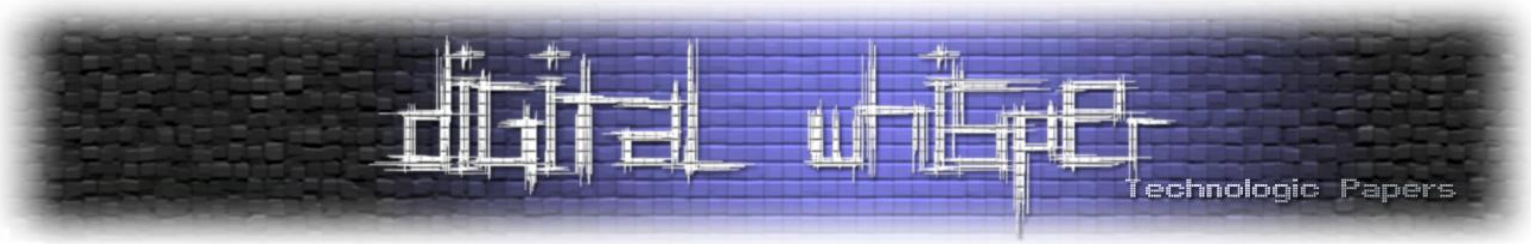


WAF - Web Application Firewall

מאת נתנאל שיין

> “The mantra of any good security engineer is: 'Security is a not a product, but a process. It's more than designing strong cryptography into a system; it's designing the entire system such that all security measures, including cryptography, work together.”
> - Bruce Schneier

הקדמה

נסו להזכר איך נראו אתרי האינטרנט הראשונים שהכרנו- טקסט כתוב על ידי בעל האתר ומספר תמונות להמחשה. מטרת אתרים אלו היתה להעביר מסמכים למיניהם על גבי הרשת בין דפדפנים שונים (ולשם כך נכתב בעצם ה-HTTP). הואיל ואתרים אלה היו כה פשוטים, לא היה צורך לדאוג לאבטחתם. אמנם, עם השנים נכנסו לחיינו ישומי הרשת. אותם ישומי רשת הם אלו שמאפשרים לגולשים באתר להשאיר בו את חותמם האישי, ליצור תכנים ולשתף אותם עם גולשים אחרים באתר.

בזכות ישומי הרשת, הפכו אתרים מהדור החדש לאתרים דינמיים, בהם התוכן מתעדכן מרגע לרגע, לא רק על ידי בעל האתר אלא גם (ובעיקר) על ידי הגולשים. אך כמובן, כמו בכל מצב, גם כאן אליה וקוץ בה- יחד עם כניסת יישומי הרשת לחיינו נכנסו גם כלל הסכנות שהם יוצרים. אחת מסכנות אלה הינה "הזרקת קוד", כלומר בעיית Injections Flew, בה התוקף יכול לשתול קוד זדוני דרך היישום למערכת אחרת כגון מערכת הפעלה או מסד הנתונים.

למעשה, מאמר זה יתמקד בחשיבות ה-WAF, חומת אש ליישומי רשת הפועלת בשכבת יישומי הרשת, בשכבה הרבה יותר גבוהה ממערכות הגנה שונות. מה היא מערכת WAF, ולאיזו מטרה פותחה? שאלות אלו יהוו נקודת מיקוד למאמר זה. בנוסף, חשוב לדעת וכדאי להזכיר כי ניתן לכנות את הנושא גם: WIDS- (Web Instruction Detection System) והנושא יתקשר בחלקו למאמר הקודם שלי בנושא.

הצורך ב-WAF

כאמור ישומים אלו שממלאים כיום את אתרי האינטרנט, מביאים עמם שלל סכנות- הן לבעל האתר והן לגולשים באתר המשתפים בו מידע. כמה דוגמאות לסיכונים שיוצרים יישומי הרשת (מבוסס על רשימת ה-"Top Ten" של פרויקט OWASP, עליו אפרט בהמשך):

- Injction Flwas- מאפשרת לתוקפים לשתול קוד זדוני דרך ישום רשת למערכת אחרת, כמו מסד נתונים.
- Cross- Site Scripting- מאפשר לתוקפים לשתול סקריפטים זדוניים אל תוך אתרים דרך ישומי רשת.
- Insecure Cryptographic Storage- ישומי רשת שלא משתמשים בהצפנה ראוייה למידע רגיל כגון מספר כרטיס אשראי או מספר תעודת זהות, ובמקרה של פריצה- מידע המשתמש יהיה חשוף לפורץ.

סכנות אלו, ועוד רבות אחרות יוצרות צורך מתמיד לאבטח ולהגן על האתר ועל משתמשיו מפני פגיעות אפשריות, כמו גם להגן על המידע שמשתף בעל האתר והגולשים דרך האתר. אם כן, מדוע כל כך קשה לפתח ישום רשת מאובטח ויציב?

מרבית המפתחים לא מתמחים באבטחת מידע, ועל כן כתיבת ישום רשת מאובטח עשויה להיות מסובכת ומסורבלת עבורם, ואין הבטחה להצלחה בכך. סקירה, עיצוב ובדיקות החדירה אל הקוד מהווים תהליך ארוך ואיטי. כידוע לנו: "זמן=כסף", ולכן מתווסף רובד נוסף לעניין- הרובד הכלכלי. העלות גבוהה מאוד ביחס לזמן שמושקע בכך- במיוחד בבדיקות חדירה, בהם התהליך לא מכסה הכל. בכדי לענות על צרכים אלו, נכנסה טכנולוגיית ה-"WAF", טכנולוגיה חסכונית (ביחס לשאר הפתרונות) שמוכנה ליישום ומספקת אבטחה מיידית בזמן מהיר ועלות מינימלית.

ניתן להגדיר את ה-WAF במספר אופנים:

1. טכנולוגיית אבטחה שנועדה להגן על אתרים מפני התקפות ולא צריכה שינויים בקוד המקור של ישום הרשת.
2. תוסף שרת (Plug-In) או פילטר שמאפשר סט של חוקים על שיחות HTTP.

3. התקן מתווך שיושב בין הלקוח (הדפדפן) לבין השרת ומפענח הודעות משכבת היישומים, על מנת לגלות הפרות בתקנות האבטחה.

מרשימה זו ניתן להבין מיד כי כל אחד רואה את ה-WAF בצורה שונה ומגדיר אותו באופן מעט שונה, אך המכנה המשותף לכל ההגדרות הללו הוא התפקיד המרכזי של ה-WAF: **מניעת התקפות זדוניות דרך ישומי הרשת.**

קריטריונים להגדרת מערכת WAF

על מנת שנוכל להגדיר מהו בעצם ישום ה-WAF, ואילו תכונות ושיטות הוא מכיל בתוכו, הוקם פרוייקט מיוחד עבור מטרה זו שנקרא: WAFEC - Web Application Firewall Evaluation Criteria ובעברית- "הערכה לפי קריטריונים של חומת אש ליישומי רשת".

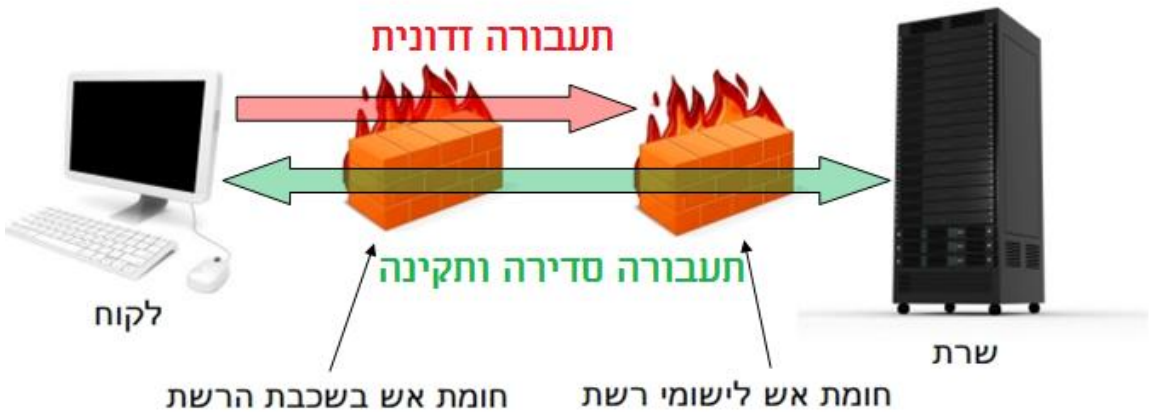
מטרת הפרוייקט היא לפתח סט של קריטריונים ל-WAF מתודולוגיה של בדיקות שיכולה להיות בשימוש על ידי כל טכנאי אחראי ולהקים פיתרון של WAF אמין ואיכותי. היעד אינו לתעד את המאפיינים שחייבים להיות בתוך כל WAF, משום שמערכת כזו היא מסובכת יתר על המידה, אלא לחבר מסמך ראוי ומסודר של מאפיינים פוטנציאלים ראויים לפרוייקט WAF כלשהו. בהמשך המאמר, אתאר מעט מהדברים שצויונו בגירסה מספר 1.0 של הפרוייקט הזה (שיצאה לאור בשנת 2006, העבודה על 2.0 עדיין נמשכת).

מטרות ה-WAF ואופן פעולתו

טכנולוגייה זו מתאימה במניעת התקפות שמערכות IDS וחומת אש בשכבת הרשת כמעט ולא מסוגלות להתמודד איתן. ניתן להסביר זאת על ידי דוגמה: חומת אש בשכבת הרשת פועלת על מנת לאפשר פורט 80, היא פועלת בשכבת התעבורה, בעוד ההתקפות נגד ישומי הרשת מבוצעות בשכבת היישומים. מכיוון שבימינו 70% מכלל ההתקפות משתמשות בשכבת יישומי הרשת, תאגידים וחברות זקוקים לכל העזרה שהם יכולים להשיג על מנת לאבטח את המערכות שלהן.

כפי שציטטתי למעלה- "Security is a not a product, but a process" , טכנולוגיית WAF לא נועדה להוות מוצר אחד כנגד כלל ההתקפות, אלא חומה חיצונית **נוספת**, מעגל נוסף בכלל מערך האבטחה של התאגיד. מכיוון שהיא ממוקמת בשכבה גבוהה (שכבה 7 במודל ה-OSI) לעומת שאר מערכות הגנה כפי שהזכרתי לעיל, היא מסוגלת למנוע את ההתקפות לפני שהן ימשיכו הלאה למחוזות קריטיים, באמצעות שימוש בחוקים מסויימים על מנת לאפשר/למנוע בקשות HTTP מסויימות כגון POST, GET וכדומה.
בנוסף, היא מספקת הגנה מפני מגוון רחב של התקפות על ישומי רשת ומאפשרת ניטור של תעבורת ה-HTTP וניתוח המידע בזמן אמת.

ברוב המקרים, ה-WAF ממוקמת בין השרת לבין הלקוח (הדפדפן) ומנטרת את התקשורת בינהם (ראו תרשים א'). בנוסף, היא מאפשרת גישה בזמן אמת למידע שניתן בשכבת היישומים, כדוגמת מידע אודות הפרמטרים שנשלחים.



אסטרטגיות זיהוי התקפות

התקפות ניתנות לזיהוי על ידי שימוש ב-2 אסטרטגיות עיקריות (ישנן אסטרטגיות נוספות שלא נפרט בחלק זה)- זיהוי על פי חוקים סטטים וזיהוי על פי חוקים דינאמים (חריגות).

חוקים סטטים

אסטרטגיית זיהוי על פי חוקים (Rule-Based) קובעת חוקים סטטים שנועדו להגדרה לפני הביצוע של תהליך הניתוח. אלו יכולים להיות חוקים פשוטים כמו זיהוי של תווים מסויימים, או חוקים מסובכים כמו קביעת חוקי שיחה (Session) קבועים. חוקים אלו נקבעים פעם אחת ונשארים באותה הצורה לכל אורך שלב הזיהוי. לכל חוק צריך להיות מבנה ספציפי משלו לכל ישום. חוקים אלו מעולים עבור מצבים ידועים מראש כגון תווי קלט, אורך הפרמטרים או סוגם וכדומה. חוקים אלו יכולים להיות מחולקים לשני מודולי זיהוי: **זיהוי חיובי** ו-**זיהוי שלילי**.

- מודל אבטחה חיובי / Positive Security Model: כאשר מודל זה מיושם, רק בקשות שידועות כנכונות מתקבלות, בעוד כל השאר פשוט נדחה. כלומר, תקנות ברירת המחדל הן שהכל נחסם מלבד מה שהוגדר כמותר, (ניתן לכנות גישה זו גם כ-White-List), מה שבתוך הרשימה הלבנה נחשב כתעבורה "נורמלית" שאין בה כוונות זדוניות. מלבד הגישה הידנית שבה צריך להגדיר כל פרט בנפרד, במודל זה ניתן להשתמש גם בשלב למידה אוטומטית - בו חשוב מאוד כי הוא יכיל רק תעבורת רשת טובה, משום שכל השאר יחשב זדוני. מודול זה עובד באופן הטוב ביותר עם

ישומים בשימוש נרחב, אך עם עידכונים מעטים. בנוסף, רוב של חומות האש בשוק בנויות היום בצורה כזו שבה כל שירות חדש צריך להירשם אל תוך הרשימה.

- מודל אבטחה שלילי / Negative Security Model: למודל זה יש ברב המקרים תקנות ברירת מחדל שמאשרות הכל- מה שמאפשר לכל בקשה לעבור הלאה, **מלבד מה שהוגדר כתעבורה זדונית** (גישה זו יכולה להחשב גם כ-Black List). התקנות מגדירות אילו בקשות **אינן** מאושרות ומה שבעצם מוגדר יסומן כהתקפה. המודל הזה נחשב קל יותר להטמעה, אך גישה זו לרב אינה יעילה במיוחד. החיסרון הגדול כמובן שהזיהוי יהיה טוב בדיוק כמו תקנות האבטחה שהוגדרו, והוא חייב לאמץ לעצמו תקנות חדשות כל הזמן ככל שהתקפות חדשות יהיו - כלומר לעדכן את עצמו כמה שיותר. כמובן שזה לא רק רע, נקודה חיובית בכך היא שאין יותר מדי התרעות שווא, היות והחוקים במערך מוגדרים לחפש **רק התקפות מוכרות** וכך בעצם תאפשר חסימה גדולה של התקפות אוטומטיות.

חוקים דינאמים (זיהוי על פי חריגות)

חוקי חריגות בנויים מחוקים דינאמיים, חוקים אלו לעומת החוקים הסטטיים אינם מוגדרים באופן ידני אלא דרך תהליך למידה. באותו שלב המערכת "מקליטה" ולומדת כל מה שנחשב "נורמלי" – זהו בעצם הדבר המשמעות ביותר. התעבורה הזו היא שתחשב תעבורה "סטירלית" - תעבורה נקייה מהתקפות (בדיוק כמו במאמר הקודם שבו דיברתי על הבסיס למערכת HIDS) גם פה, התעבורה הזו נחשבת ל-"תעבורת הבסיס". מטרת שלב הלמידה היא להגדיר מה נחשב "נורמלי"- מכיוון שלאחר שלב זה, ביישום שיטה זו התעבורה שתיקלט תשווה את עצמה למה שנלמד בשלב הלמידה ותחליט מה לא נראה "נורמלי" ותפעיל את האזעקה- .

אסטרטגיות הגנה

מכיוון שההתקפות מנסות לבצע פעולה זדונית, כך או אחרת יש בידינו 3 אטסטרגיות להגן על הישומים שלנו ועל כל מה שעלול להפגע דרכם.

יישום של WAF

דוגמה למערכת WAF על בסיס קוד פתוח - ModSecurity:

ModSecurity הינה חומת אש לישומי רשת מבוססת קוד פתוח (חשוב לציין כאן כי ModSecurity אינה פועלת בפני עצמה, אלה בעצם מודל של Apache. בנוסף לכך, היא גם מערכת לזיהוי חדירות (IDS) וגם מערכת למניעת חדירות (IPS)), ולעומת שאר הגרסאות המסחריות של WAF - שמציעות פיצ'רים רבים ועולות המון כסף, היא מציעה דווקא את הפיצ'רים שבאמת נחוצים וכל זה בחינם. מלבד 2 האסטרטגיות

הנפוצות שרשמתי למעלה בזיהוי התקפות, ModSecurity נותנת למשתמש את היכולת להשיג ולייישם בעצמו את מודל האבטחה שהוא בוחר. וכמו שכבר ציינתי, מלבד חוקים סטטים ישנם עוד מודולים נפוצים

כגון:

- **טלאי וירטואלי - VirtualPatching -** שפת חוקים אשר עושה את ModSecurity לכלי אידאלי לטלאים. מכיוון שברב האירגונים לוקח המון זמן (כמה שבועות) על מנת להטמיע טלאי שיתקן חולשה כזו או אחרת בישום, ב-ModSecurity ישומים קריטיים יכולים להיות מוטלאים מבחוץ מבלי הצורך לגעת/לשנות את קוד המקור של הישום (ואפילו ללא גישה אליו), דבר שעושה את המערכת מאובטחת עד שטלאי יציב יותר יצא לאור.

- **מודל הוצאת זיהוי - Extrusion Detection Model -** המערכת יכולה לנתר את המידע יוצא, לזהות ולחסום בעיות גלויות במידע (דוגמה: הודעות שגיאה מפורטות או מספרי כרטיס אשראי)

בלב ליבה של המערכת יש מנוע חוקים גמיש אשר נועד להיות קל לשימוש ומעוצב בפורמט פשוט ונוח. המערכת משתמשת בו על מנת לקבוע אילו חוקים יאכפו על ידה.

יישום ModSecurity

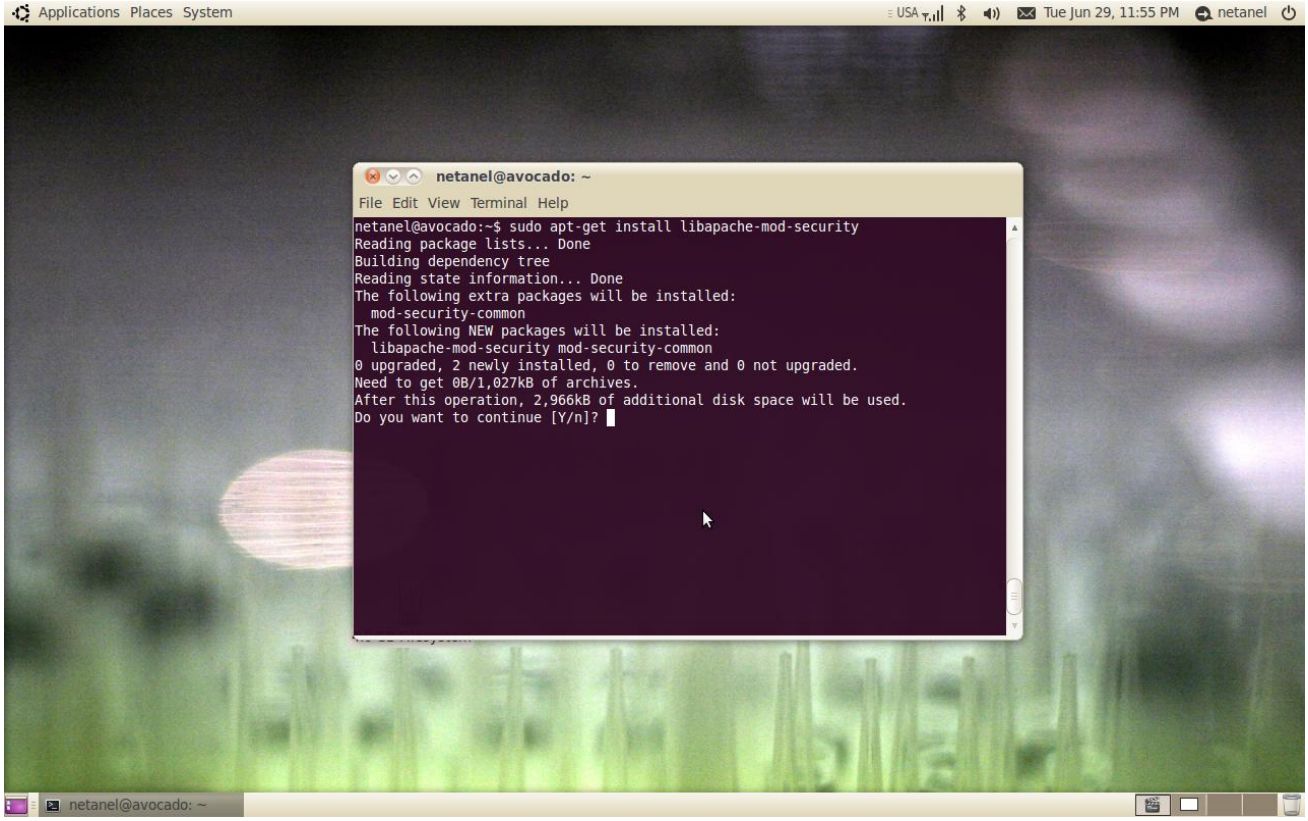
הורדה והתקנה

ניתן להוריד מהאתר הראשי של ModSecurity את הגירסה הנדרשת לכל מערכת ואפילו להדר אותה באופן עצמאי אם רוצים, אני אישית משתמש בהפצת אובונטו ולכן במאמר זה אראה כיצד ניתן להתקין את ModSecurity באובונטו.

על מנת להתקין את ModSecurity נצטרך לרשום במעטפת את הפקודה הבאה:

```
sudo apt-get install libapache-mod-security
```

לאחר מכן, המערכת תסביר כי היא דורשת גם את חבילת "mod-security-common" לביצוע השלמת פעולת ההתקנה שמכילה קבצי תיעוד וקבצי הגדרות לדוגמה, ולכן תתקין אותה בנוסף.



מכאן המערכת מותקנת, נראה כמה דוגמאות פשוטות לשימוש במנוע החוקים על מנת לפתור בעיות יום יומיות כגון מניעת התקפות מסוג SQL Injection:

```
SecFilter "DELETE[[:space]]+FROM"
```

או מניעת הזרקת קוד Javascript (כגון מתקפות Cross Site Scripting):

```
SecFilter "<script"
```

כמובן שניתן לעקוף את החוקים הללו בלא שום בעיות, חוקים אלו נועדו להדגמה בלבד, אך כאשר מדובר במערך חוקים שלם המתפקד כמקשה אחת- ביצוע פעולת המעקף היא הרבה יותר מורכבת ואף כמעט בלתי אפשרית. (כמעט..)

דוגמא נוספת של חיוב שימוש שמירת SESSION בעוגיות מאובטחות (HTTPOnly / Secure Flag), ניתן לראות בקישור הבא:
http://blog.modsecurity.org/2008/12/helping-protect-cookies-with-httponly-flag.html

פרוייקט מעניין שדרכו ניתן ללמוד רבות על יצירת חוקים ל-ModSecurity הוא " Securing WebGoat using ModSecurity" – אבטחת המערכת WebGoat של OWASP.

סיכום

קשה שלא להיתקל בימינו ביישומי אינטרנט כאלה או אחרים, בין אם אנחנו בעלי אתרים או סתם גולשים ברשת, ועל כן עלינו להיות מודעים לכלל הסכנות האורבות לנו ביישומים אלה וגם לפתרונות שקיימים לכך בשוק. אחד מגלגלי השיניים הפועלים במנגנון המורכב של ההגנה ברשת הינו רכיב ה-WAF שהוא מכלול של אמצעים הפועלים על פי עקרונות ההגנה על יישומי הרשת ומספקים לנו עוד שכבת הגנה ואבטחה. זוהי עוד מדרגה בדרך שלנו להפוך את האינטרנט (ואת העולם) למקום בטוח (ומאובטח) יותר.

על הכותב

נתנאל שיין עוסק בפיתוח ובאבטחת מידע בפרט, מעורב בפרויקטים שונים בנושא הקוד הפתוח בעיקר בהתנדבות, חבר בעמותת המקור, כיום עובד בהייטק וסטודנט למדעי המחשב באוניברסיטה הפתוחה.

קישורים חיצוניים לקריאה נוספת:

הבלוג של נתנאל שיין:

http://netshine.wordpress.com/

מחלק הדוקומנטציה באתר הבית של הפרוייקט:

http://www.modsecurity.org/documentation

הבלוג הרשמי של הפרוייקט:

http://blog.modsecurity.org

מאמר פרקטי ומעניין בנושא, נכתב ע"י Shreeraj Shah:

http://www.infosecwriters.com/text_resources/pdf/Defending-web-services.pdf

ה-TOC של הספר "ModSecurity Handbook":

https://www.feistyduck.com/books/modsecurity-handbook/ModSecurity_Handbook_1ed_TOC_and_Preface.pdf

מצגת מ-"WhatTheHack" מאת Daniel Fernández Bleda ו-Christian Martorella:

http://wiki.whatthehack.org/images/8/8c/Wth-slides-modsecurity.pdf

מצגות בנושא Web Intrusion Detection With ModSecurity מאת Ivan Ristic:

http://www.modsecurity.org/documentation/Web_Intrusion_Detection_with_ModSecurity.pdf

http://www.modsecurity.org/documentation/ApacheCon_Europe_2008-Web_Intrusion_Detection_with_ModSecurity.pdf

מרכז מידע בנושא Web Application Firewalls:

http://www.xiom.com/