערכת – Requirements

בשלב זה יש להגדיר גם את דרישות אבטחת המידע הרלוונטיות למערכת בנוסף לדרישות האחרות (ר' גם מסמך מפורט לכתיבת מסמך דרישות לפיתוח מערכת בממשל זמין – אשר הינו נספח למדיניות ניהול הפרויקטים בממשל זמין). דרישות אבטחת המידע צריכות לכלול בין היתר גם את הנושאים הבאים:

3.1.1. שמירה על פרטיות המשתמשים.

3.1.2. שמירה על חיסיון הנתונים.

3.1.3. שמירה על שלמות המידע.

3.1.4. שמירה על אמינות המידע.

3.1.5. שמירה על זמינות המערכת.

3.1.6. שמירה על שלמות תהליכים עסקיים.

3.1.7. יכולת לשחזר אירועי א.מידע.

3.1.8. מניעת התכחשות משתמשים לביצוע פעולות.

3.1.9. זיהוי חזק של משתמשים.

3.1.10. זיהוי חזק בין תת מערכות ומערכות חיצוניות (בסיס נתונים, שירותי רשת וכד').

3.2 שלב אפיון ועיצוב המערכת – Design

3.3.1. בשלב זה יש לתכנן כיצד תתן המערכת מענה לכל אחת מדרישות אבטחת המידע. יש לפרט את מנגנוני אבטחת המידע אשר ימומשו, איומים עיקריים על המערכת ודרכי התגוננות.

3.3.2. לאחר ששלב האפיון מסתיים יש לעשות בדיקת אבטחת מידע (כחלק מה- Design Review) בשיתוף מערך הגנה בסייבר.

3.3.3. בין היתר, מסמך האפיון אמור לתת מענה למימוש המנגנונים הבאים:

3.3.3.1. הצפנת נתונים רגישים.

3.3.3.2. בחירת מנגנון זיהוי משתמשים.

3.3.3.3. תיעוד אירועי אבטחת מידע.

3.3.3.4. תיעוד פעולות משתמשים.

3.3.3.5. תיעוד פעולות תחזוקת המערכת.

3.3.3.6. שיטת ההרשאות ומימושה.

3.3.3.7. מנגנון נעילת משתמשים.

3.3.3.8. מדיניות ניהול משתמשים.

3.3.3.9. ממשקים מאובטחים למערכות נוספות.

3.3.3.10. פירוט של רכיבי צד שלישי אשר יהיו בשימוש.

3.3.4. תכנון מנגנוני הגנה בשלב איפיון ועיצוב המערכת

תכנון מנגנוני הגנה נועד לספק רכיבי בקרה (Controls) המיועדים להתמודד עם האיומים שזוהו בשלב זה.

בדומה לשלב תכנון הפתרון הפונקציונאלי במערכת מידע, לא ניתן לספק "ספר מתכונים" המכיל פתרונות מן המוכן. עם זאת, ניתן למפות את סוגי הפתרונות הידועים, כדי להקל על איתור או פיתוח של מנגנוני הגנה.

ניתן לחלק את כל מנגנוני ההגנה לשלושה טיפוסים עיקריים:

- מניעה
- גילוי
- תגובה

ובתוכם לטיפוסי משנה. הטקסונומיה להלן מנסה לפרט מנגנוני הגנה ברמה מופשטת, אשר אינם אופייניים למערכות מידע בלבד. יש לזכור שמנגנוני ההגנה המפורטים להלן מיועדים למגוון טיפוסי התקפות על מידע:

- סודיות
- שלמות
- נגישות

מנגנוני מניעה מנסים למנוע את התרחשות הנזק למערכת באופן פסיבי. אמצעי מניעה לעיתים קרובות משולבים באמצעים מטיפוסים אחרים כדי לאפשר הגנה אקטיבית. ניתן לחלקם בין אמצעים שתפקידם צמצום סבירות הנזק וכאלה שמיועדים לצמצום היקף הנזק.

לפירוט מנגנוני ההגנה הניתנים למימוש בשלב האפיון ועיצוב המערכת - ר' נספח נפרד בסוף מסמך זה.

3.3.5. ניתוח עלות/תועלת וקביעת קדימויות

לאחר תכנון אמצעי ההגנה למערכת מתבצע ניתוח עלות/תועלת עבור האמצעים השונים. על פי רוב קיימים מספר אמצעי הגנה שנותנים מענה לאיומים שאותרו קודם לכן, ומדיניות האבטחה משמשת לקביעת קדימויות לגבי בחירת אמצעי

מסוים. שיקולים בהעדפת אמצעי אבטחה נובעים מעלות כספית אשר חלה על מימוש הפתרון, עלות ונגישות כוח האדם הנדרש לתחזוקתו, זמן המימוש, עלות הפתרון כנגד עלות האיום וכדומה.

3.3 שלב יישום המערכת – Implementation

3.4.1. בשלב זה מפותחת המערכת ומנגנוני אבטחת המידע אשר תוארו קודם לכן.

3.4.2. כתיבת קוד המקור צריכה להתבצע בהתאם למסמך מדיניות זה.

3.4.3. לאחר סיום מימוש המערכת, מומלץ לבצע סקר קוד (Code Review) על מנת לוודא את יישום דרישות אבטחת המידע.

3.4.4. רצוי שסקר זה יתבצע ע"י מומחה אבטחת מידע אפליקטיבי חיצוני ולא ע"י מתכנתי המערכת, על מנת שהבקרה תתבצע ע"י גורם מומחה בלתי תלוי.

3.4.5. עקרונות מונחים למימוש מנגנוני הגנה

במימוש מנגנוני ההגנה מתבצע מאמץ ליישם את הרכיבים שתוכננו, מבלי לייצר מפגעי אבטחה נוספים. בדומה ליישום הפתרון הפונקציונלי, איכות היישום מושפעת משיטות עבודה מובנות, ומאוסף של "טעויות קלאסיות" מהן מנסה המיישם להימנע. במקרים רבים, מגבלות סביבת היישום (שפה, מערכת הפעלה, רכיבי צד ג') מחייבים תכנון מחודש של חלק ממנגנוני האבטחה.

שיטות העבודה ביישום הפתרון, משותפות לפיתוח הפתרון הפונקציונלי ולפיתוח אמצעי האבטחה. ככלל, יש לבצע:

3.4.5.1. תיעוד מערכת מלא הכולל: רכיבים, מבנה נתונים, נתיבי תקשורות, תיאורים דינמיים ותלויות ברכיבי צד שלישי (מערכות הפעלה,רכיבים מוכנים וכו').

3.4.5.2. התאמה לתכנון, יש לבדוק שמימוש המערכת נעשה לפי הקווים המנחים.

3.4.5.3. בקרת איכות.

3.4.5.4. Code Review- יש לעשות בחינת קוד על ידי מומחה אבטחה.

3.4.6. בחינת מימוש לשגיאות נפוצות

יישום מערכות אבטחה סובל משגיאות טיפוסיות שחוזרות במערכות רבות. קיימים מקורות רבים הסוקרים בעיות בסביבות ספציפיות (בשפות כגון: C, C++,C#, Java , או במערכות הפעלה כגון: WinNT, Unix, OS2 וכדומה) אין כוונה במסמך זה לסקור את הסביבות הספציפיות. הרשימה המובאת להלן סוקרת מספר בעיות נפוצות מבלי להתייחס לסביבת יישום מסוימת:

3.4.6.1. **שגיאות תנאי:**

שגיאות הנוצרות בעקבות בדיקה חסרה או לא מספקת של תנאי זרימה. על פי רוב כתוצאה של בדיקה חסרה של פרמטרים, נתונים שהוקלדו על ידי המשתמש, קודי סטטוס שמוחזרים ממערכת ההפעלה וכן הלאה. שגיאות תנאי גורמות כיום ליותר מ 90% של בעיות האבטחה ביישום מערכות. כל בעיות ה - Buffer Overflow נובעות מבדיקה לא מספקת, או חסרה לחלוטין של הפרמטרים של המערכת.

3.4.6.2. **מצבי אטומיות:**

שגיאות שנוצרות כאשר לא הובאה בחשבון הפרעה בזרימת התוכנית. במקרים רבים קיימת הנחה, שזרימת התוכנית של תופרע. הנחה זו מוטעית לחלוטין במערכות מרובות משתמשים, בעלי משימות מרובות או מספר מעבדים. מצבי Race Conditions, מהווים אחד הגורמים המובילים לבעיות אבטחה. לדוגמא: מערכת שבודקת תאימות של שדה להנחות תחביר, אבל לא מביאה בחשבון שהמשתמש יכול לשנות את השדה בין זמן הבדיקה וזמן הכנסת המידע לבסיס הנתונים.

3.4.6.3. **דלתות אחוריות:**

שגיאות מכוונות שנוצרות כאשר מפתחי המערכת השאירו רכיבי בדיקה או תחזוקה שעוקפים את מערכת האבטחה, או לחלופין חושפים מידע חסוי על המערכת (מפתחות, סיסמאות וכדומה). ברוב המקרים נשארות דלתות אחוריות ללא כוונת זדון מצד צוות הפיתוח, אבל אין להוציא את האפשרות מכלל חשבון.

3.4.6.4. **מצבי קצה:**

סוג של שגיאת תנאי שנוצר כאשר המערכת אינה בודקת תנאי קצה: משאבים מצומצמים (זיכרון, מקום פנוי בדיסק, גישה בעומס גבוה) וכתוצאה פועלת באופן לא צפוי.

3.4.6.5. **גבולות בין רכיבים :**

כאשר מידע זולג מרכיבים מאובטחים לרכיבים ציבוריים, או כאשר קיימת אפשרות גישה בנתיב ההפוך. בעיית גבולות קורה פעמים רבות דרך השימוש במשאב משותף. לדוגמא: כתיבת מידע חסוי

לזיכרון משותף, מחייבת את מחיקתו לפני שהזיכרון יעבור לשימוש רכיב אחר, או ייכתב ל- Swap File .

3.4.6.6. **שגיאות אינטגרציה:**

שגיאות שנובעות מבדיקה חסרה של סביבת היישום. כאשר שילוב המערכת ברכיבי סביבה יוצר מצב לא בטוח (למשל מערכת הפעלה ללא בקרת גישה, רכיבי צד ג' לא בטוחים, רכיבי תקשורת לא בטוחים וכד').

3.4.6.7. **זליגת מידע:**

כאשר המערכת "מנדבת" מידע פנימי ללא צורך שעשוי לסייע לתוקף למקד את ההתקפה. לדוגמא: גרסה, משתמש נוכחי.

3.4 שלב בדיקת המערכת והטמעתה

3.5.1. בשלב זה המערכת מתפקדת באופן מלא ונעשות בדיקות איכות על מנת למצוא תקלות אפשריות באופן פעולתה השוטף. בדיקות איכות אלו נעשות באמצעות צוות ה - QA , אשר מתעד כל תקלה, מעביר אותה למתכנתים ובודק אם תוקנה. לאחר שלב זה, יש לבצע בדיקות אבטחת מידע למערכת כולה. האחריות לבדיקות אלו היא של מערך הגנה בסייבר ומבלי אישורה, המערכת לא תעלה לסביבת הייצור.

3.5.2. ניתן לבצע את בדיקות אבטחת המידע במספר אופנים:

3.5.2.1. **White-Box -** מעבר על קוד המקור של המערכת בהיבטי אבטחת מידע, בדיקת תצורת המערכת, תצורת התשתית ותצורת הרשת. בדיקה זו יסודית ביותר ועלולה לקחת זמן רב (יחסית) במערכות גדולות. היתרון הנוסף של צורת בדיקה זאת היא בכך שלאחר שמתבצע שינוי במערכת, ניתן לבדוק Delta של השינוי בלבד ואין צורך לבדוק את כל המערכת מחדש.

3.5.2.2. **Black-Box** - בדיקה זאת מהווה אינדיקציה על מצב המערכת מנקודת מבטו של הפורץ, היא פחות יסודית מקריאת קוד המקור, אך בדרך כלל אורכת פחות זמן בצורה משמעותית.

3.5.2.3. **Gray-Box** -שילוב של שתי השיטות הנ"ל, בו נבדקים רק חלקי קוד אשר הוגדרו כרגישים במיוחד ושאר המערכת נבדקת כמשתמש. היתרון של צורת בדיקה זו היא ביעילות שלה הן מבחינת זמן והן מבחינת היסודיות.

3.5.3. **דגשים בביצוע הבדיקות:**

3.5.3.1. בכל אחת מהבדיקות יש לקבל דו"ח המתאר את ממצאי אבטחת המידע והמלצות לתיקונם.

3.5.3.2. יש לקיים ישיבה טכנית עם נציג מצוות הפיתוח, לאחר שקרא את הדו"ח, עם יועץ אבטחת המידע שבדק את המערכת. בישיבה זאת יוסברו הממצאים ויוחלט על זמן לתיקונם.

3.5.3.3. לאחר תיקון הממצאים יש לערוך בדיקת אבטחת מידע חוזרת.

3.5.3.4. יש לקבל את אישורו של מנהל אבטחת מידע לצורך העלאת המערכת לסביבת הייצור לאחר ביצוע תיקון של כל הליקויים שנמצאו.

3.5.4. **העברת המערכת לסביבת הייצור:**

את המערכת אשר אושרה על ידי מערך הגנה בסייבר ניתן להעביר לסביבת הייצור תוך כדי הקפדה על מספר כללי יסוד הבאים:

3.5.4.1. יש להחליף את כל הסיסמאות של המערכת למזהים קשים לניחוש על פי מדיניות הסיסמאות של ממשל זמין.

3.5.4.2. יש לדאוג למחיקת כל קוד מערכת מיותר שנועד לצורכי Debug , כגון מחלקות עזר - utility classes – ושאר כלים העוזרים לפתח את האפליקציה ולבצע בדיקות של המערכת.

3.5.4.3. יש למחוק את כל חשבונות המשתמשים אשר היו קיימים במערכת.

3.5.4.4. יש לדאוג להפרדה מלאה בין סביבת היצור לסביבות נוספות (פיתוח, בדיקות וכדומה) ולהשתמש במזהים שונים עבור כל סביבה.

3.5.4.5. חל איסור לעשות שימוש בנתוני אמת בסביבת הפיתוח והבדיקות.

3.5.4.6. חובה להקשיח את המערכת - להשבית (Disable) שירותים מיותרים ולחסום פרוטוקולים מיותרים אשר אינם נחוצים לפעולת המערכת. זאת בנוסף לביצוע כל הקשחות היצרן הנדרשות.

3.5.4.7. חובה לסרוק את קוד המערכת (כל רכיב תוכנה – גרסאות מערכת וכל קובץ תוכנה) במערכות אנטי-וירוס לפני העברת הקוד לתשתיות הייצור. סריקת קוד המערכת מתחייבת גם אם הקוד פותח ונבדק בתשתיות הפיתוח והבדיקות של ממשל זמין.

3.5 שלב עדכון ו/או תחזוקה של המערכת

3.6.1. לאחר שהמערכת פועלת בסביבת הייצור, יש לדאוג לבצע עדכוני א.מידע שוטפים לתשתית המערכת (Patches).

3.6.2. כל עדכון אפליקטיבי במערכת מחייב בדיקה ואישור של מערך הגנה בסייבר. במידה והתבצע Code Review על המערכת, ניתן לבדוק רק את החלק החדש שהשתנה. במידה ולא התבצע Code Review, יש לבדוק את כל המערכת מחדש על מנת לוודא כי רמת אבטחת המידע של המערכת לא נפגעה.

3.6.3. מעבר לכך, יש לבצע בדיקות Black-Box תקופתיות על מנת לוודא שטכניקות פריצה חדשות או ששינויים במערכת לא יכולים לפגוע במערכת ובמשתמשיה.

3.6 שלב הסרת / החלפת המערכת במערכת חדשה

3.7.1. כאשר מעדכנים גרסת מערכת או מחליפים אותה במערכת חדשה, יש לבצע גיבוי מלא למערכת הישנה ולנתוניה על מנת שנוכל לבצע שיחזור במידת הצורך.

3.7.2. יש לתת דגש על שמירת דיוק הנתונים בעת העברתם למערכת החדשה.

3.7.3. יש לשים לב שבשלב המעבר, לא נחשפים נתונים רגישים של המערכת לגורמים פנימיים ו/או גורמים חיצוניים.

3.7.4. את המערכת הישנה ואת נתוניה יש לשמור בגיבוי במקום אשר מאובטח פיסית (כספת / חדר נעול) על מנת למנוע זליגת הנתונים. כמו כן, במידה והמערכת החדשה מותקנת על גבי חומרה חדשה, יש לוודא שהכונן הקשיח עליו הותקנה המערכת הישנה גובה ואחר כך הושמד פיסית.

4. בדיקות אבטחת מידע

שלב בדיקות אבטחת מידע יתבצע כשלב האחרון לפני עליית המערכת לאוויר. בדיקת המערכת תתבצע בסביבת ייצור בלבד (נתון לשיקול מערך הגנה בסייבר), כאשר המערכת במצב מוכן לעלייה לאוויר, ומנהל הפרויקט הכריז על הקפאת התצורה. שלב זה כולל בין היתר את החלקים הבאים:

- הכרזה על סיום פיתוח
- הכרזה על סיום התקנה
- השלמה מוצלחת של בדיקות קבלה
- מסירה של תיק מערכת הכולל פירוט ארכיטקטורה
- מסירה של המערכת לראש צוות אפליקציה במחלקת אבטחת מידע
- מתן נגישות ברמת תקשורת
- מתן הרשאות, כולל כרטיסים חכמים אם יש צורך בכך.
- ניקויי כל הספריות הזמניות מהשרת

4.1 בדיקות חדירות וסקר קוד

כל אתר חדש / גרסה חדשה נדרשים לעבור בדיקות חדירות וסקר קוד מקיפים כשלב מקדים לפני ביצוע בדיקות חוסן. מבדקי חדירות לאתרי/אפליקציות רשת (web application/site)

הנחיות:

א. בהעברת מערכת לבדיקות חדירות יש לצרף מסמך אפיון הכולל לכל הפחות תיאור הפונקציונאליות וממשקים חיצוניים של המערכת – מידע זה חיוני ונדרש לבודקי המערכת לצורך ביצוע הבדיקות.

ב. מבדקי חדירות יבוצעו על כל גרסה של אתר / אפליקציה.

ג. מבדקי חדירות יבוצעו על גרסאות שסיימו את שלבי הפיתוח והבדיקות (QA) – גרסאות סופיות/שחרור.

ד. האתרים/אפליקציות ייבדקו בסביבת ביניים (Stage) ולאחר אישור תקינותם יועברו לסביבת הייצור (Production).

ה. בסיום המבדק יסווגו האתר/אפליקציית הרשת על פי הפירוט הבא:

ו. אין ממצאים – האתר/אפליקציית הרשת יאושרו לעליה לאוויר

ז. יש ממצאים – האתר / אפליקציית הרשת יוחזרו לצוותי הפיתוח והבדיקות לצורך טיפול בממצאים ויצירת גירסה חדשה ומעודכנת. הגרסה החדשה תעבור בדיקות קוד וחוסן חוזרים.

מבדקי חדירות תקופתיים – אתר / אפליקציית רשת שלא עודכנה להם גירסה במהלך 24 חודשים נדרש לבצע מבדק חדירות מלא.

הערה חשובה: לאחר ביצוע בדיקות החוסן וסקרי הקוד – יתקיים תהליך במהלכו יקבל כל ממצא וליקוי (ככל שיימצא במערכת) ציון אשר יסווג את חומרת הממצא/ליקוי. ממצאים וליקויים עם רמת סיווג/חומרה גבוהה או קריטית יקבלו התייחסות הולמת תוך הסתמכות על סטנדרטים מקובלים ומתודולוגיות מוכרות בתעשיית ההיי-טק בכלל ועולם הסייבר בפרט – ובהתאמה לכל תקני האיכות, אבטחת המידע וההגנה בסייבר.

4.2 תחזוקה

במהלך חיים רגיל של מערכת מידע, יש לעדכנה מעת לעת. עדכונים אלו נובעים ככלל הן מדרישות פונקציונליות חדשות והן מהתגלותן של תקלות אבטחת מידע חדשות בתשתית ובקוד כאחד. התחזוקה תתבצע על ידי איש תפעול ממשרדי ממשל זמין בלבד.

למען הסר ספק: **לא תתאפשר גישה מרוחקת לצרכי תפעול** מאתר פיזי שאינו ממשל זמין

4.3 בדיקה תיקון ליקויים (רגרסיה)

בדיקת תיקון לקויים תתבצע על ידי אותו הבודק שביצע את הבדיקה המקורית . הבדיקה תתבצע טרם עליית המערכת לאוויר. באחריות מנהל הפרויקט לשריין זמן מתאים לכך.
בדרך כלל זמן הבדיקה יהיה כ 20 אחוז מזמן בדיקה ראשונית שבוצעה במערכת.

4.4 עדכון גרסה

עדכן גרסה של האפליקציה מחייב בדיקה מחודשת על מנת לוודא כי העדכון לא משנה את רמת אבטחת המידע במערכת. יש לתאם את הבדיקה לפי הנוהל של בדיקת תיקון ליקויים.

4.5 ציוד מיוחד

בממשל זמין מתאכסנות גם מערכות, הדורשות ציוד חומרה או תוכנה מיוחדים לצורך עבודתן התקינה, כגון אפליקציות לטלפון נייד או למערכת הפעלה אחרת. באחריות מנהל הפרויקט למסור חומרה, תוכנה וכל פריט אחר טרם תחילת העבודה. כמו כן, מנהל הפרויקט אחראי לבצע הדרכה על הציוד לצוות הבודק.

4.6 הנחיות נוספות

במחשבי ושרתי ייצור של המערכת לא יותקנו סביבות פיתוח וגרסאות Beta

ערפול קוד – Code Obfuscation נדרש על מנת למנוע Reverse Engineering וקריאת הערות המפתח – זאת כמענה לדרישה מתקן אבטחת הנתונים בתעשיית כרטיסי האשראי – PCI-DSS.

ככלל, יש לבצע ערפול קוד בכל מצב בו יותקנו אפליקציות על פלטפורמה אשר אינה מתארחת פיזית בתשתיות ממשל זמין וכאשר האפליקציה (הקוד המקומפל) נגישה למשתמש הקצה – לדוגמא, אפליקציית מובייל או Sign and Verify.

מעבר לכך, ערפול הקוד מתייתר כאשר מדובר על קוד הנמצא בשרתים אחוריים אליהם אין גישה של המשתמש.

4.7 הסרת מחלקות עזר

בזמן הפיתוח, נעשה לעיתים שימוש במחלקות עזר (Utility classes) העוזרות לפתח את האפליקציה ולבצע בדיקות של המערכת. צעד זה הינו חשוב בעת הפיתוח, אך יש להקפיד בעת המעבר לסביבת הייצור כי קבצים אלו יוסרו בשל העובדה כי הם מהווים פתח נוסף לאפליקציה, פתח אשר אינו נחוץ כלל לצורך ריצה תקינה של התוכנית. בנוסף, במידה ולא הושקעה מחשבה בעקרונות אבטחת מידע והם לא נלקחו בחשבון בעת כתיבת המחלקות, מצב זה עלול להוביל ליצירת "דלת אחורית" (Back Door) במערכת.

4.8 קבצי Debug בייצור

יש לוודא כי הגרסה המקומפלת של האפליקציה בסביבת הייצור הינה גרסת ה-Release ולא גרסת ה-Debug.

גרסת ה-Debug של האפליקציה מכילה מידע אשר מאפשר בעזרת הכלים המתאימים לקבל בחזרה את קוד המקור של האפליקציה (Reverse Engineering).
בנוסף יש לוודא כי גם קבצים מסוג PDB(מכיל מידע הקשור לפעולת ה Debug) הוסרו מסביבת הייצור.

4.9 EOL

תשתיות ממשל זמין בכלל וחוות האירוח בפרט כוללות מוצרי תוכנה וחומרה רבים מיצרנים שונים. לכלל המוצרים מוגדר "מחזור חיים" על ידי היצרנים השונים הכולל שלב של הפסקת תמיכה במוצר הן ברמת תיקוני אבטחת מידע והן ברמת תיקון באגים.

מוצרים שאינם מקבלים עדכוני אבטחה ו/או תיקוני באגים מהיצרן מהווים "חוליה חלשה" במערך אבטחת המידע של ממשל זמין ובכך מחלישים את מארג אבטחת המידע כולו. הסיכון לפגיעה באותן "חוליות חלשות" ודרכן פגיעה ברשתות אליהן הן מקושרות עולה ככל שמועד הוצאת המוצר מסביבת הייצור מתקרב למועד סיום תמיכת היצרן במוצר.

כחלק מתהליך ניהול הסיכונים בממשל זמין הוגדר שמוצרים שהגיעו לסוף "מחזור החיים" יוצאו מסביבות הייצור ויוחלפו במוצרים מגרסאות עדכניות להן ניתנת תמיכת יצרן. יש להיערך לכך בזמן הפיתוח ובהסתכלות להמשך.

5. נספח א- הנחיות ארכיטקטורה

5.1 חוקיות

המערכת המתאכסנת בממשל זמין תהיה כפופה לחוקי מדינת ישראל והחוק הבינלאומי. בין היתר, יש לשים דגש בעמידתה של המערכת בחוק הגנת הפרטיות של מדינת ישראל ותקנותיו השונות. על המערכת לבצע עדכונים לצורך תאימות לחוק תוך זמן סביר.

5.2 סיווג המידע

סיווג המידע בכלל המערכות המתאכסנות בממשל זמין הינו – **בלתי מסווג – בלמ"ס**

5.3 רגישות מידע

תפעולי - מידע תפעולי ישמר על פי best practices הנהוגות בסביבה המדוברת ובהתאם להנחיות אבטחת מידע שניתנו למערכת

פרטי - מידע פרטי רגיש יאוחסן בצורה מוצפנת במאגר נתונים בהתאם להנחיות פיתוח מאובטח.

עסקי - כל מידע אחר, אשר חשיפתו לציבור אינו רצוי, ואשר לא נופל תחת ההגדרות תפעולי או פרטי.

5.4 הזדהות

5.4.1. **משתמשים רגילים** - ככלל גישת משתמשי המערכת תהיה ברמה של קריאה בלבד, בעת הצורך להרשאת כתיבה, הפעולה תעשה בצורה מזוהה על ידי כרטיס חכם או מערכת המאפשרת סיסמא חד פעמית (OTP) ובאישור מערך הגנה בסייבר של ממשל זמין.

5.4.2. **מנהלי המערכת** - מנהלי המערכת יזוהו על בסיס כרטיס חכם בלבד. על המערכת להתממשק למערכות הזדהות כרטיס חכם של ממשל זמין או לממש זאת בצורה עצמאית.

5.5 ממשק ניהול

על המערכת לפרסם את ממשק הניהול בכתובת IP שונה מהכתובת שחשופה למשתמשי הקצה. המערכת לא תאפשר גישה של משתמש הקצה אל ממשק הניהול. כמו כן ממשק הניהול לא ייחשף לעולם בצורה פומבית ללא אישור מערך הגנה בסייבר של ממשל זמין.

5.6 ממשקי עדכון תוכן

ממשק עדכון תוכן יונגש למעדכני תוכן על בסיס כתובת IP אחרת מהכתובת הפומבית למשתמשי המערכת.

5.7 הפרדת שכבות

תכנון נכון של ארכיטקטורת המערכת מהווה את אחד הגורמים היותר חשובים להצלחת הפרויקט כולו. הארכיטקטורה שהוכחה כארכיטקטורה המאובטחת והמומלצת הינה ארכיטקטורה הנקראת ארכיטקטורת שלוש השכבות (3-Tier Architecture). כמובן ניתן לבנות מערכת בארכיטקטורה בעלת מספר שכבות גדול יותר.

על המערכת להפריד בין שכבת הנתונים לשכבת התצוגה (שכבת הפרזנטציה). במידה וגורמי אבטחת מידע דרשו הפרדה של שכבת הלוגיקה העסקית (Business Layer) במערכת יש להפרידה לשרת נפרד גם כן.

- **שכבת התצוגה**: כוללת את הקוד אשר מציג את המידע למשתמש. שכבה זו תטפל רק בקלט מהמשתמש והפלט אשר יוצג לו ולא תכיל קטעי קוד ארוכים אשר אינם מטפלים בממשק המשתמש.
- **שכבת הנתונים:** כוללת את הנתונים עצמם. זו השכבה הרגישה ביותר ומכילה את כל הנתונים הרגישים של הארגון. שכבה זו ממומשת בשכבת מסד הנתונים.
- **שכבת הלוגיקה העסקית**: שכבה זו ממומשת בד"כ על ידי שרתי אפליקציה והיא מנהלת את התהליכים העסקיים והלוגיקה. כמו למשל: מנגנוני אבטחה, חישובים וכו'.

5.8 מנגנון ניהול אירועים

על מערכת המידע להכיל מנגנון ניהול אירועים. מנגנון מסוג זה מועיל במיוחד במקרים של תפקוד לא נכון, שגיאות וניסיונות פריצה. טרם פיתוח המערכת יש לסכם עם צוות

אבטחת מידע את סוג, תבנית וכמות האירועים במערכת. צוות אבטחת מידע יאסוף נתונים אלו במערכותיו לצורך חיווי על רמת אבטחת המידע וניסיונות תקיפה אפשריים.

5.9 העלאת קבצים

העלאת קבצים תבוצע על בסיס תשתיות תוכנה והלבנה הקיימות בממשל זמין. על המערכת להתממשק לתשתיות אלו ולא ליישם מנגנון בצורה עצמאית.

5.10 קוד צד לקוח

ככלל, המערכות לא יפעילו קוד בצד הלקוח במטרה אחרת מאשר עיצוב ותצוגה נוחים יותר.

במידה וקיים צורך עסקי אחר, כגון, הצפנה , חתימה או הפעלת פעילות אחרת הדורשת המצאות של קוד במחשבו של הלקוח, הסיבה תתואר, ותוגש לאישורו של מנהל מערך הגנה בסייבר של ממשל זמין.

היה ואישר מנהל מערך הגנה בסייבר את הצורך , הקוד יהיה חתום דיגיטאלית ולא ידרוש שינוי בהגדרות אבטחת המידע במחשב המשתמש.

חריגה מהנחיה זו תהיה באישור מנהל מערך הגנה בסייבר של ממשל זמין.

5.11 הפניית לקוח לשירות צד שלישי

לצורך סעיף זה, מערכות צד שלישי הינן אותן המערכות שניהול אבטחת מידע שלהן אינו ברשות ממשלת ישראל.

כאשר מתבצעת הפנייה ייזומה של הלקוח לאתר צד שלישי, על המערכת להוציא הודעה ברורה על כך שתוכן האתר אינו באחריות ממשלת ישראל.

כאשר האתר המאוחסן בממשל זמין מבצע embedding של רכיב שמקורו באתר צד שלישי, חלק זה בעמוד ייצבע בצבע אדום בולט עם תוספת של טקסט שמסביר זאת.

5.12 שימוש בספריות צד שלישי

בעת שימוש בספריות צד שלישי בין אם יושבות על השרת ובין אם נלקחות כקישור מאתר אחר יש לוודא שספריות אלו עדכניות לגרסתן האחרונה, בנוסף מעת לעת תעודכן רשימת הספריות הניתנות לשימוש וגרסאות עדכניות באתר **https://e.wcf.egov.gov.il/** תחת לשונית Documentation אשר נגיש לכלל המשרדים באמצעות כרטיס חכם.

5.13 שירותי אינטרנט

שירותי אינטרנט (web services) יעבדו דרך ממשק הDATA POWER ויפותחו על פי תקן ה wsdl.gov.il .

6. נספח ב' - הנחיות פיתוח מאובטח

6.1 מדיניות שמות משתמש

6.1.1. אורכם של שמות המשתמשים של המערכת יהיה לפחות 8 תווים.

6.1.2. שימוש במספרים ואותיות אנגליות בלבד.

6.1.3. שם משתמש לא יסגיר פרטים אישיים.

6.1.4. לא יוגדרו במערכת משתמשים בעלי שם משתמש טריוויאלי, כגון 'admin'.

6.1.5. שם משתמש יהיה חד-חד ערכי לאורך כל חיי המערכת (אם משתמש מבוטל/נמחק אין להשתמש בשם המשתמש שלו שוב)

6.2 מדיניות סיסמאות

6.2.1. הסיסמא לא תעבור גלויה ברשת אלא על גבי תווך מוצפן.

6.2.2. המערכת תספק למשתמש את היכולת להחליף את הסיסמא בעצמו, בצורה בטוחה, בכל עת.

6.2.3. המערכת תכיל בדיקה שתוודא ששם המשתמש והסיסמא יהיו שונים זה מזה ולא יכילו אחד את השני (שם המשתמש לא יהיה חלק מהסיסמא ולהיפך).

6.2.4. המערכת לא תאפשר הקלדת תווים רצופים במקלדת או סדר עולה או יורד של אותיות וספרות

6.2.5. סיסמת המשתמש לא תהייה קצרה מ-8 תווים ותהייה מורכבת מ-3 מתוך 4 קבוצות התווים:

- אותיות קטנות;
- אותיות גדולות;
- ספרות;
- תווים מיוחדים;

6.2.6. סיסמתו של מנהל המערכת (Administrator) תהייה באורך של 12 תווים לפחות.

6.2.7. תוקפה של סיסמת משתמש ניהול המערכת או בעל הרשאות גבוהות במערכת יפוג כל 120 יום ועל המשתמש יהיה להחליף את סיסמתו בהתאם למבנה המתואר לעיל.

6.2.8. מנגנון החלפת הסיסמא ישמור את היסטוריית הסיסמאות של 3 מחזורים לפחות ולא יאפשר למשתמש לחזור על אף אחת מהסיסמאות הללו (או דומה להן) בעת החלפת הסיסמא.

6.2.9. טרם החלפת הסיסמא על המשתמש יהיה להקיש את סיסמתו הנוכחית.

6.2.10. החלפת הסיסמא לא תתאפשר בטווח של 24 שעות מהחלפת הסיסמא האחרונה.

6.2.11. הסיסמא הראשונית של המשתמש תהייה רנדומאלית ובהתאם למבנה שהוגדר לעיל.

6.2.12. המערכת תחייב את המשתמש להחליף את סיסמתו הראשונית בעת ההתחברות הראשונה למערכת.

6.2.13. תוקף הסיסמא הראשונית יהיה 3 ימים, ולאחר מכן המשתמש ינעל ולא יוכל להשתמש בה.

6.2.14. במקרה בו המשתמש שכח את סיסמתו, המערכת תיצור לו סיסמא חדשה. כמו הסיסמא הראשונית, סיסמא זו תהייה מוגבלת בתוקף והמשתמש יהיה מחויב להחליפה בעת השימוש הראשון בה.

6.2.15. יצירת סיסמא חדשה במקרה בו המשתמש שכח את הנוכחית תהייה אך ורק לאחר זיהוי המשתמש באמצעים אחרים, כגון כתובת דואר אלקטרוני, שאלות סודיות וכדומה. מומלץ להשתמש בשילוב של השיטות כגון: שליחת מייל למשתמש עם קישור חד פעמי ומוגבל בזמן המשמש לאיפוס סיסמא, ואחרי שהמשתמש גולש הוא נשאל שאלות בטחון ואם הוא מצליח אז ניתנת לו האפשרות לשנות את סיסמתו.

6.2.16. אין להציג בשום שלב במחשבי המערכת, במחשבים של משתמשי המערכת, בקוד המקור של דפים וטפסים המועברים למשתמש, את מזהי האימות של המשתמשים השונים במערכת.

6.2.17. סיסמת המשתמש תשמר בצורת hash & salted בבסיס המידע ואין לשמור אותה כ-Clear Text.

6.2.18. יש להשתמש באלגוריתמי Hash בטוחים כגון SHA-256 או SHA-512. יש להימנע משימוש באלגוריתמים לא בטוחים כגון SHA-1 ו-MD5.

6.2.19. **הערה חשובה:** לפרטים נוספים יש לעיין בנספח א' למסמך מדיניות פיתוח מערכת מאובטחת ***מסמך הנחיות מפורטות לפיתוח מערכת המנהלת משתמשים.***

6.3 נעילת משתמשים

6.3.1. נעילת המשתמשים תתבצע לאחר 3 ניסיונות הזדהות כושלים.

6.3.2. במקרה של מיעוט משתמשים אין להשתמש במנגנוני שחרור אוטומטי, אלא השחרור יבוצע על ידי מנהל המערכת לאחר קבלת הפניה מהמשתמש ווידוי זהותו.

6.3.3. במקרה של מערכת פתוחה לציבור ו/או בעלת משתמשים רבים יש ליישם מנגנון שחרור אוטומטי לאחר פרק זמן קבוע (למשל 30-15 דקות).

6.3.4. נעילת המשתמש תתבצע בצד שרת המערכת ולא ברמת ה-Session או ה-Client.

6.3.5. משתמש ניהול המערכת לא ינעל לאחר ניסיונות זיהוי כושלים על מנת למנוע מצב של מניעת שירות של המערכת.

6.3.6. במקרה של נעילת המשתמש אין להודיע על כך באופן מפורש בכדי למנוע מצב של מיפוי שמות משתמשים בעזרת נעילה מכוונת. ניתן לדווח על הנעילה ישירות לדואר האלקטרוני של המשתמש או להטמיע אותה בהודעת הכישלון בהתחברות הרגילה ("שם משתמש וסיסמא לא נכונים, יתכן והמשתמש ננעל אם ניסית מספר רב של פעמים")

6.3.7. יש לאפשר למנהל המערכת לבצע, במידת הצורך, ביטול/הקפאת/נעילת חשבונות משתמשים. מנגנון כזה יסייע בצמצום השפעותיה של תקיפה מוצלחת של המערכת במקרה ומנגנוני ההגנה נפרצו.

6.4 ניהול משתמשים והרשאות

6.4.1. הרשאותיהם של המשתמשים יקבעו לפי עקרון ההרשאות המינימאליות הדרושות, כלומר כל משתמש מערכת יקבל את הרשאותיו בהתאם לדרישות עבודתו במערכת ולא מעבר לכך.

6.4.2. הרשאות המשתמש ייבדקו בכל השכבות ובכל הרכיבים של המערכת.

6.4.3. יש לבצע בדיקת הרשאות משתמש בכניסה לכל דף במערכת.

6.4.4. יש לבצע בדיקת הרשאות משתמש טרם ביצוע פעולות במערכת לרבות פעולות צפייה במידע, מחיקה, עדכון או הוספה.

6.4.5. אין להסתמך על מנגנון זיהוי כמנגנון הרשאות. משתמש מזוהה במערכת אינו בהכרח מורשה לכל חלקיה.

6.4.6. בקרת הגישה תתבצע בצד השרת בלבד ולא תסתמך על נתונים השמורים במחשבו של הלקוח, לדוגמא cookies

6.4.7. יש לבצע הפרדת תפקידים ולוודא כי לכל בעלי תפקיד ניתנו ההרשאות הנדרשות להן ולא מעבר לכך. לא כל המשתמשים נדרשים לבצע את כל התהליכים לכן יש לחלק את כל המשתמשים לקבוצות ולהתאים לכל קבוצה את ההרשאות הנדרשות לה ולא מעבר לכך. לאחר יישום המנגנון יש לוודא שמשתמשים בעלי רמת הרשאות נמוכה אינם יכולים לבצע פעולות הדורשות רמת הרשאות גבוהה יותר.

6.4.8. מנגנון הרשאות דקלרטיבי מאפשר גישה למשאבים רק עבור משתמשים אשר נמצאים ב-Role מסוים או בקבוצה (Group) שנמצאת ב-Role מסוים. כך ניהול המשתמשים נעשה במקום אחד מרכזי (כל סוג של LDAP, קבצי תצורה וכו') והינו מתאים להרשאות בין מערכות בעיקר.

6.4.9. הרשאות ברמת הקוד מאפשרות גמישות רבה בזמן כתיבת הקוד אך גורמת לאבדן יתרון הגמישות בזמן העבודה עם המערכת משום שלצורך תחזוקת מנגנון זה יהיה צורך לשנות את קוד האפליקציה. מומלץ לעשות שימוש במנגנון זה רק כאשר מנגנון ההרשאות הדקלרטיבי אינו גמיש דיו במענה לצרכי האפליקציה, על מנת ליישם מנגנון הרשאות משלים או על מנת למנוע שינויים זדוניים בקבצי התצורה.

6.4.10. ניהול ההרשאות ניתן לביצוע בשתי רמות עיקריות: הרשאות לפי פעולות או הרשאות לפי משאבים.על כל פניה צריכה להבדק תחילה רמת ההרשאות (האם המשתמש רשאי לבצע את הפעולה המבוקשת או האם הוא רשאי לגשת למשאבים מסוימים). לדוגמא, האם מנהל רשאי לראות פרטים אישיים של לקוחות, או האם מנהל רשאי לראות את פרטי כל הלקוחות או רק את פרטי הלקוחות השייכים למחלקה שלו.

6.4.11. הרשאות ריצת המערכת: טעות נפוצה היא להריץ את התהליך עם משתמש בעל הרשאות נרחבות. במצב כזה אם ישתלט פורץ על האפליקציה תהיה לו שליטה על כל השרת וגם על אפליקציות אחרות הקיימות באותו שרת. לכן יש להגדיר משתמש מיוחד עבור התהליך ולתת למשתמש זה את מינימום ההרשאות הנדרשות. יש ליישם עקרון דומה גם על משתמשי בסיס נתונים. על המערכת לעבוד עם משתמש ייעודי בעל הרשאות מינימאליות הנדרשות לפעולות מסוימות ולטבלאות מסוימות.

6.5 אימות קלט

6.5.1. בקרת קלט מהמשתמש תיבדק בשכבות השונות בהתאם לסוג המידע שאמור להתקבל שימוש ב white list – regular expression, קרי, סינון על פי ערכים

מותרים ידועים מראש ולא שלילת ערכים, וזאת משום שניתן להציג קלטים ביותר מצורה אחת על ידי שימוש בקידוד שונה.

6.5.2. הבדיקות יתבצעו גם בצד המשתמש וגם בצד הלקוח.

6.5.3. מומלץ לבצע אימות קלט בכל אחת משכבות האפליקציה – למשל קוד, מסד נתונים, שכבות Web Services וכו'.

6.5.4. בגישה ל WS וגישות SOAP באמצעות XML, יש לבצע בדיקות לקלט שמועבר למערכת לפי סכמות XSD מוגדרות מראש לכל פעולה \ מתודה בשרות אליו מתבצעת הגישה, על פי נוהל פיתוח סכמות מאובטחות WS-Gov.il.

6.5.5. יש לבצע את בדיקות התקינות של הקלטים לפי קיום, סוג, אורך, טווח ומבנה. יש לקבל רק תווים מסוג שרלוונטי לאותו שדה (שמות, מספרים, תאריכים, סכומים וכו'). יש לוודא שהמספר המתקבל בקלט הינו בגבול/בטווח המותר, בתבנית המותרת (לדוגמא תאריך), באורך המותר (לא ארוך יותר ולא קצר יותר) ובכל מקרה לוודא שאינו כולל פקודות SQL או HTML. יש לשים דגש מיוחד לתו ה-NULL מאחר וכאשר התוקף מחדיר NULL בתוך פרמטרים בטופס, הדבר עלול לסיים מחרוזות מוקדם מהמצופה או לעקוף מסננים מסוימים. אין להסתמך על אורך הקלט בלבד (Size=x) לצורך אימות הקלט.

6.5.6. יש לתעד נסיונות מובהקים של הכנסת תווים בעייתיים המשמשים בהתקפות SQL Injection או Cross Site Scripting. כמו כן יש לתעד נסיונות לשתול בקלט תווים מסוימים הנקראים Meta Characters אשר הינם בעלי משמעות מיוחדת במערכות הפעלה נפוצות ויכולים להתקבל כפקודות לביצוע פעולות מסוימות במערכות אלו.

6.6 הגנה על מידע רגיש

6.6.1. יש להצפין נתונים רגישים במערכת. כגון:

- נתונים רגישים וחסויים של משתמשי המערכת.
- במידה וקיימים קבצים (כגון קבצי Word, PDF, תמונות וכדומה) אשר מכילים מידע רגיש בבסיס הנתונים יש להצפינם גם כן.
- נתוני זיהוי של רכיבי תוכנה שונים, כגון נתוני הזיהוי של שרת האפליקציה לשרת בסיס הנתונים (Connection String) וכדומה.

- מפתח ההצפנה יישמר במקום מאובטח על שרת המערכת, כגון ה-Registry. הגישה למפתח תוגבל לאפליקציה ולאדמיניסטרטור של השרת בלבד.
- יש לבצע הצפנה של המפתח ע"י שימוש בהצפנת DPAPI, יש לשמור עותק של המפתח במקום מוגן נפרד (פיזי) למקרה שלא ניתן לשחזר את המפתח המקורי.
- חל איסור על שמירת מפתחות פומביים באתרי אחסון משותפים ושטחי אחסון אחרים אשר נמצאים מחוץ לממשל זמין וזאת מחשש שהמפתחות יהיו נגישים גם בפני גורמים בלתי מורשים.

6.6.2. אחסון סיסמאות באופן מאובטח תעשה באופן הבא:

- הסיסמאות אינן דורשות הצפנה דו כיוונית כיוון שאין צורך באחזורן, לפיכך הסיסמאות ישמרו בבסיס הנתונים לאחר ביצוע HASH על ערכן.
- לכל משתמש בעת יצירת סיסמא ייבחר ערך רנדומאלי אשר ישורשר לסיסמא טרם ביצוע ה-HASH. ערך זה נקרא ערך SALT.
- ערך ה-SALT ישמר בבסיס הנתונים יחד עם פרטי המשתמש.
- על מנת לבדוק כי הסיסמא שהמשתמש הזין הינה נכונה, משרשרים אליה את ערך ה-SALT מבסיס הנתונים ומבצעים על הערך החדש את פעולת ה-HASH שבוצעה בעת שמירת הסיסמא. אם ערך ה-HASH החדש תואם את ערך ה-HASH אשר שמור בבסיס הנתונים, הרי שהסיסמא נכונה.

6.6.3. יש להשתמש באלגוריתמי Hash בטוחים כגון SHA-256 או SHA-512. יש להימנע משימוש באלגוריתמים לא בטוחים כגון SHA-1 ו-MD5. יש להתעדכן לפי העדכונים האחרונים בהקשר זה.

6.6.4. יש להשתמש במנגנוני הצפנה מוכרים ובדוקים ובהצפנות מקובלות כיום בשוק, כגון RSA, ולא לבנות אלגוריתם הצפנה ייחודי למערכת. במנגנון הצפנה חדש אשר לא מומש כראוי עלולה להיווצר פרצת אבטחה .

6.6.5. אין לאפשר שמירת נתונים רגישים של המערכת במחשבו של המשתמש.

6.6.6. יש למנוע את שמירת נתוני המערכת בספריית הקבצים הזמניים ובמנגנוני ה-Cache במחשב המשתמש.

6.6.7. יש להצפין את ה- Web.Config של האתר .

6.7 הגנה על מידע בתעבורה

6.7.1. עקב רגישות המידע והעברתו בתווך אינטרנט, יש להעביר את כלל המידע הרגיש באופן מוצפן, לרבות כל המידע והמסכים המוצגים לאחר הזדהות כלשהי בפני מערכת ממשל זמין.

6.7.2. כל התעבורה תתבצע על גבי תווך מוצפן (HTTPS) – פרטים נוספים ניתן למצוא במסמך נפרד – נספח למדיניות אבטחת המידע - הכולל הנחיות ליישום מדיניות של HTTPS-Only.

6.7.3. נא להשתמש במתודות של POST בלבד בכדי שהנתונים לא יחשפו לגורם אשר רואה את הפנייה.

6.7.4. אין להשתמש במתודות של GET להעברת פרמטרים רגישים.

6.7.5. מומלץ להוסיף הגבלות אלו ברמת האפליקציה.

6.7.6. אין חובה להוסיף הגנה ברמת התעבורה על אתרים פומביים שאינם מכילים מידע רגיש או הזדהות.

6.8 ניהול מופעי משתמשים (Session Management)

6.8.1. יש להבטיח כי נתוני Session נשמרים בצורה בטוחה במהלך חיי המערכת ובפעולות המערכת השונות המתבצעות עם האובייקטים \ משתמשים.

6.8.2. יש להבטיח כי קיימת הפרדה בין ניהול הזהויות לבין שימוש ב Session כך שלא יתכן מצב כי משתמש שלא ביצע הזדהות יוכל להשתמש ב Session פעיל של משתמש שביצע הזדהות כנדרש (גניבת זהות), כלומר יש להבטיח כי המערכת אינה מסתמכת על נתוני Session בכדי לאפשר למשתמש חשיפה למידע ופעולות רגישים במערכת.

6.8.3. בעבודה עם Session, נא להגדיר אותו כ Secure ו HTTPOnly.

6.8.4. יש להשתמש ברכיבי Session רק עבור שמירת מצב משתמש בין בקשות http שונות במערכת וכן לצורך ביצוע personalization עבור משתמש.

6.8.5. אין לשמור מידע רגיש ב Session , במידה ונדרש יש לבצע הצפנה של מידע זה.

6.8.6. בכל מצב שבו נשמר מידע רגיש ב Session יש להבטיח כי המידע נשמר בצורה בטוחה ולא תתאפשר גישה אליו שלא דרך מקור מוסמך ומאושר (כלומר מהאפליקציה שייצרה את המידע) .

6.8.7. על המערכת להימנע במידת האפשר בשימוש ב client – side state management כגון view state, hidden files,cookies לצורך קבלת נתונים עבור Session .

6.8.8. האפליקציה תעשה שימוש רק בזהות אשר נתקבלה בתהליך ההזדהות בכניסה לאפליקציה ואשר מבצעת שימוש ב- Session ID ייחודי וזמני.

6.8.9. יש למנוע ביצוע גישה למערכת ללא Session תקין.

6.8.10. יש למנוע ביצוע גישות מרובות מאותו Session למערכת.

6.8.11. אין להעביר את נתוני הזיהוי של המשתמשים בין מחשב המשתמש לשרתי המערכת, למעט דף הכניסה למערכת.

6.8.12. יש לקיים מנגנון Idle Timeout אשר יסיים את ה-Session של המשתמש לאחר מספר דקות מוגדר, כ-15 דקות, של חוסר פעילות במערכת.

6.8.13. יש לקיים מנגנון Session Timeout אשר יסיים את ה-Session לאחר זמן ארוך של פעילות במערכת, כ-8 שעות. מנגנון זה נועד למנוע שימוש במערכת באמצעות סקריפטים וכדומה.

6.8.14. יש לשקול ניתוק Session במצבי שגיאה מסוימים.

6.8.15. ניתוק ה-Session יבוצע על ידי סיום תוקף ה-Session בצד השרת, ולא על ידי העברת הלקוח לדף הכניסה בלבד.

6.9 ניתוק שיחה

6.9.1. האפליקציה תאפשר יציאה מסודרת ונוחה מהמערכת בכל דף החל מדף הכניסה (Login).

6.9.2. ניתוק זה יבטיח כי משתמש לא יוכל לבצע שימוש חוזר במערכת ללא ביצוע הזדהות מלאה מחדש וזאת על ידי סגירת ה-Session שלו וכלל המשאבים שהוקצו לו בטרם הניתוק.

6.9.3. במקרה של זיהוי פעילות חשודה במערכת (כפי שהוגדרה במידול הסיכונים) כגון ניסיונות לביצוע SQL Injection או הזנת סקריפטים זדוניים בשדות קלט, נדרש לבצע ניתוק כפוי של המשתמש, לבצע רישום ללוג וכן להתריע על כך למנהל המערכת.

6.9.4. במקרה של סגירה לא מבוקרת של האפליקציה – סגירת הדפדפן, כיבוי המחשב עצמו וכד' – יש להגדיר זמן Timeout ואם השרת לא קיבל בקשות בפרק הזמן

המוגדר אזי יימחק אובייקט ה-Session ותידרש הזדהות חוזרת על מנת להכנס למערכת.

6.10 שימוש בתעודות והצפנות

6.10.1. עבור מידע המוגדר כרגיש, יש לאפשר טיפול באמצעי מידור הן ברמת מנהלי המערכת והן ברמת המשתמש. כולל:

- תמיכה בסוגי מידע שונים.
- יכולת הגדרה במערכי ה-Audit לרישום גישה או ניסיונות גישה למידע המוגדר כרגיש. רישום ה-Audit יבוצע באופן מלא בכל שכבה, ובביצוע האחזור ניתן יהיה להפריד באופן מובהק בין התהליכים ובין השכבות השונות שבהן בוצע ה-Audit.

6.10.2. כאשר מתבצעת הצפנה למידע רגיש יש לממש אלגוריתמי הצפנה לפי הכללים הבאים:

- אין לבצע שימוש באלגוריתמים שפותחו בצורה עצמאית.
- יש לבצע שימוש באלגוריתמים מוכרים כגון:

  AES עבור הצפנה סימטרית

  RSA עבור הצפנה א-סימטרית

  Sha-2, Sha-256 או Sha-512 עבור hash חד כיווני
- עבור יצירת מספרים רנדומאליים יש להשתמש במנגנונים מומלצים על ידי יצרני המוצר או הסביבה.

6.10.3. הערה: במקרים של ספק ניתן לפנות למערך הגנה בסייבר לקבלת ייעוץ

6.11 הגנה על מידע תפעולי רגיש

6.11.1. **כללי**

- על המערכת להימנע משמירת מידע רגיש בקבצי הגדרות, קבצים זמניים, cookies, זיכרון מטמון וכו'. במידה ומידע נשמר במקומות אלו, נדרש לוודא כי לאחר סיום עבודה במערכת מידע זה ימחק.

6.11.2. **מפתחות הצפנה**

- יש לאבטח את מפתח \ מפתחות ההצפנה הנמצאים בשימוש המערכת מפני גישה \ שימוש זדוני ללא הרשאה בהתאם לסוג המפתח – ציבורי \ פרטי.
- יש להגן על המפתח מפני הרס או שינוי בצורה לא מורשית.
- יש לנהל בקרה ודיווח לגבי ביצוע גישות ושימוש במפתחות הצפנה.
- יש להבטיח יכולות שיחזור וגיבוי בשימוש במפתחות הצפנה (כדי להבטיח שיהיה ניתן לשחזר מידע רגיש שהוצפן עם מפתח שאבד).

6.11.3. **התחברות לבסיס הנתונים**

- יש להשתמש בזיהוי מבוסס מערכת הפעלה (Windows Authentication) ולהעדיפו על פני זיהוי מסד נתונים במידה והדבר נתמך (למשל ב- SQL Server)
- יש לשמור את פרטי ההתחברות לבסיסי נתונים בקבצי ההגדרות בצורה מוצפנת
- יש להגביל גישה לקבצי ההגדרות לפי הקבוצה המינימאלית האפשרית

6.11.4. **ניהול שגיאות**

- הודעות שגיאה שיוצגו למשתמש כתוצאה משגיאות המתרחשות באפליקציה יהיו הודעות שאין בהן כדי לחשוף את אמצעי האבטחה במערכת. יש לוודא כי הודעות שגיאה אינן חושפות מידע רגיש בנוגע למבנה המערכת ומשאבי המערכת. הודעות השגיאה שיוצגו יהיו ג'נריות וכלליות.
- הודעות שגיאה שיוצגו למשתמש יהיו הודעות שאין בהן כדי לחשוף את התשתית האפליקטיבית לגרסאותיה השונות כגון: מערכות הפעלה, שרתי web, שרתי אפליקציה, בסיסי נתונים, פרוטוקולים בשימוש, Web Services בשכבות נמוכות וכדומה.
- אין להציג כל מידע רגיש (כולל : מספרי אשראי, סיסמאות, מפתחות הצפנה וכו') בהודעות שגיאה המוצגות למשתמש.
- כאשר קלט המשתמש אינו מתאים לתבנית הנדרשת בשדה קלט, יש להציג למשתמש הודעת שגיאה המפרטת מהי התבנית בה נדרש להשתמש.
- על המערכת לנהל מערך ללכידת שגיאות בזמן ריצה :
  - יש לצפות שגיאות מראש וללכוד אותן בקוד המערכת.

- בשגיאות שהוגדרו כשגיאות כתוצאה מפעילות הקשורה באבטחת מידע יש לנהוג לפי מה שהוגדר במידול הסיכונים של המערכת, כולל דיווח למנהל המערכת, חסימת משתמש וכו'.
  - יש לדאוג לכך שמידע משגיאות יהיה מתועד ע"י המערכת בדפי ה log שלה.
  - ניתן להציג למשתמש קוד המתאים לרשומה בלוג לשם טיפול בשגיאה. קוד הרשומה לא ירמוז בשום צורה על קוד השגיאה והסיבה להתרחשותה.
  - על המערכת להתמודד עם שגיאות בהיבט של זמינות כך שאם למשתמש מסוים מתרחשת שגיאה הוא אינו חוסם גישה למשתמשים אחרים שמריצים את המערכת (קריסה כללית).
  - במערכות רגישות ובסיכון בינוני ומעלה, המערכת תכלול יכולת לאחזור הודעות שגיאה (אחזור מלא או אחזור חלקי לפי פרמטרים שונים).
  - פורמט הדיווח של הלוגים צריך להתאים לפורמט מערכת SIM \ SOC כך שיהיה ניתן לאסוף את הודעות השגיאה.

6.11.5. **חיווי ובקרה**

- האפליקציה תתעד את הנתונים הבאים, במידה והם מוגדרים, עבור כל פעולה במערכת:
  - Timestamp.
  - זיהוי המשתמש (ללא סיסמא)
  - מיקום המשתמש (מחשב/IP).
  - מיקום המשתמש במערכת (מסך, טופס, טבלה וכדומה).
  - פרטים מלאים של הפעולה המבוקשת.
- בנוסף יתועדו הפעולות הבאות:
  - צפייה במידע במערכת.
  - עדכון מידע במערכת.
  - כתיבה ומחיקה של מידע במערכת.
  - כל פעולות הניהול במערכת.
  - כל פעולות הזיהוי במערכת, כולל כישלונות של פעולות אלו והסיבה לכך.
  - כל פעולות ההרשאות במערכת, כולל כישלונות של פעולות אלו.
  - שגיאות מערכת.
  - ועוד, בהתאם לצורך.

- התיעוד יתבצע בשתי שכבות: תיעוד פעולות משתמשי מערכת באפליקציה, תיעוד גישה לנתוני המערכת בבסיס הנתונים.
- חשוב להדגיש כי התיעוד לא יכיל את נתוני הזיהוי של משתמשים או נתונים רגישים אשר שמורים בבסיס הנתונים של המערכת, בדגש על נתונים רגישים אשר עלול לחשוף חלקים מהמערכת לגורמים עוינים.
- כל פעולת תיעוד, בכל הרמות של המערכת, חייבת להכיל את המשתמש המבצע את הפעולה בפועל על מנת למנוע התכחשות משתמשים לפעולותיהם.

<u>לגבי המעקב</u>

- יש לוודא כי נתוני התיעוד והמעקב נשמרים באופן מאובטח במערכת. ניתן לכתוב את נתוני התיעוד למערכת חיצונית באמצעות API ולאפשר שליחה של הודעות לרישום בלבד (להסיר הרשאות לעדכון ומחיקת הודעות). יש להתיר למשתמשים מסוימים בלבד את הגישה למערכת החיצונית לצורך כתיבת רשומות (ללא יכולת עדכון או מחיקת רשומות).
- יש לוודא כי רישומי התיעוד אינם נגישים למשתמשים ללא הרשאות מנהל מערכת.
- יש לוודא כי נתוני התיעוד מגובים יחד עם שאר נתוני המערכת.
- יש לקיים מנגנון ארכיב לנתוני תיעוד ישנים אשר אינם נחוצים לשם פעולתה התקינה של המערכת.
- הקווים המנחים להגנה על קבצי החיווי (קבצי לוג) הינם כדלהלן:
  - יש להגן על קבצי החיווי מפני גישה בלתי מורשית, שינוי ומחיקה.
  - רמת ההגנה על קבצי החיווי נדרשת להיות זהה לרמת ההגנה של המערכת ממנה נוצרים קבצי חיווי אלו.
  - מעבר לנושא הרשאות הגישה לקבצי החיווי, גיבוי קבצי החיווי וניהול קבצי החיווי – יש למקם את קבצי החיווי במחיצות וספריות בהן קיים סיכון נמוך לפגיעה במרכיבי אבטחת המידע שלהם.
  - במערכת כגון אתר אינטרנט או שירות רשת אחר: אין למקם קבצי חיווי בתיקיות פרויקט ושרתים קדמיים (כגון שרתי IIS) החשופים לעולם.

6.11.6. **חתימת קבצים וקוד**

- יש לבצע שימוש ב strong name ולחתום את קוד הפרויקט לאחר יצירת גרסת ייצור יציבה.
- יש להחליף חתימה זאת בכל שחרור של גרסה חדשה.

6.11.7. CAS

- מומלץ לממש מנגנון CAS במערכת על מנת להגביל את גישת האפליקציה למקורות מידע שלא נדרש לבצע אליהם גישה כגון FTP, Unmanaged code וכו'.

6.11.8. ניהול הגדרות

- על המערכת לפרט באפיון את כל אמצעי גישות ניהול ההגדרות שבמערכת:
- יש להגדיר תחת איזה חשבון רצה המערכת (משתמש , מנהל , חשבון מערכת ...) , הדרישה היא כי המערכת תרוץ תחת חשבון עם רמת הרשאות הנמוכה ביותר הניתנת כך שלא תאפשר ביצוע פעולות לא רצויות ע"י משתמשים רגילים. לא יופעל שום רכיב עם זיהוי SYSTEM ו/או הרשאות ADMIN או מקבילות להם.
- במידה והמערכת דורשת הרשאות גבוהות רק בחלק קטן מהמערכת, יש להריץ את המערכת תחת משתמש נמוך הרשאות ולהתחזות למשתמש בעל הרשאות גבוהות יותר רק בקטע הקוד הרלוונטי.
- יש להגדיר דרכי גישה מאובטחות למשאבים חיצוניים כגון בסיס נתונים, מערכת קבצים וכו' (למשל ע"י הצפנת מחרוזת קישור לבסיס הנתונים).
- יש להגדיר גישה מאובטחת לאדמיניסטרציה במערכת כולל זיהוי חזק והגבלת הגישה למורשים בלבד. יש לשקול הגדרת כתובות IP מסוימות שרק מהן ניתן לגשת לממשק הניהול.
- יש לדאוג לכך שלא יהיה ניתן לחשוף\לגשת למידע רגיש הקיים בקבצי הגדרות למשתמשים ללא הרשאות מתאימות. לשם כך יש להצפין הגדרות רגישות (ברמת האפליקציה ו/או ברמת מערכת הקבצים) ולהגביל אליהם גישה ברמת מערכת ההפעלה.
- אין לשמור מידע רגיש בקבצי הגדרות. במידה ונדרש יש להצפין אותו ע"י שימוש במנגנוני הגנה המומלצים על ידי יצרן מערכת ההפעלה.

6.11.9. גישה לקבצים

- אין להסתמך על גישה לקבצים באופן יחסי (לדוגמא: \data\datafile.dat\...\...). יש להשתמש במיקומים מוחלטים.

- אין להסתמך על ברירת המחדל של מערכת ההפעלה לצרכי פתיחת הקובץ. יש לפתוח הקובץ תוך ציון מצב הפתיחה (קריאה בלבד, או כתיבה).
- בדיקת קיום קובץ לפני הגישה אליו. לפני פתיחת קובץ על האפליקציה לבדוק את קיומו. גישה לקבצים לא קיימים או קבצים אשר אינם ניתנים לקריאה יכולה להעיד על התקפה.

6.12 הגנה מפני מתקפות אפליקטיביות

6.12.1. מניעת התקפות cross site scripting

- יש לבצע בדיקות תקינות בצד השרת על כל הקלט המגיע מצד המשתמש. בדיקת הקלט תכלול את הבדיקות הבאות:
  - יש לבדוק את קיומו של הקלט ולא לאפשר הזנת ערכים ריקים.
  - יש לבדוק ולהגביל את אורך הקלט (עפ"י האפיון שימוש ברשימות white list וב regular expression לפני הכנסת קלט למערכת).
  - יש לבדוק שטיפוס הקלט המתקבל הוא מהסוג המצופה.
  - יש לבדוק כי טווח הערכים שמתקבל מתאים להגבלות שנקבעו.
  - יש לבדוק את הרכב התווים בקלט, ולוודא שהוא אינו מכיל תווים אסורים. ככלל יש להימנע ככל האפשר מקבלת קלט שאינו מכיל ערכים אלפא נומריים, למעט רווחים.
  - יש לוודא כי ערכו של הקלט תואם ללוגיקה העסקית של רכיב היעד.
  - יש לוודא כי הקלט ב encoding המתאים למערכת.
  - יש להעביר את כלל התווים שאינם אלפאנומריים קידוד HTML בטרם הצגתם למשתמש. תהליך הקידוד יבטיח כי קוד שתול יוצג כטקסט ולא ירוץ על הדפדפן. מומלץ להשתמש בתשתית אפליקטיבית מוכרת (כגון AntiXSS ב-.NET).
- אין להכניס לבסיס הנתונים תווים הנובעים מקלט ישירות לתחום הפעולה של client side scripting (תגי script, אירועי HTML וכדומה).

6.12.2. מניעת הזרקות SQL

- אין לאפשר גישה ישירה לבסיס הנתונים. גישה לבסיס הנתונים תתבצע באמצעות שיכבה מתווכת כגון WS או DAL בפרויקט נפרד \ תשתית .

- בכל מקרה יש לבצע סינון מסודר של תווים למניעת הזרקת שאילתות SQL.
- כל תעבורת השאילתות תבוצע ע"י שימוש ב- stored procedures כאשר השאילתות אינן מיושמות כ- Dynamic Queries. שימוש ב- Stored Procedures אינו מאפשר לתוקף לשנות את מבנה השאילתא אלא רק איזורים מוגדרים מראש לפי קביעת כותב האפליקציה ולכן זו הדרך העדיפה לבצע גישה והרצה של שאילתות מול בסיס הנתונים.
- בגישה לבסיס הנתונים (בשכבות ה- DAL או בכל מקום אחר) יש להשתמש בתשתית האפליקטיבית המתאימה והמומלצת לשם מניעת התקפות SQL Injection:
  - ב-.NET: Parameterized Queries, LinQ (Linq to SQL, Entity Framework).
  - ב-Java: Prepared Statements.
- **בכל מקרה יש להימנע משימוש בשרשור מחרוזות בעת הרכבת שאילתות או פקודות מסד הנתונים (לרבות פקודות עדכון נתונים).**

6.12.3. מניעת מתקפות חסימת שירות וזליגת מידע

- בכדי למנוע זליגת מידע של מסדי נתונים פומביים באמצעות תוכנות רובוט (Bot), וכמו כן בכדי למנוע התקפות מניעת שירות על רכיבים שונים באפליקציה, יש להטמיע רכיב CAPTCHA בכל דף חיפוש של מאגר מידע ו\או כל רכיב אחר באפליקציה המאחזר נתונים למשתמש ממאגרי מידע. רכיב ה- CAPTCHA משמש כמנגנון הבודק כי מדובר במשתמש אנושי על ידי בקשה להקלדת תוכן של תמונה המוצגת למשתמש או באמצעות הקלטה קולית המשמיעה את תוכן התמונה. כמו כן דרישה זו תקפה גם בגישה למאגרי מידע שאינם פומביים אך חשופים למספר רב של משתמשים. על רכיב ה-CAPTCHA לעמוד בתנאים הבאים:
  - תוכן המוצג בתמונה צריך להיות באורך מינימלי של 5 תווים\סימנים.
  - יש לשלב בין סגנונות שונים של ערבול הן בתמונה והן בקול.
  - רצוי להשתמש גם באותיות עבריות על מנת למנוע שימוש בשירותים פותרי CAPTCHA.
  - ניתן לעבוד עם רכיבים כגון BotDetect מ- captcha.com.

- יש לקחת בחשבון את כלל האיומים העלולים לגרור מתקפות Denial Of Service (מניעת שירות) ולגבש בקרות כנגדם.
- במנגנון נעילת משתמשים יש לקחת בחשבון את אלמנט הזמינות כך שיהיה ניתן לשחרר משתמש שננעל בצורה מהירה יחסית. כמו כן ניתן להשתמש ברכיב ה-CAPTCHA לאחר מספר ניסיונות כושלים בכניסה למערכת.

6.12.4. הגנה מפני Buffer overflow

- יש לאמת פרמטרי מחרוזות כקלט ופלט – יש לוודא את אורך המחרוזת שלא תחרוג מהמקסימום.
- יש לאמת גבולות של מערכים.
- יש לאמת אורך נתיב לקבצים.
- בקוד unmanaged (C, C++) יש להימנע משימוש בפונקציות שאינן מומלצות כגון strcpy ותחתן להשתמש בגרסאות הבטוחות של אותן פונקציות כגון strncpy (פונקציות מתוך Strsafe.h).

6.12.5. הגנה מפני מתקפות Network eavesdropping

- הצפנת תווך תקשורת\ הודעה בזמן ביצוע הזדהות.
- הצפנת תווך תקשורת \ הודעה בזמן העברת מידע הקשור בפרטי זיהוי משתמש כגון החלפת סיסמא וכו'.
- הצפנת תווך תקשורת \ הודעה בזמן העברת מידע עסקי/אישי רגיש.

6.12.6. הגנה מפני מתקפות Brute Force & Dictionary Attacks

- מימוש מדיניות סיסמאות חזקה.
- מימוש מנגנון נעילת משתמשים.
- שמירת סיסמאות ע"י שימוש ב Hash בתוספת ערך רנדומאלי (Salt)
- הטמעת רכיב CAPTCHA

6.12.7. הגנה מפני מתקפות Session Hijacking & Session Replay

- אין לאפשר פתיחה של יותר מ Session אחד עבור משתמש (במערכות רגישות בלבד).
- יש לבצע בדיקות אימות ל Session לפני מתן גישה כלשהי.

- יש לבצע שימוש בתווך מוצפן כדי שלא יהיה ניתן לגנוב Cookie המועבר לאפליקציה.
- למניעת מתקפות Replay יש ליצור ערך חד ערכי עבור כל הודעה הנשלחת, כמו כן מומלץ לשלב חתימה בגוף ההודעה – Timestamp.

6.12.8. **מניעת שמירת נתונים במטמון הדפדפן**

- אופציית אחסון הדפים (Caching) תהיה מבוטלת עבור כל הדפים באפליקציה ולכל סוגי הדפדפנים.

6.12.9. **מניעת חשיפת תוכן תיקיות השרת**

- יש לבטל את מאפיין ה-Directory Listing בכל אחת מהתיקיות הווירטואליות על שרת האפליקציה.

6.12.10. **מניעת אפשרות אחסון פרטי הזדהות בדפדפן**

- יש לבטל את אפשרות ה- Password Auto complete ע"י שליחת מאפיין מתאים בתגי ה-Password וה-Form בדף ה-HTML. דוגמא:

<INPUT TYPE="password" AUTOCOMPLETE="off">

6.12.11. **מניעת גישה לדפי בדיקות ודפים שאין אליהם קישורים נדרשים באפליקציה**

- יש למנוע גישה לדפי סביבת הבדיקות ולהסיר אותם מסביבת הייצור של המערכת.
- יש לפעול לפי נוהל העברה מסודרת לסביבת הייצור.

6.12.12. **מניעת התקפות CSRF**

- יש לשלוח עם כל בקשה של דף שדורש אימות TOKEN אקראי, שיבדק בצד השרת על מנת לאמת את המשתמש.

6.13 עקרון הקריסה המבוקרת

6.13.1. עקרון קריסה מבוקרת (Fail Closed) מתייחס למצב בו לאחר קריסה של המערכת, ממשיכים מנגנוני ההגנה לתפקד ואינם יוצרים מצב של "דלתות פתוחות".

6.13.2. יש לבצע הכנות בקוד לתהליכים של קריסת המערכת ולבחון את השפעת הקוד שנכתב על אבטחת המערכת בעת הנפילה. תכנון נכון של קוד יגרום לכך שהמערכת לא תבצע פעולות שדורשות רמת אבטחה מסוימת במידה ורמת אבטחה זו אינה קיימת במערכת באותו רגע נתון.

6.13.3. על הדף המצביע על כשל במערכת להיות כללי ולא לחשוף למשתמש את פרטי השגיאה. הפרטים המלאים יירשמו במנגנון התיעוד של המערכת בצד השרת.

7. נספח איומי אבטחת מידע

7.1 הקדמה

התקפות על רמת האפליקציה יכולות להיות מכוונות כלפי כל אחת משכבות המערכת: בשכבת הפרזנטציה (באמצעות התקפות על שרתי ה- Web או בממשקי המשתמש של המערכת), בשכבת האפליקציה (באמצעות התקפות המנסות לעקוף הגבלות על ביצוע פעולות ועוד) ועל שכבת הנתונים (על ידי התקפות שמטרתן להוציא מידע בצורה בלתי מורשית מבסיסי הנתונים או לבצע שינויים בנתונים).

בסעיפים הבאים מפורטים מספר סוגים של איומים אשר המערכות עלולות להיות חשופות אליהם במידה ולא יינקטו אמצעי האבטחה המתאימים ופיתוח הקוד לא ייעשה באופן מאובטח.

7.2 גניבת זהות בעקבות מדיניות סיסמאות לקויה

במידה ומדיניות הסיסמאות בארגון או באפליקציה מסוימת חלשה, כלומר סיסמא קלה לניחוש, גם – האפליקציה המאובטחת ביותר, תהיה פריצה בקלות יחסית, וזאת בשל הזדהות חלשה. תוקף מיומן יכול להריץ כלים אשר נועדו לשבור סיסמאות, או לנסות ולנחש צירופים הגיוניים, ובכך לפרוץ למערכת ולהתחזות למשתמשים אחרים.

7.3 מניעת User Enumeration

במקרה בו הזנת משתמשים נעשית במספר מקומות - בנוסף על מסך ההזדהות – במקומות כגון שליחת דואר פנימי, שינוי סמא וכו' - יש למנוע מצב בו האפליקציה מגיבה באופן שונה עבור הזנת משתמש קיים ועבור הזנת משתמש שאינו קיים.

7.4 הזרקת שאילתות זדוניות (SQL INJECTION)

לצורך הצגת מידע דינאמי פונה האפליקציה אל בסיס נתונים, בין אם היא מבצעת זאת ישירות ובין אם היא מבצעת זאת דרך שרתי אפליקציה מתווכים. הקשר מול בסיס הנתונים מתבצע באמצעות שפת SQL . בתוך שאילתות ה - SQL מוטמעים בדרך כלל גם פרמטרים אשר מגיעים מהמשתמש (כגון, מחרוזת לחיפוש, שם משתמש וסיסמא...). על

ידי בנייה מתוחכמת של הפרמטרים הללו, יכולים התוקפים במקרים מסוימים לבצע שאילתות בלתי חוקיות בבסיס הנתונים .

7.5 PARAMETER TAMPERING

בעיות אבטחת מידע רבות נגרמות כתוצאה מאי וידוא קלט שמגיע ממשתמשים או מאובייקטים שאליהם אנו - מתממשקים דבר המאפשר לתוקף לבצע שינוי בפרמטרים (Parameter Tampering) אשר יאפשר לו לבצע התקפות שונות על המערכת. תוקף יכול לשנות את הקלט כך שהוא לא יתאים למה שאנו מצפים לקבל. בהתאם למערכת והפונקציונאליות הספציפית שמשתמשת בקלט זה, יכול התוקף לגרום לנזקים שונים החל מהשבתת המערכת, דרך ביצוע פעולות בלתי מורשות וכלה בהשתלטות מלאה על המערכת. במקרים מסוימים אנו מבצעים בדיקות על הקלט ברמת ה - Client מבלי לקחת בחשבון שתוקף יכול בקלות לעקוף בדיקות שמתבצעות ברמת הלקוח (Client)

7.6 DENIAL OF SERVICE - מניעת שירות

התקפת מניעת שירות (DoS – Denial of Service) היא התקפה שיכולה להשבית מערכת שלמה ע"י מניעת שירות מכל משתמשי המערכת או מחלק מהם. התקפה זו מתבצעת ע"י גרימת ניצול קריטי של משאבים בצד השרת. המשאבים יכולים להיות משאבי זיכרון, משאבי דיסק קשיח, משאבי מעבד ומשאבי רשת .

7.7 חריגת הרשאות

היכולת של משתמשים לחרוג מההרשאות המותרות להם היא אחת מבעיות האבטחה הקשות באפליקציות. מערכת ההרשאות של כל אפליקציה היא מורכבת ומשתתפים בה מספר רכיבים. כך למשל קיימות הרשאות ברמת מערכת ההפעלה אשר אחראיות על הגישה לקבצים ומשאבים של מערכת ההפעלה. ישנן הרשאות ברמת בסיס הנתונים אשר אחראיות על הגישה לטבלאות ואובייקטים בבסיס הנתונים וישנן הרשאות של משתמשים ברמת האפליקציה אשר מגדירות באלו תהליכים לוגיים יכול המשתמש לעשות שימוש.
כל מערכות ההרשאה הללו צריכות להתחבר למערכת אחת אשר דואגת לכך שמשתמשים לא יוכלו לחרוג מההרשאות שלהם בכל שלב של עבודה מול המערכת. הבעיה העיקרית היא שלא קל לשלב בין מערכות הרשאה אלו. כך למשל האפליקציה ניגשת למערכת הקבצים ולבסיס הנתונים ובדרך כלל בהרשאות נרחבות ביותר. במידה והמשתמש מצליח

לנצל פרצה מסוימת באפליקציה הוא יכול לגשת למערכת הקבצים או לבסיס הנתונים ולבצע שם פעולות שונות של שליפה ושינוי מידע.

מנגנון ההרשאות צריך להיות מיושם בכל השכבות (שכבת הפרזנטציה, שכבת הלוגיקה העסקית, שכבת הנתונים). יש אפליקציות שמיישמות מנגנון זה בשכבת הפרזנטציה בלבד. באפליקציות מסוג זה מסתמכים על כך שהמשתמש רק ילחץ על הקישורים והכפתורים שקיבל בתצוגה אך בפועל משתמשים זדוניים יכולים להשיג קישורים ישירים אל לב ליבה של המערכת וכך לעקוף בקלות את מנגנון ההגנה הזה.

כמו כן, עצם העובדה שכל אחד מהרכיבים שתוארו מפותח בדרך כלל על ידי גורם אחר, לא מקל על הבעיה. האפליקציה מפותחת על ידי מספר מפתחים ובסיס הנתונים מפותח על אנשי בסיס נתונים ובמרבית המקרים הקשר הרופף בין כל הגורמים יכול ליצור פרצות במעבר בין שכבה לשכבה.

7.8 טעויות קונפיגורציה

טעויות קונפיגורציה יכולות לגרור בעיות אבטחה קשות. טעות קונפיגורציה יכולות להופיע הן במוצרי התשתית (לדוגמה: שרת WEB) והן באפליקציה עצמה במידה והמתכנתים הכניסו אופציה לכך. במהלך התקפה זו מנסים התוקפים לנצל קונפיגורציות ברירת מחדל או ליקויים בהגדרות הקונפיגורציה של אותו רכיב על מנת לבצע פעולות זדוניות שונות במערכת.

7.9 "דלתות אחוריות" ואופציות DEBUG

בעת תהליך כתיבת הקוד של מערכות גדולות ומסובכות, נוטים לפעמים המתכנתים להשאיר "דלתות אחוריות" (Backdoors) אשר יכולות לאפשר להם גישה לנקודות שונות במערכת ללא ביצוע הזדהות מסודרת, או לאפשר להם לבצע פונקציונאליות מסוימת שהמערכת לא אמורה לאפשר למשתמשים. במקרים מסוימים שוכחים המתכנתים להסיר את הדלתות האחוריות הללו בסוף התהליך וגורם זדוני אשר גילה אותן יכול לנצלן לרעה. במקרים אחרים, מושארות הדלתות האחוריות במתכוון מתוך כוונות זדוניות ולכן יש לבצע בדיקות של כל קוד לפני העברתו לייצור ולוודא שאין בו "דלתות אחוריות" או אפשרויות DEBUG אשר עלולות לאפשר פגיעה במערכת או השתלטות עליה.

7.10 BUFFER OVERFLOW

חוצץ (Buffer) הוא אזור בזיכרון המחשב המוקצה לאחסון רצף של נתונים בכלל, ולאחסון זמני של נתונים בפרט. סוג הנתונים הנשמר בחוצץ הוא בד"כ מחרוזות תווים, שורות קלט מקבצים, הודעות תקשורת וכו'. בד"כ משמש החוצץ לשמירת הנתונים בדרך מנקודת הקלט (מקלדת, דיסק, תקשורת) אל נקודת העיבוד. במקרים רבים עובר רצף תווי הקלט דרך מספר חוצצים עד לנקודת העיבוד. הסיבה לכך היא בראש ובראשונה אופן הבניה המודולרי של מערכות תוכנה גדולות.

גלישה היא מצב בו התוכנה כותבת אל חוצץ יותר נתונים מכפי גודלו. מכיוון שהחוצץ הנו בעל גודל מוגדר וסופי, הרי שכתוצאה מפעולה זו גולשים הנתונים העודפים אל מחוץ לחוצץ ודורסים את התאים הנמצאים מייד אחרי החוצץ בזיכרון.

גורם המעוניין לתקוף את המערכת נדרש לגלות אילו מנתוני הקלט אינם נבדקים כנגד גלישה. לאחר שהתוקף זיהה את הנקודה הבעייתית כל שנותר לעשות הוא לנחש כמה ארוך צריך להיות הקלט כדי ליצור גלישה ואז להשתמש בטכניקות ידועות כדי לנצל אותה. בכל אחד מרכיבי התוכנה, ללא קשר למקורם, עלולות להימצא בעיות גלישה.

הרכיבים המועדים ביותר להמצאות גלישות הם רכיבי התוכנה המורכבים ביותר כגון שירותי מערכת ההפעלה או שירותי תקשורת מורכבים. כמו כן ישנה רגישות גבוהה לבעיות גלישה בתוכנות אשר נבנו בשיטה של טלאי על טלאי. מלבד רכיבי התשתית המוכרים בהם מתעוררות בעיות גלישה, הרי שגלישות יכולות להיווצר גם באפליקציות שפותחו בארגון עצמו או אפליקציות שפותחו עבור הארגון ע"י צד שלישי. כמו כן, בהחלט ייתכנו בעיות גלישה ברכיבים בסיסיים יותר כמו רכיב ה - BIOS של המחשב (למעשה, התגלו בעיות כאלה בעבר). גלישת חוצצים יכולה לאפשר לתוקף לפגוע בזמינות המערכת או להשתלט על המערכת .

7.11 BYPASS FLOW - עקיפה לוגית

בשכבת הלוגיקה העסקית, קיימים לרוב תהליכים המורכבים ממספר שלבים. שלבים אלה יוצרו על מנת לספק למשתמש תהליך עבודה. אחת מההתקפות המהותיות ביותר כנגד מנגנון מסוג זה, היא לנסות ולעקוף את הלוגיקה והתהליכים אשר אליהם התכוון מתכנן המערכת. דוגמא טובה לכך היא תהליך פתיחת חשבון המחייב מספר שלבים: הראשון הקלדת נתוני החשבון והשני אישור מסגרת האשראי בחשבון. משתמש זדוני, ינסה לעקוף את תהליך אישור מסגרת האשראי, ולעבור לתהליך הבא אחריו ישירות, תוך דילוג על שלב

מסוים מתוך התהליך הסדור. לכן, מפתחי המערכת צריכים למנוע מצבים כאלה ולוודא זרימה נכונה של התהליך ללא יכולת לדלג על שלבים.

7.12 נפילה לא מאובטחת של אפליקציות

אפליקציות 'נופלות' כתוצאה מבאגים שונים באפליקציה או בתקלות בתשתית עליה האפליקציה 'רצה'. חלק מבעיות האבטחה של מערכת נחשפות רק כאשר המערכת 'נופלת'. המערכת יכולה 'ליפול' לגמרי או חלקית, תוך כדי שחלקים מסוימים במערכת לא מתפקדים וחלקים אחרים כן מתפקדים. בעת 'נפילת' המערכת עלול להיווצר מצב שבו פונקציונאליות מסוימת האחראית על אבטחת המידע איננה עושה את פעולתה וכתוצאה מכך המערכת נותרת ללא אבטחה נאותה. כמו כן 'נפילה' יכולה לחשוף את הארכיטקטורה של המערכת ע"י הצגת הודעות שגיאה מפורטות.

7.13 יירוט התעבורה (MAN IN THE MIDDLE)

תעבורת המידע בין מודולי וממשקי המערכת השונים עוברת בדרך כלל דרך רכיבי תקשורת רבים אשר לא כולם בשליטתנו. בכל רכיבי התקשורת, במידה ולא מתקיימת הצפנה של המידע וזיהוי חד ערכי של הגורמים המורשים למידע, קיימת יכולת של גורמים עוינים המאזינים לתווך התקשורת ולנתונים העוברים בין המערכות ובין המודולים שלהם לגנוב את המידע או לשנותו. כמו כן אותם גורמים יכולים להקים שרת מתחזה הדומה לשרת המקורי (מה שנקרא Phishing Attack) וכך לגנוב פרטי הזדהות או מידע רגיש העוברים בדרך או אל השרת המתחזה.

7.14 ניתוח פרוטוקולים

מערכות המבוססות שרת לקוח נוטות לבסס חלק מאבטחת המידע של המערכת על בדיקות שמתבצעות ברמת היישום בצד הלקוח. במקרה זה, על ידי ניתוח הפרוטוקול שעובר ברשת בין הלקוח לשרת, ניתן במקרים רבים לעקוף את מנגנון האבטחה של המערכת ולשלוח פקודות בלתי מורשות למערכת.

7.15 פרצות במוצרי צד שלישי

במקרים רבים האפליקציה מותקנת על שרת אשר מהווה מוצר מדף או משתמשת ברכיבי תשתית שונים מתוצרת צד שלישי. מדי פעם, עלולות להתגלות באותם מוצרי מדף, רכיבי תשתית ורכיבים מצד שלישי פרצות אבטחת מידע ידועות אשר עלולות לשמש תוקפים ולאפשר להם להשיג מידע על המערכת הנתקפת ולבצע פעולות בלתי מורשות בה. במהלך התקפה זו מנסים התוקפים לנצל את הפרצות הידועות המפורסמות בדרך כלל באתרי WEB ברחבי האינטרנט שנמצאו בתוכנות התשתית של המערכת ולפרוץ באמצעותם ליישום או במקרים מסוימים לקבל שליטה מלאה על המערכת.

7.16 נעילת מוות (DEAD LOCK)

נעילת מוות (Dead Lock) היא מצב שבו תהליך A נועל קובץ א' ואז מנסה לקרוא משהו מקובץ ב'. בו בזמן תהליך B נועל קובץ ב' ואז מנסה לקרוא משהו מקובץ א'. שני התהליכים לא ישחררו את הקבצים שהם נעלו עד שהם לא יצליחו לקרוא מהקובץ. זוהי נעילת מוות שלא תשתחרר לעולם. כאשר מצב זה אפשרי משתמש בודד יכול לגרום לנעילה של משאב מסוים ובכך להשבית פונקציונאליות עבור משתמשים אחרים.

7.17 מרוצים - RACE CONDITIONS

חשוב לזכור שאפליקציות לא רצות בתהליך אחד ולא עובדות מול משתמש אחד. השימוש במספר תהליכים במקביל עלול לגרום לבעיות אבטחת מידע שונות כגון מרוץ. מרוץ הוא מצב שבו התוצאה של פעולה מסוימת תלויה בשני תהליכים או יותר ונקבעת על ידי הסדר שבו תהליכים אלו אמורים להתבצע. מערכות הפעלה לא מבטיחות לנו ששורת קוד ב' תתבצע מיידית אחרי שורת קוד א' מבלי ששום דבר אחר יתבצע ביניהן. תזמון תהליכים יכול לגרום למצב שבו שורת קוד א' תתבצע ולאחריה יפעל תהליך אחר לגמרי לפרק זמן מסוים ואז יחזור התהליך הראשון ותתבצע שורת קוד ב'. מצב זה יכול לגרום לבעיות אבטחה שונות אם הוא לא נלקח בחשבון מלכתחילה. דוגמא פשוטה היא כאשר שורת קוד א' יוצרת קובץ במערכת ההפעלה ואז שורת קוד ב' משנה את ההרשאות של הקובץ כך שגורמים לא מורשים לא יוכלו לגשת לקובץ. מצב שבו תהליך אחר מופעל בין שורת קוד א' לשורת קוד ב' יכול לאפשר לאותו תהליך לכתוב לאותו קובץ או להתחבר אליו כך שגם לאחר הניסיון של התהליך הראשון לשנות את ההרשאות, עדיין תהיה לתהליך השני אפשרות גישה לקובץ.

7.18 איומים ייחודיים לאפליקציות WEB

להלן תיאור ההתקפות והסיכונים העיקריים הבאים לידי ביטוי באפליקציות בסביבת WEB:

7.18.1 מניפולציות שדות Hidden

בהתקפה זו מנסה התוקף לשנות ערכים שונים המגיעים לדפדפן "מוחבאים" על ידי שימוש בשדה Hidden . במקרים מסוימים האפליקציה עלולה להסתמך על ערכים אלו ועל ההנחה כי הם נסתרים מהמשתמש ואין ביכולתו לשנותם. הנחה זו איננה נכונה מאחר ומשתמש בעל כוונות זדוניות יכול לבצע שינויים מושכלים של ערכים אלו ולשלוח אותם בחזרה לשרת. כך לדוגמא, אם האפליקציה השתמשה ב- Hidden Fields על מנת לאחסן את מספר הלקוח ועל פיו מעניקה הרשאות, התוקף יכול לשנות את מספר הלקוח ולקבל במקרים מסוימים גישה לחשבונות של משתמשים אחרים.

7.18.2 הרעלת Cookies

שימוש במנגנון ה Cookies מאפשר לאפליקציה לשמור מידע במחשב המשתמש. מערכות ואפליקציות בלתי מאובטחות ששומרות נתונים זמניים הרלוונטיים לאותו משתמש ב Cookie כגון מחיר של המוצר שהוא קנה עכשיו ועל פיו מבצעות החלטות במערכת או מתבססות על נתונים אלה ללא בדיקה של תקינותם עלולות לגרום לפרצות ביישום ולאפשר התקפה על המערכת. בהתקפה זו התוקף מנתח את ה - Cookies שהאפליקציה שולחת למשתמש ובודק האם ניתן לבצע שינויים ב – Cookie אשר יאפשרו לו גישה למידע של משתמשים אחרים או לבצע פעולות שאין לו הרשאה לבצע. יש להבין כי נתונים הנשמרים ב Cookies הינם קלטים למערכת לכל דבר והמערכת צריכה לבדוק את תקינותם בצד השרת. יש לצאת מנקודת הנחה כי גורמים זדוניים ינסו לשנות מידע זה על מנת לתקוף את המערכת או לפרוץ אליה. כמו כן, חלק מהאפליקציות יכולות להצפין את ה - Cookies לפני העברתם למשתמש אך אם נשמר בהם מידע הנוגע לזיהוי או להרשאות עלול להיווצר מצב בו גורם שיגנוב Cookie כזה יוכל לבצע Replay Attack ולהשתמש ב Cookie זה על מנת להתחזות למשתמש חוקי או לקבל הרשאות גבוהות יותר. לכן, אין להסתמך על נתונים הנשמרים ב Cookies ללא בדיקה של תקינותם, ולהשתמש בהם בצורה מושכלת.

7.18.3 Forceful Browsing

לכל אפליקציה יש נקודת גישה מרכזית אחת או יותר. בדרך כלל מדובר בדף ההזדהות של האפליקציה דרכו חייב המשתמש לעבור. לאחר למידה מדוקדקת של האתר, ניתן לפעמים לגשת ישירות לדפים בשרת שהמתכנתים לא תכננו שניגש אליהם ישירות, ללא מעבר דרך דף ההזדהות. במידה ובאותם דפים שאליהם אנו ניגשים ישירות לא מתבצע תהליך מחודש של בדיקת ההזדהות של המשתמש, אזי ניתן יהיה לעקוף את מנגנון ההזדהות של המערכת ו/או לקבל גישה לקבצים ונתונים אשר איננו מורשים לראות.

7.18.4 XSS - Cross Site Scripting

התקפה זו הנפוצה כיום בעיקר ביישומי WEB מאפשרת לתוקף לנצל את אי בדיקת תקינות הקלט והפלט ברמת ממשק התצוגה של האפליקציה על מנת להריץ קוד זדוני במחשב המשתמש או במערכות משיקות המשתמשות בקלט הזדוני. הפעולות אותן ניתן לבצע כוללות בין השאר:

- **גניבת זהות** – גניבת משתנה ה – SESSION המשמש לזיהוי המשתמש מול המערכת ושליחתו לתוקף אשר יוכל להשתמש בו על מנת לבצע התחזות למשתמש לגיטימי ולבצע פעולות בשמו.
- **ביצוע הונאות** – שתילה של הודעות כוזבות אשר נראים כאילו הגיעו מהמערכת אשר ישכנעו את המשתמש להזין פרטים רגישים (כגון סיסמה , פרטי כרטיס אשראי) תוך ניצול בטחונו במערכת ואשר יאפשרו שליחת המידע הרגיש לתוקף או ניצול המידע לרעה.
- **שינוי מראה אתר** – ביצוע Defacement לאתר המערכת תוך פגיעה תדמיתית בארגון.
- **שתילת סוסים טרויאניים ותוכנות זדוניות –** שתילת סוסים טרויאנים ותוכנות זדוניות במחשב המשתמש תוך ניצול פרצות בדפדפן המשתמש. קיימת ספריה בתשתית Net. בשם antiXss המספקת פונקציונאליות להגנה כנגד התקפת XSS .

7.18.5 CSRF -Cross Site Request Forgery

- מתקפה זו נפוצה מאד באפליקציות Web וכל פונקציה ללא הגנה ייעודית הינה מטרה להתקפה.
- על מנת להתמודד עם מתקפות אלו יש לעשות שימוש באמצעים הבאים:

- Captcha
- הזדהות מחודשת (Re-Authentication) – דבר המחייב את התוקף לדעת את הססמא של הלקוח.
- חתימה חד פעמית (One Time Token) – דבר המחייב את התוקף לדעת מהי החתימה נכון לאותו רגע.
- חתימה ייחודית (Unique Request Token) – דבר המחייב את התוקף לדעת מהי החתימה הייחודית לגולש מסוים בשיח (Session) מסוים.
- בנוסף, קיימת ספריה המספקת פונקציונאליות ומענה להתקפה זו בשם OWASP CSRFGuard Project

7.18.6 הזדהות באמצעות שדה Referrer

- שדה ה-Referrer נשתל על ידי הדפדפן ונשלח לשרת ה-WEB עם כל בקשה. השדה מציין את שדה ה-URL האחרון בו ביקר המשתמש לפני שליחת הבקשה. אין להשתמש בשדה זה לצרכי הזדהות או מתן/קבלת הרשאות מאחר והוא ניתן לשינוי וזיוף בקלות.

7.18.7 סמאות

- כאשר המשתמש מקיש את הססמא היא צריכה להיות ממוסכת (באמצעות כוכביות או בולטים). כל תקשורת בה מועברים מזהים בכלל וסמאות בפרט צריכה להתבצע בתווך מוצפן.

7.18.8 תקשורת בין המשתמש לשרת

- פרוטוקול ה-SSL מאפשר למנוע האזנה או שינוי של התשדורת בין המחשב של המשתמש לבין שרת ה-WEB. פרוטוקול זה מבטיח למשתמש כי הוא יוצר תשדורת עם השרת הנכון ולא עם שרת מתחזה ומבטיח כי המידע הרגיש אשר הוא מעביר לאתר עובר באופן מאובטח ממנו לאתר ובכיוון השני. חובה לעשות שימוש בגירסת TLS מאוחרת ועדכנית. נכון למועד כתיבת גירסה זו של המסמך מדובר בגירסה 1.2.
  - יש לוודא כי התעודה בה משתמש שרת ה-WEB היא חוקית, תקינה ובתוקף.
  - יש להשתמש במפתח שאורכו 256 ביט או ארוך יותר.
  - יש לוודא כי אורך המפתח בו נעשה שימוש הוא המכסימאלי נכון למועד בניית המערכת.

7.18.9 אימות קלט

- אין להסתמך על בדיקות הקלט המתבצעות בצד הדפדפן.
- אין להניח מראש כי המידע המתקבל מצד המשתמש הינו נכון גם אם מדובר במידע אשר נשלח תחילה משרת ה-WEB.
- אין להסתמך על אורך הקלט (Size=X) בלבד כאמצעי לאימות הקלט המתקבל.

7.18.10 איסור התחברות (Login) אוטומטית

- אין להשתמש בלוגין אוטומטי של משתמשים על בסיס Cookies או כל שיטה אחרת. משתמשים חייבים לבצע Login ולהעביר את נתוני הזיהוי שלהם בכל פעם שהם ניגשים מחדש לאפליקציה.

8. נספח אמצעי הגנה

נספח זה מפרט את הכלים באמצעותם ניתן – עוד בשלב פיתוח התוכנה - להגן על מרכיבי אבטחת המידע. כלים אלו נחלקים לשלוש קבוצות – ע"פ רכיב אבטחת המידע שעליו הם נועדו להגן:

- אמצעי הגנה על סודיות המידע
- אמצעי הגנה על שלמות המידע
- אמצעי הגנה על נגישות/זמינות המידע

8.1 אמצעי הגנה על סודיות המידע

אמצעי	אופי הגנה	דוגמאות
הצפנה	אמצעי מנע, הפועל על ידי הורדת הסבירות להתרחשות הנזק באמצעות הרמת עלות התקיפה הנדרשת כדי לשבור את מנגנון ההצפנה	הצפנת תקשורת בין רכיבים בתוך המערכת, הצפנת תקשורת עם רכיבים משיקיים, הצפנת מידע על גבי מדיית אחסון (למשל: קובץ סיסמאות, רשומות בבסיס הנתונים), הצפנת מידע על מערכת הגיבוי, מנגנון Challenge-Response להזדהות.
Watermarking	אמצעי מנע המשמש הסוואה ופועל כ "דיו סתרים" דיגיטלי כדי להסתיר את עצם קיומו של המידע (בשונה מהצפנה שמנסה להסתיר את תוכנו) פועל על ידי צמצום חשיפה והרמת עלות התקיפה	אמצעי Watermarking משמשים להחבאת מידע רגיש בתוך מידע נושא (Cover) באופן שקשה לזהות את קיום המידע או להסירו: העברת מידע סודי בתוך תמונות/אודיו, החבאת מנגנוני הגנה בתוך מורכבות פונקציונאלית של מערכת התוכנה, וכדומה.

פיתיון, Fingerprinting	אמצעי גילוי לחשיפת מידע שפועל על ידי הוספת מידע ייחודי אך שגוי למאגר הרגיל. מידע זה, כאשר נחשף בתווך גלוי משמש אות שבוצעה חדירה, ומאפשר לזהות את מקורה.	הכנסת ידיעות כוזבות למערכת מודיעינית פנימית, הכנסת תיק רפואי פיקטיבי של אישיות בעלת חשיפה תקשורתית גבוהה למאגר הרפואי כדי לגלות אם מתבצעת הדלפה, הכנסת מידע ייחודי לתמונות/אודיו כדי לטפל בבעיית זכויות יוצרים (לזהות איזה משתמש חוקי מאפשר העתקת המידע).
החלפת סוד	אמצעי שפועל לשמירת "טריות" המערכת. יכול לפעול בשני אופנים:כאמצעי מנע לצמצום חלון החשיפה, או כמנגנון תגובה על ידי יצירת תחליף בתגובה לחשיפה.	החלפת מפתחות הצפנה (Revocation), החלפת סיסמאות באופן קבוע, החלפת מנגנוני הגנה/נתיבי גישה למשאבים, החלפת כתובות רשת, הרשאות לזמן מוגבל.
מחיקת מידע	אמצעי מנע שפועל לצמצום החשיפה על ידי מחיקת המידע הרגיש ממדית האחסון ברגע שאינו נחוץ יותר.	מחיקת מפתחות הצפנה מהזיכרון/קבצי מערכת. ניקוי רכיבי סנכרון/קבצים זמניים. מחיקת זיכרון שמכיל סיסמאות, מחיקת מידע מקורי לאחר הצפנתו.
אקראיות	אמצעי מנע שמנסה למנוע אפשרות לחזות תכולת מידע רגיש במערכת, ובכל להעלות את עלות התקיפה.	סיסמאות אקראיות שמוגרלות על ידי המערכת, סיסמאות מעורבות (תווים + ספרות) מפתחות הצפנה אקראיים, מספרים סידוריים עם התחלה אקראית וכן הלאה.
הורדת רזולוציה/ הוספת רעש	אמצעי מנע לצמצום חשיפה במערכות בהן נדרשת חשיפה	מערכת רפואית שמנסה להגן על פרטיות החולה, ועדיין

	חלקית. מנגנון ההגנה מונע גישה למידע שלם, אבל מאפשר גישה למידע חלקי.	לאפשר שאילתות סטטיסטיות ההגנה תתבצע על ידי: הסתרת פרטים מזהים (שם, גיל), מניעת שאילתות סטטיסטיות על קבוצות קטנות, הוספת רעש אקראי למדגם, שינוי פרטים מזהים (למשל תווי פנים בתמונה) וכדומה.
תגובה מושהית	אמצעי מנע שפועל לצמצום החשיפה והאטת התפשטות הנזק. מנגנוני תגובה מושהית משתמשים להסתרת אמצעי ההגנה עצמם על ידי ביצוע התגובה רחוק מזמן איתור התקיפה.	הצגת הודעת שגיאה רק לאחר הקלדת שם+סיסמא, גם אם השם אינו קיים. הפעלת מנגנוני תגובה (למשל חסימת פעולות) בהשהיה ביחס ביצוע העברה, הפניית התוקף סביבה מוגנת תוך ניסיון לאתר את מקור התקיפה.
השמדה עצמית	אמצעי תגובה שמבצע השמדת מידע רגיש כתגובה לזיהוי חדירה. על פי רוב משתמש להגנה על מנגנוני ההגנה עצמם.	מנגנון Tamper resistance בכרטיסים חכמים, שמבצע מחיקה של מפתחות ההצפנה בתגובה לניסיונות חדירה פיזיים (באמצעות מעגל חשמלי מיניאטורי שנסגר עם חיבור probe למעגל הראשי, או איתור ניסיונות לניתוק מקור הכוח), סיסמאות משתמש שנמחקות לאחר מספר ניסיונות שגויים עד שמנהל המערכת יוצר חשבון חדש עבורו.

8.2 אמצעי הגנה על שלמות המידע

אמצעי	אופי הגנה	דוגמאות
שער (Gate)	אמצעי מנע נפוץ שפועל באמצעות ניתוב כל הגישות למידע למספר מוגבל של נקודות מוגנות היטב. באופן זה מושגת הקטנת חשיפה יחד עם הרמת עלות התקיפה.	בניית בקרת גישה במערכת ברוטינה אחת, מערכות Firewall , הזדהות מול מערכת הפעלה או Database ככלי בקרת גישה מרכזי למערכת, מערכות Single Sign On
אישור	אמצעי מנע הפועל להורדת סבירות הפגיעה באמצעות בקשת אישור לביצוע הפעולה, או הספקת אישור שהפעולה בוצעה	אזהרה לפני ביצוע פעולה במערכת (יציאה, מחיקה, ביצוע), אישור בפרוטוקול תקשורת (אישור קבלה)
אישור כפול	אמצעי מנע הפועל להורדת סבירות הפגיעה באמצעות חלוקת אחריות ודרישת אישור נוסף (למשל על ידי גורם מאשר בלתי תלוי) לביצוע פעולה.	כפל חתימות בפעולה מורכבת ,כפל מפתחות במערכת נשק גרעינית, זיהוי מחודש לפני בצוע פעולה רגישה במיוחד, הזדהות כפולה בכניסה למערכת (למשל סיסמא + טביעת אצבע, כרטיס מגנטי + קוד)
Checksum, Hashing	אמצעי זיהוי נזק למידע על ידי הוספת נתון שמאפיין את המידע באופן חד ערכי ביחס מידע ששודר מנקודה מסוימת.	סיפרת ביקורת בשדה ת.ז., בדיקה שרכיבי מערכת לא השתנו (נוסח Tripwire), חתימות התקפה על מערכת (חתימות באפליקציות אנטי

		וירוס או - IDS), חתימת יחידת מידע משודרת (TCP/IP), הוספת CRC לבדיקת שלמות בקובץ שמועבר בין מערכות שונות.
Error correction codes	אמצעי זיהוי ותגובה להתאוששות ותיקון נזק. קודי תיקון שגיאות משתמשים ביתירות מידע על מנת לתקן שגיאות אקראיות בזמן העברת יחידות מידע	רכיבי תקשורת, רכיבי חומרה וזיכרון.
בדיקת תחביר/תוכן	אמצעי לזיהוי מידע לא תקין, על פי רוב בזמן קלט ממערכות חיצוניות לדוגמא: משתמש, מערכת ההפעלה, מערכות משיקות רכיבי צד ג'. המערכת מבצעת בדיקות באם המידע הנקלט מתאים להנחות התחביריות והתוכניות שהיא מניחה לגביו, ולא מאפשרת כניסת מידע שאינו עומד בתנאי הכניסה	בדיקות פורמט שדות (ערך אלפא נומרי בשדה - תאריך), בדיקות ולידציה (ערכים בטווח נכון או בתנאים לוגיים מתאימים, יחס בין ערכים ברשומה נקלטת), בדיקת קודי שגיאה מרוטינות של מערכת ההפעלה.
בדיקת אנומליות	אמצעי לזיהוי פעולות מותרות ואסורות על ידי איתור התנהגות חריגה במערכת, או למידע בעל משמעות שגויה.	מערכות בנקאיות (סכומים לא שגרתיים, בהיקף לא שגרתי ללקוח, סדר פעולות תמוה), מערכות רפואיות (סדר טיפולים לא שגרתי, תרופות לא מתאימות, מינון לא תקין)
Double Entry	אמצעי זיהוי שנועד לאתר בעיות שלמות מידע באמצעות תכונה שחייבת להישמר	במערכות הנהלת חשבונות (למשל בנקאות), כל התנועות חייבות להתאזן. במערכת מלאי, יריד

	במערכת לאחר כל פעולה מותרת. אם המערכת אינה מכילה את אותה תכונה. פירושו של דבר שנפגעה שלמות הנתונים.	כמות מלאי חייבת להתבטא דרך פעולת מכירה של כמויות בסכומים זהים, בכניסה פיזית של אנשים לאתרים מאובטחים, מספר הכניסות חייב להתאזן עם מספר היציאות כדי לוודא שאיש לא נותר בפנים בסוף יום.
חתימה דיגיטלית	אמצעי זיהוי משתמש. על פי רוב פועל בשיתוף עם מנגנון Hashing (לעיל) כדי להבטיח שלמות מקור המידע יחד עם שלמות פנימית של המידע עצמו.	זיהוי שרת בפרוטוקול SSL , מערכות PKI המאפשרות חתימה על מסמכים. Message Authenticating Code (MAC) : זיהוי חד ערכי של שולח ומשלוח
הזדהות הדדית (Handshake)	מנגנוני הזדהות הדדית פועלים לזיהוי של כל הצדדים המעורבים בפעולה. אמצעים אלה פועלים למניעת התקפות Man-in-middle (מצב של התחזות לאחד הצדדים)	מערכת Kerberos , פרוטוקולי Key Exchange בהצפנה. מנגנון הזדהות בין כרטיס חכם ל- Reader . מנגנון הזדהות של המערכת למשתמש ולהפך.
Callback	אמצעים לזיהוי מקור המידע (למשל משתמש) על ידי ניתוק הקשר, והתקשרות מחודשת של ספק השירות ללקוח. (תוך בדיקה אם הלקוח זכאי לשרות כזה)	התקשרות יזומה למערכות משיקיות, לאחר שאלה מבקשות מידע, חיוג חוזר של מערכות תחזוקה אל המפעיל.
Audit Trail	אמצעי השארת ראיות, הפועל על ידי תיעוד כל פעולה, בעלת השלכה אבטחתית, על גבי מדיום נפרד מהמערכת הראשית, באופן שמאפשר ניתוח שלאחר מעשה	קבצי Log של מערכות הפעלה או אפליקציות, דוחות שינויים/הבדלים תקופתיים, מערכות היסטוריה

גישה אקסקלוסיבית	אמצעי מנע שמוודא גישה אטומית ליחידת מידע כך שלא נוצר מצב בו שני גורמים עצמאיים מנסים בו בעת לשנות את המידע, כך שבסוף התהליך מתקבל מידע שגוי.	נעילת רשומות/שדות/טבלאות במערכת, נעילת משאבי מערכת הפעלה משותפים שנמצאים בשימוש (זיכרון, קבצים, מנגנוני סנכרון).
Rollback	אמצעי תגובה שמוודא ביצוע אטומי של פעולה (או סדרת פעולות) על ידי שיחזור מצב קודם שידוע שהינו תקין, אם פעולת עדכון הנתונים לא הסתיימה בשלמותה.	מגנוני Transaction Processing בבסיסי נתונים, מערכות גיבוי/העברה להיסטוריה. קליטת אוסף רשומות ממערכות משיקות כיחידה שלמה.
מספר סידורי רציף	אמצעי גילוי שנועד לגלות בעיות שלמות ברצף של פעולות (שלמות הטרנזקציה) על ידי שמירה של רצף, בו חייבים כל הערכים להופיע, ובסדר הנכון.	מספר חשבונית רציף, זמן ביצוע פעולה, מספר רשומה בתוך Batch , מספר Packet בפרוטוקול תקשורת, מערכות Time stamping .
Session	הוא אמצעי נפוץ שמנסה לצמצם חשיפה שכרוכה בהזדהות משתמש עבור ביצוע כל פעולה. מנגנון Session מלווה את המשתמש משלב ביצוע הזיהוי הראשוני (Login) ועד סיומו, ומזדהה עבורו לביצוע פעולות. מערכות אלה כוללות בדרך כלל Sign-Off אוטומטי בעקבות חוסר פעילות כדי לצמצם עוד יותר את חלון החשיפה	מערכות הפעלה, מערכות Single sign on , פרוטוקולי תקשורת (למשל TCP)
Fingerprinting	מנגנוני Fingerprinting פועלים להרמת עלות התקיפה	הוספת salt לפני הצפנת סיסמאות בקובץ סיסמאות,

	על ידי יצירת גרסאות יחידניות של מערכת ההגנה. שיטה כזו מקשה על בניית מנגנון התקפה אוטומטי.	שידור בתדר מדלג (spread spectrum) שמונע חסימת שידור, מנגנון מוטציה שמשמש וירוסים להגנה עצמית מפני גילוי.
פיתיון (Decoy)	מערכות שמטרתן לצמצם את סבירות התקיפה על ידי הכנת אוסף מערכות דמי מקבילות שניתנות להקרבה, כך התוקף מבזבז משאבי תקיפה על מערכות חסרות חשיבות	מלכודת דבש - מימוש מערכת באופן פריץ במתכוון בתוך הרשת הרגילה על מנת שתהיה יותר מושכת לפריצה, עם נתונים מזויפים ועם מערכות לאיסוף מידע על התוקף
בלימה הדרגתית	בלימה הדרגתית היא יותר ארכיטקטורה מאשר אמצעי הגנה בודד. מערכות הגנה הדרגתיות מנסות למנוע אזעקות שווא על ידי בניית ההגנה במעגלים פנימיים כאשר כל מעגל הוא בעל סף חדירה גבוה יותר, ובעל רגישות גילוי גבוהה יותר. באופן כזה נחסמות ההתקפות הפשוטות בקלות על ידי המעגל החיצוני, ומערכת ההגנה יכולה להתייחס ברצינות רבה יותר לתקיפות מורכבות.	בדיקת אנומאליות של מידע רק לאחר שהסתיימה בדיקת תחביר, מערכת Intrusion detection מאחורי מערכת Firewall, מערכת שחוסמת גישה לאחר שלושה ניסיונות כושלים אבל מתריעה למנהל המערכת לאחר עשרה ניסיונות.

8.3 אמצעי הגנה על נגישות המידע

אמצעי	אופי הגנה	דוגמאות
הוספת אמצעי גישה למשאבים	אמצעי מנע הפועל בדרך של יתירות על ידי הכפלת נתיבי גישה למשאבים, כדי לוודא שגם אם נחסם נתיב גישה יחיד, יכולה המערכת לאפשר גישה (לפחות חלקית).	מערכת RAID , מערכות Clustering , הכפלת רכיבי תקשורת, מערכות אל פסק.
מעקף	מעקף הוא מנגנון Fallback אקטיבי למקרה שגישה למשאבים קריטיים נחסמת. מעקף מעביר את המידע (או את נתיבי הגישה אליו) למערכת חליפית על מנת לצמצם את הנזק.	מערכות אל פסק, שרת נתונים משני עם נתונים - חלקיים, בסיס נתונים משני עם יכולות קריאה בלבד.
סינון / ויסות	אמצעי תגובה שנועד לנהל גישה למשאבים באופן דינמי על ידי בקרה על תווך הגישה למשאבים ובדיקת בקשות הגישה אליהם.	תיעדוף משתמשים בגישה למשאבים על בסיס עומס, Load balancing של אמצעי גישה למשאבים, ניתוק אפיק תקשורת ממנו מתבצעת תקיפה, שינוי הקצאת מקום בשרת על בסיס צורך, override גישה למשתמשים בפרופיל גבוה בזמן חרום.