مجله علمی- پژوهشی "پدافند الکترونیکی و سایبری"

سال اول، شماره ۴، زمستان ۱۳۹۲، ص ۷۶-۶۷

مدلی جدید مبتنی بر مارکوف جهت انتخاب بهترین راه‌کار امن‌سازی برنامه‌های کاربردی وب

سید جواد فتحی[1*]، تقی نوشی‌فرد[2]، مسعود باقری[3]، سعید پارسا[4]

۱ و ۲ – کارشناس ارشد کامپیوتر نرم‌افزار، دانشگاه جامع امام حسین (ع)

۳- دانشجوی دکترای نرم‌افزار، دانشگاه علم و صنعت ایران ٤- دانشیار، دانشکدهٔ مهندسی کامپیوتر، دانشگاه علم و صنعت ایران

(دریافت: ۹۱/۷/۵ ، پذیرش: ۹۱/۱۲/۲۱)

چکیده

در سال‌های اخیر با گسترش روز افزون فضای وب برای تبادل اطلاعات و ارائهٔ خدمات، تولید برنامه‌های کاربردی تحت وب، رشد قابل ملاحظه‌ای پیدا کرده است. مطابق گزارش‌های آماری در حال حاضر بیش از ۷۰ درصد حملات اینترنتی از طریق وب انجام می‌شود. بنابراین بخش عمده‌ای از تهدیدات ناشی از عدم تأمین امنیت کافی در برنامه‌های کاربردی و به‌طور خاص برنامه‌های کاربردی تحت وب است. برای مقابله با این تهدیدات راه‌کارهای متنوعی ارائه گردیده است. در این مقاله راه‌کارهای مختلف برای امن‌سازی برنامه‌های کاربردی وب مورد بررسی قرار گرفته و با توجه به معیارهای تدوین شده و مدل مارکوف پیشنهادی، روش جدیدی برای انتخاب راه‌کار مناسب برای امن‌سازی برنامه‌های کاربردی وب در دوره‌های مختلف چرخهٔ حیات سامانه، ارائه و مورد ارزیابی قرار گرفته است. روش پیشنهادی جواب‌گوی یکی از مهم‌ترین نیازهای کشور در حوزهٔ پدافند سایبری برای مدل کردن تهدیدات و نحوهٔ امن‌سازی برنامه‌های کاربردی وب می‌باشد. در روش پیشنهادی برای بیان توانمندی هر کدام از راه‌کارهای مطرح شده از طیف لیکرت به عنوان ارزیاب توانمندی استفاده شده است.

واژه‌های کلیدی: برنامه‌های کاربردی وب، امن‌سازی، بازبینی کد، دیوارهٔ آتش برنامه‌های کاربردی وب، سامانه تشخیص و جلوگیری از نفوذ، پویشگر امنیتی

۱. مقدمه

در سال‌های اخیر با گسترش روز افزون فضای وب به‌منظور تبادل اطلاعات و ارائهٔ خدمات، تولید برنامه‌های کاربردی تحت وب، رشد قابل ملاحظه‌ای پیدا کرده است. در این بین با وجود این که راه‌کارهای امنیت شبکه رشد خوبی داشته‌اند و محصولات متنوعی نظیر دیوارهٔ آتش، شبکهٔ خصوصی مجازی[1]، مدیریت یکپارچه تهدیدات[2] و غیره ارائه می‌شوند، موضوعات مرتبط با امنیت فضای وب به اندازهٔ کافی مورد توجه نبوده‌اند.

مطابق گزارش‌های آماری در حال حاضر بیش از ۷۰ درصد حملات اینترنتی از طریق وب انجام می‌شود [۱]. بنابراین بخش عمده‌ای از تهدیدات ناشی از عدم تأمین امنیت کافی در برنامه‌های کاربردی و به‌طور خاص برنامه‌های کاربردی تحت وب است.

اخیراً سازمان‌ها و مؤسسات مختلفی در زمینهٔ امنیت برنامه‌های کاربردی به‌صورت تخصصی، فعالیت می‌نمایند که بخشی از این فعالیت‌ها منجر به تولید پویشگرهای برنامه‌های کاربردی وب[3]، روش‌های برنامه‌نویسی امن[4] و دیوارهٔ آتش‌های برنامهٔ کاربردی وب[5] شده است.

یکی دیگر از فعالیت‌های این سازمان‌ها، دسته‌بندی و گروه‌بندی حملات یا آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب است که این امر، منجر به توسعهٔ دانش شده و کمک بسزایی در تولید محصولات امنیتی نموده است.

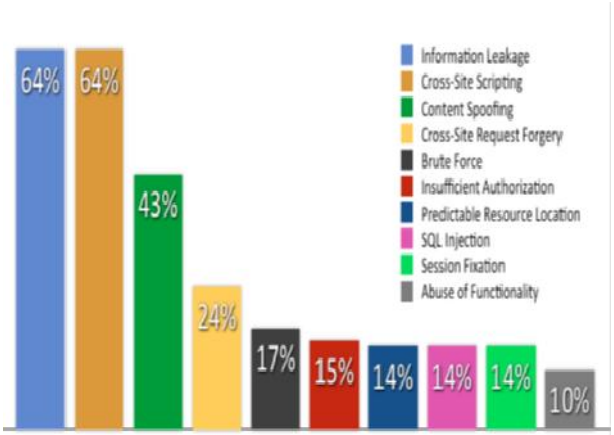
انجمن OWASP[6] و WASC[7] از جمله سازمان‌هایی هستند که در این زمینه، دسته‌بندی‌هایی را ارائه نموده‌اند.

1- Virtual Private Network
2- Unified Threat Management
3- Web Application Scanner

* رایانامه نویسنده پاسخگو: sjfathi@ihu.ac.ir

4- Secure Programming Methodology
5- Web Application Firewall
6- Open Web Application Security Project (www.owasp.org)
7- Web Application Security Consortium (www.webappsec.org)

شکل ۱ نمایشگر میزان وقوع این آسیب‌پذیری‌ها در برنامه های کاربردی طبق آمارهای ارائه‌شده در سال ۲۰۱۱ است [۲].



شکل ۱. آمار درصد وقوع ده آسیب‌پذیری مهم برنامه‌های کاربردی [۲]

۲. راه‌کارهای تأمین امنیت در لایهٔ کاربرد

۱.۲. سامانه‌های تشخیص و جلوگیری از نفوذ

یکی از تجهیزات امنیت شبکه که برای امن‌سازی برنامه های کاربردی وب در نظر گرفته می‌شود، سامانه تشخیص و جلوگیری از حملات است. این سامانه‌ها در دو نوع مبتنی بر میزبان و شبکه ارائه می‌شوند. برای امن‌سازی برنامه‌های کاربردی بر روی سرور می توان تنها سامانه‌های تشخیص و جلوگیری از نفوذ مبتنی بر شبکه[1] (NIPS) را مورد بحث قرار داد. این سامانه‌ها غالباً امکان تشخیص حملات را با کمک دو شیوهٔ تشخیص ناهنجاری[2] و تشخیص سوء استفاده[3] (مبتنی بر الگوی حملات) انجام می‌دهند [۳]. در واقع سامانه‌های NIPS با تمرکز بر تشخیص الگوهای تعریف شده ای از حملات فعالیت می‌کنند. این الگوها بیشتر حملات در سطح شبکه را پوشش می‌دهند. در حالی که در برنامه‌های کاربردی وبی تمرکز روی قرارداد HTTP بوده و باید حملات مخصوص این قرارداد را تشخیص داد. برای رسیدن به این هدف، NIPS باید کل ترافیک HTTP را موشکافی نموده و سپس برای تشخیص حملات بررسی کند. باتوجه به اینکه برخی از مهاجمان از جزئیات خاص در قرارداد HTTP مطلع هستند، به راحتی از سامانه‌های NIPS گذر می‌کنند. برای نمونه در حمله SQL Injection، مهاجم با کمک مفهوم Unicode در قرارداد HTTP می‌تواند از الگوهای NIPS فرار کند [۴]. سامانه‌های تشخیص و جلوگیری از نفوذ، مکانیزم‌های امنیت محیطی را فراهم می‌کنند و در ایجاد امنیت داخلی برای شبکه‌ها و همچنین امنیت نرم‌افزارهای وبی دارای چالش‌های عمده هستند. راه‌حل‌های سامانه تشخیص نفوذ نوعاً برای بازرسی بعد از وقوع حمله استفاده می‌شود و راه‌حل‌های سامانه جلوگیری از نفوذ، برای شناخت و انسداد ترافیک حمله تمرکز دارند. هر دو سامانه نقش حیاتی در جلوگیری از حملات خارجی دارند ولی حفاظت کاملی در مورد تهدیدات داخلی ندارند [۵]. این سامانه‌ها برای ایجاد امنیت داخلی دارای مسائل زیر می‌باشند:

سامانه‌های مذکور قابلیت درک منطق قراردادهای ارتباطاتی برنامه‌های وب، مانند HTTP, SOAP و غیره را ندارند و نمی‌توانند بین یک درخواست طبیعی و غیرطبیعی مربوط به برنامهٔ کاربردی وب در لایه شبکه تفاوتی قائل شوند [۶].

بسیاری از برنامه‌های کاربردی مبتنی بر وب از قرارداد SSL[4] استفاده می‌کنند. این قرارداد، امکانات امنیتی تشخیص هویت، جامعیت و محرمانگی را برای برنامه‌های تحت وب فراهم می کند. NIPSها عموماً هیچ‌گونه بازرسی برای ترافیک‌های رمزشده نظیر SSL را انجام نمی‌دهند. با توجه به این‌که یک محصول NIPS الگوهای حمله را روی ترافیک رمزشده نظیر ترافیک رمز نشده اعمال می‌کند، بنابراین می‌توان نتیجه گرفت که این محصول نمی‌تواند هیچ حملهٔ وابسته به محتوای لایهٔ کاربرد را در قرارداد SSL تشخیص دهد. این واقعیت به دلیل رمزنگاری محتوای لایهٔ کاربرد در قرارداد SSL است [۷].

بنابراین این سامانه‌ها در تحلیل قرارداد HTTP مشابه قراردادهای شبکه‌ای، نمی‌توانند به‌طور مناسب داده‌های وب را در لایهٔ کاربرد، تجزیه و تحلیل کنند و زیرمؤلفه های آن را مانند سرآیندهای درخواست، کوکی ها، نام متغیرها و محتوای اجرایی صفحات بشناسند. سامانه‌های جلوگیری از نفوذ با داده‌های HTTP معمولاً به‌صورت یک متن بزرگ برخورد کرده، که باعث افزایش نرخ اخطارهای نادرست مثبت[5] و منفی[6] می‌شود [۶]. اصولاً سامانه‌های تشخیص و جلوگیری از نفوذ سامانه‌های امنیتی بر پایه نشانه هستند، بنابراین محدودیت حفاظت در مقابل حملات ناشناخته[7] را دارند. متأسفانه بیشتر نشانه‌های مورد استفاده در سامانه‌های جلوگیری از نفوذ، بر مبنای آسیب‌پذیری‌های عمومی نرم‌افزارها می‌باشند که برای برنامه‌های کاربردی وب مؤثر نیستند [۶].

بیشتر سامانه‌های جلوگیری از نفوذ قابلیت‌های نظارت ترافیک شبکه را داشته و تا حدودی می‌توانند انحراف‌هایی را تشخیص دهند

1- Network Intrusion Protection System (NIPS)
2- Anomaly Detection
3- Misuse Detection

4- Secure Socket Layer
5- False Positive
6- False Negative
7- Zero Day Attack

ولی به اندازه کافی قابلیت کاربرد و نظارت هر برنامهٔ مستقل را ندارند. حملات برنامه‌های وب مانند جلوگیری از سرویس، جستجوی کامل و حملات تکه تکه شده قرارداد HTTP برای هر سایتی، آستانه‌ای (حد و مرز) دارد. هر وقت که یک برنامه از سطح آستانه‌اش تجاوز کرد، سامانه امنیتی باید درخواست‌هایش را متوقف کند. NIPSها از انجام این عمل عاجزند [۶].

سامانه‌های تشخیص و جلوگیری از نفوذ، بیشتر روی داده های ورودی و کمتر روی داده‌های خروجی از برنامه‌های وب تمرکز دارند. مهاجمین اغلب از پیغام‌های خطای صادر شده، برای تنظیم حملات خود به منابع بانک‌های اطلاعاتی استفاده می‌کنند [۷].

۲.۲. پویشگرهای برنامه‌های کاربردی وب[1]

از جمله ابزارهایی که در چرخهٔ امن‌سازی برنامه‌های کاربردی وب تاثیرگذارند، پویشگرهای برنامه‌های کاربردی وب هستند که در مراحل تست قبل از تحویل برنامه یا آزمون‌های بعد از اعمال تغییرات، از آنها استفاده می‌شود. این پویشگرها ابزارهای اتوماتیکی هستند که برنامه‌های کاربردی وب را با هدف کشف آسیب‌پذیری‌های عمومی نظیر تزریق اسکریپت‌های مخرب[2]، تزریق دستورات SQL[3]، پیمایش دایرکتوری‌ها[4]، تنظیمات غیر امن و اجرای دستورات از راه دور[5] مورد آزمون قرار می‌دهند [۱].

این ابزارها توسط دو دسته از افراد استفاده می‌شوند. دستهٔ اول، متخصصان امنیت برنامه های وب هستند که برای کشف آسیب‌پذیری‌ها و اصلاح آنها از این ابزارها استفاده می‌کنند و دستهٔ دوم نفوذگرانی هستند که به منظور سوء استفاده، از این ابزارها برای کشف آسیب‌پذیری‌ها بهره می‌برند.

پویشگرها به دو دسته تجاری و متن‌باز تقسیم می‌شوند و استفاده از آنها بخش مهمی از ارزیابی امنیتی برنامه‌های کاربردی وب را تشکیل می‌دهد. همچنین به‌عنوان تکمیل کننده بخشی از استانداردهای امنیتی، نظیر بخش ۶.۶ از استاندارد امنیتی دادهٔ صنایع کارت‌های پرداخت الکترونیکی[6]، مورد استفاده قرار می‌گیرند [۹]. پویشگرهای برنامه‌های کاربردی وب فقط آسیب‌پذیری‌هایی را شناسایی می‌کنند که قبلاً با نشانه‌های[7] آن آشنا شده و روش کشف آن را آموزش‌دیده باشند. در غیر این‌صورت قادر به شناسایی آسیب‌پذیری نیستند، از طرف دیگر این ابزارها به فناوری ساخت برنامه‌های وب وابسته نیستند و آنها را با وجود انواع فناوری‌های توسعه مورد ارزیابی قرار می‌دهند [۱۰].

پویشگرهای برنامه‌های کاربردی وب به هر دو صورت جعبه سیاه[8] و جعبه سفید[9] قادر به انجام ارزیابی هستند [۱۱]. منظور از حالت جعبه سفید این است که بعضی از پویشگرهای برنامه های کاربردی وب در صورت دسترسی به کد برنامه و اضافه کردن کد ارائه شده توسط شرکت به برنامهٔ کاربردی، برنامه را با دقت بالاتری مورد ارزیابی قرار میدهد [۱۲].

در سند [۸] ارزیابی توانایی‌های پویشگرهای وب برای شناسایی آسیب‌پذیری‌های برنامه‌های کاربردی وب ارائه شده است. همچنین سند NIST ۲۶۹-۵۰۰ ویژگی‌های حداقلی برای یک پویشگر برنامه‌های کاربردی وب را بیان کرده است [۹].

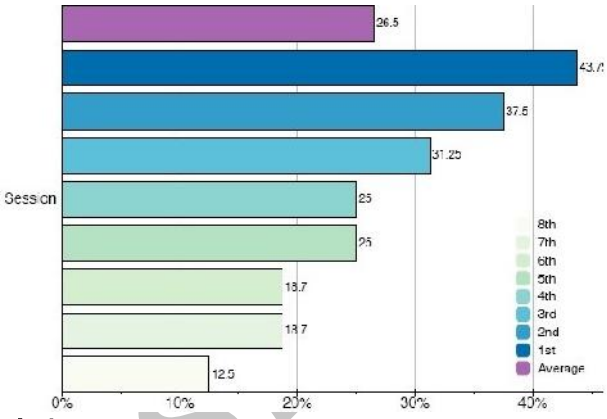
تحقیقات [۱۰ و ۱۱] نشان می‌دهد که پویشگرها همیشه قادر به کشف آسیب‌پذیری‌های موجود در برنامه‌های کاربردی وب نیستند.

همچنین [۱۳] بیان می‌کند که تعداد آسیب‌پذیری‌هایی که پویشگرها کشف می‌کنند کم بوده و دارای خطای بالایی است. در سال ۲۰۰۷ تحقیقاتی در مورد توانایی پویشگرهای وب برای شناسایی آسیب‌پذیری‌ها انجام شد که نتیجهٔ آن نشان داد بین ۲۰ تا ۷۷ درصد شناسایی‌های آنها اشتباه است و همچنین ۹ درصد آسیب‌پذیری‌ها نیز شناسایی نشده‌اند [۱۴].

توانایی پویشگرها در شناسایی آسیب‌پذیری‌های وب‌سرویس ها نیز توسط [۱۵] مورد آزمایش قرار گرفته است که نتیجهٔ آن نشان‌دهندهٔ درصد کم شناسایی آسیب‌پذیری‌ها و میزان خطای بالای پویشگرها است. تحقیقاتی نیز روی دستهٔ خاصی از حملات انجام شده است که نشان می‌دهد، پویشگرها در شناسایی حملات Stored XSS، تزریق‌های پیشرفته SQL و حملات علیه مکانیزم‌های مدیریت جلسه دچار ضعف‌های شدیدی[10] هستند [۱۶].

1- Web Application Security Scanner
2- XSS
3- Standard Query Language
4- Directory Traversal
5- Remote Command Execution
6- PCIDSS
7- Signature
8- Black Box
9- White Box
10- Session

نمونه‌ای از نتایج [۱۶] را در شکل ۲ مشاهده می‌کنید که در بهترین حالت ۴۳/۷ درصد آسیب‌پذیری‌های مربوط به مدیریت نشست را شناسایی کرده است.



شکل ۲.درصد شناسایی مشکلات امنیتی مربوط به مدیریت جلسه [۱۶]

پویشگرهای برنامه‌های کاربردی وب از نظر نحوهٔ به‌کارگیری به دو صورت مورد استفاده قرار می‌گیرند. حالت اول نشانه گیری و شلیک[1]، حالت دوم آموزش داده‌شده[2] [۱۰]. در حالت اول استفاده‌کننده نیاز به هیچ تنظیمی روی ابزار ندارد و در حقیقت پویشگر با همان تنظیمات اولیه اجرا می‌شود. با توجه به نتایج تحقیقات [۱۰]، می‌توان نتیجه گرفت که در این حالت به‌طور میانگین ۵۵ درصد آسیب‌پذیری‌ها توسط پویشگرها شناسایی نمی‌شوند. در حالت دوم استفاده‌کننده باید به‌طور میانگین ۱ تا ۱/۵ ساعت وقت صرف تنظیمات پویشگر برای انجام آزمون برنامهٔ کاربردی وب نماید. البته این زمان شامل زمان صرف شده برای خواندن مستندات ابزار و همچنین میزان تبحر استفاده‌کننده نسبت به ابزار نیز می‌باشد. در این حالت نزدیک به ۴۸ درصد آسیب‌پذیری‌ها توسط این ابزارها شناسایی نمی‌شوند.

باتوجه به این مطالب، برای سازمان‌های بزرگ به‌صرفه نیست که این ابزارها را با این سطح کارآیی فقط به‌منظور یک بار تست یا بعد از هر تغییری در برنامه وب استفاده کنند. علاوه بر این به‌دلیل میزان بالای خطا در گزارش‌های تولید شدهٔ این ابزارها، و عدم توانایی تفکیک گزارش‌های صحیح از غلط، زمان زیادی از متخصصین امنیت برنامه‌های کاربردی وب برای تحلیل این گزارشات هدر می رود. تحقیقات [۱۰] نشان می‌دهد که این زمان به‌طور میانگین معادل ۲۰ ساعت است. نکته جالبی که از روش انجام تحقیق [۱۰] به‌دست می‌آید این است که تولیدکنندگان پویشگرهای تجاری ابزارهای خود را روی سایت‌های پیشنهاد شده برای تست، از قبل برنامه‌ریزی کرده‌اند. برای اثبات این موضوع می‌توان با تست این ابزارها بر سایت‌های [۱۷و۱۸] که از آن‌ها به‌منظور آزمایش پویشگرها و همچنین آموزش کاربران در مورد امنیت برنامه‌های کاربردی وب استفاده می‌شود، به این نتیجه رسید [۱۹].

۳.۲. کدنویسی امن و بازبینی کد برنامه‌های کاربردی[3]

بسیاری از آسیب‌پذیری‌های موجود در برنامه‌های کاربردی، ناشی از عدم دقت در تولید این برنامه‌هاست. ملاحظات امنیتی در تمام مراحل شیوه‌های تولید نرم‌افزار ارائه شده است. با این حال تولید امن برنامه‌های کاربردی هنوز هم یک چالش اساسی در امنیت نرم افزار است. علاوه بر ارائهٔ رهیافت‌های امنیتی در شیوه‌های تولید، به طور خاص بازبینی کد و ارائه رهیافت در نحوهٔ کدنویسی امن، راه کار مناسبی برای بهبود وضعیت نرم‌افزار تولید شده می‌باشد.

گرچه راه‌کارهای کدنویسی امن و بازبینی کد در راستای امنیت برنامهٔ کاربردی بسیار مناسب می‌باشند، در مقایسه با راه‌کارهای دیگر هنوز نمی‌توانند جایگزین کاملی باشند [۲۰]. در زیر دلایل آن ذکر شده است.

به‌طور معمول در فرآیند تولید نرم‌افزار، راه‌کارهای کدنویسی امن و بازبینی آن کمتر مورد توجه مدیران تولید قرار می‌گیرند. برای نمونه با وجود تأخیر و نزدیکی به زمان‌های حساس در تولید، امنیت به سادگی قربانی خواهد شد.

گرچه ممکن است در فرآیند تولید اولین نسخهٔ برنامهٔ کاربردی، ملاحظات امنیتی در تولید لحاظ شود، در زمان حل مشکلات برنامه و ارائه نسخ جدید که ملاحظات زمان و هزینه مطرح می‌شود، باید منتظر نقص کدنویسی امن بود. به عنوان مثال، طبق گزارش تهدیدات اینترنت شرکت سیمانتک [۲۱] میانگین زمان لازم برای اعمال وصلهٔ[4] امنیتی برای سازمان‌ها ۵۵ روز می‌باشد. در حالی‌که در گزارش آماری امنیت وب گروه امنیتی Whitehat زمان لازم برای اصلاح آسیب‌پذیری تزریق کد SQL کشف شده در برنامه های کاربردی وب ۱۳۸ روز می‌باشد [۲].

در بسیاری از موارد برنامه‌های کاربردی از طریق برون سپاری تولید شده است و در زمان کشف یک آسیب‌پذیری نمی توان کد برنامه را تغییر داد. بنابراین باید راه‌کار امن‌سازی مستقل از تغییر کد را ارائه نمود. باید توجه کرد که حجم زیادی از برنامه‌های کاربردی در

1- Point and Shoot
2- Trained

3- Secure Coding
4- Patch

Overall Summary			Acunetix		Appscan		Burp Suite Pro		Haistorm		NTOSpider		Qualys	Web Inspect	
			PaS	Trained	PaS	Trained	PaS	Trained	PaS	Trained	PaS	Trained	PaS	PaS	Trained
Grand Totals		Vuln Found	۶۴	۷۳	۸۴	۸۵	۳۲	۵۶	۶۱	۹۶	۱۴۲	۱۴۶	۴۴	۴۶	۵۲
		Found٪	٪۴۱.۵۶	٪۴۷.۴۰	٪۵۴.۵۵	٪۵۵.۱۹	٪۲۰.۷۸	٪۳۶.۳۶	٪۳۹.۶۱	٪۶۲.۳۴	٪۹۲.۲۱	٪۹۴.۱۶	٪۲۸.۵۷	٪۲۹.۸۷	٪۳۳.۷۷
Valid Vulns	۱۵۴	Vulns Missed	۹۰	۸۱	۷۰	۶۹	۱۲۲	۹۸	۹۳	۵۸	۱۲	۹	۱۱۰	۱۰۸	۱۰۲
Pages	۳۱۵	Missed٪	٪۵۸.۴۴	٪۵۲.۶۰	٪۴۵.۴۵	٪۴۴.۸۱	٪۷۹.۲۲	٪۶۳.۶۴	٪۶۰.۳۹	٪۳۷.۶۶	٪۷.۷۹	٪۵.۸۴	٪۷۱.۴۳	٪۷۰.۱۳	٪۶۶.۲۳
		FP Reported	۳	۳	۵	۳	۲	۶	۹	۷	۳	۳	۲	۳	۲
		Scan Time	۸:۳۳	۱۰:۴۴	۶:۵۴	۶:۱۸	۰:۴۲	۱:۴۹	۲:۳۱	۹:۲۸	۸:۰۳	۷:۴۵	۱:۲۸	۲:۵۳	۴:۱۸
		Training Time	N/A	۱:۱۰	N/A	۱:۳۰	N/A	۲:۰۵	N/A	۴:۱۰	N/A	۰:۰۵	N/A	N/A	۱:۲۵
		Total Time	۸:۳۳	۱۱:۵۴	۶:۵۴	۷:۴۸	۰:۴۲	۳:۵۴	۲:۳۱	۱۳:۳۸	۸:۰۳	۷:۵۰	۱:۲۸	۲:۵۳	۵:۴۳

شکل ۳. نتایج حاصل از بررسی نمونه‌های مختلف از پویشگرهای برنامه‌های کاربردی وب [۳]

سازمان‌ها با برون‌سپاری[1] تولید می‌شود و هزینهٔ تغییر کد بعد از تولید نسخ اولیه، منوط به تمدید قرارداد تولیدکننده و حتی انتخاب پیمان‌کار دیگر برای درک کد برنامه و رفع مشکل می‌باشد [۲۰].

بسیاری از سازمان‌ها چندین نوع از خدمات خود را به صورت برنامه‌های کاربردی تحت وب ارائه می کنند. با کشف یک آسیب‌پذیری روی برنامه‌های کاربردی، اصلاح یک به یک این برنامه‌ها هزینهٔ بالایی خواهد داشت.

۴.۲. دیواره‌های آتش برنامهٔ کاربردی وب[2]

در حیطه ابزارهای امنیت اطلاعات، محصولات دیوارهٔ آتش را می‌توان یکی از عمومی‌ترین تکنولوژی‌ها در راستای امن‌سازی نام برد. البته این محصولات توان امنیتی خود را بیشتر در زمینه امن‌سازی شبکه نشان داده‌اند. ولی واقعیت این است که دیواره‌های آتش تنها در مباحث امنیت شبکه مورد استفاده قرار نمی‌گیرند. تکنولوژی دیواره‌های آتش از زمان اولین نسل آن‌ها، یعنی حدود سال ۱۹۹۵ میلادی تاکنون، سیر توسعه و تحول ویژه‌ای را طی نموده است. این سیر توسعه در دو زمینه کلی رشد قابلیت‌های امنیتی در محصولات دیوارهٔ آتش و افزایش انواع دیواره‌های آتش با توجه به کاربردها و حیطه استفاده از آن‌ها، قابل دسته‌بندی است. محصولات دیوارهٔ آتش از نظر حیطه کاربرد شامل دیوارهٔ آتش شبکه[3]، دیوارهٔ آتش شخصی[4]، دیوارهٔ آتش برنامهٔ کاربردی وب و دیوارهٔ آتش پایگاه‌داده[5] می‌باشد. همان‌طور که از نام این محصولات مشخص است، هر کدام در حیطهٔ خاصی کاربرد دارند. بدیهی است که دیواره‌های آتش شبکه‌ای، نیاز به دیوارهٔ آتش شخصی یا بقیه دیواره‌های آتش را مرتفع نمی‌سازد. البته از تمام انواع دیواره های آتش، نوع شبکه‌ای آن از محبوبیت بالایی در امن‌سازی برخوردار است. تکنولوژی این دیواره‌های آتش با حضور محصولات UTM، امکان بازرسی محتوای لایهٔ کاربرد را برای انواع ویروس‌ها و Spamها فراهم می‌کند. با این وجود دیواره‌های آتش شبکه‌ای درک کاملی از برنامه‌های کاربرد موجود در شبکه‌ها نخواهند داشت. برای نمونه زمانی که در یک سازمان برنامه اتوماسیون اداری استفاده می شود، دیواره آتش شبکه‌ای درک کاملی از این برنامه ندارد و تنها می‌تواند نوع سرویس، آدرس‌های شبکه و آدرس‌های سرویس گیرنده را بازرسی نماید. در محصولات UTM نیز تنها بازرسی ویروس و Spam به این قابلیت‌ها اضافه می‌شوند. بنابراین باید از دیواره‌های آتشی استفاده نمود که برنامه‌های کاربردی را از این تهدیدات محافظت نماید. بدیهی است برای بسیاری از این تهدیدات، ابزار امنیتی باید بتواند بیش از گزینه‌های قرارداد HTTP، برنامه کاربردی را درک و بازرسی نماید.

دیواره‌های آتش برنامه‌های کاربردی وب، گونهٔ جدیدی از فناوری‌های نوظهور در عرصهٔ امنیت اطلاعات و ارتباطات است که به‌منظور محافظت ویژه از برنامه‌های تحت وب در مقابل حملات مختلف و رو به افزایش در این حوزه، ایجاد شده است. سامانه‌های WAF قادر به متوقف کردن حملاتی هستند که دیواره های آتش شبکه و سامانه‌های تشخیص نفوذ مرسوم و معمول، در حوزهٔ امنیت از دفاع در مقابل آن‌ها عاجزند. این سامانه‌ها به‌گونه ای ساخته می‌شوند که به ویرایش کدهای منبع نرم‌افزارهای تحت وبی که این سامانه از آن‌ها محافظت می‌کند نیازی نداشته باشد [۲۰]. این دیواره‌های آتش به‌طور خاص بازرسی ترافیک برنامه‌های کاربردی مبتنی بر وب را انجام می‌دهند. در این بازرسی علاوه بر اطلاعات شبکه‌ای ترافیک، اطلاعات خاص برنامه‌های کاربردی نیز مورد نیاز

1- Outsource
2- Web Application Firewall
3- Network Firewall
4- Personal Firewall
5- Database Firewall

می‌باشد. برای نمونه محصولات WAF کنترل ویژه‌ای روی مفهوم نشست[1] انجام می‌دهند. باید توجه داشت که این مفهوم کاملاً مربوط به برنامهٔ کاربردی است و به‌عبارت دیگر بالاتر از قراردادهای لایهٔ کاربرد است.

به‌طور معمول یک محصول WAF یا به‌صورت نرم افزاری و یکپارچه در داخل کارگزار وب[2] یا به صورت یک Appliance و به‌عنوان یک ابزار شبکه‌ای می‌تواند مستقر شود. معمولا محصولات تجاری در بازار به‌صورت یک Appliance ارائه می شوند. این محصولات می‌توانند مستقل از نوع کارگزار وب، خدمات WAF را ارائه کنند. همچنین با حضور چندین نوع برنامهٔ کاربردی می‌توان با استفاده از یک WAF به‌صورت Appliance از تمام برنامه های کاربردی موجود محافظت نمود. برخی ملاحظات مربوط به استفاده از WAF عبارتند از:

محصولات WAF تمرکز روی قرارداد HTTP داشته و حملات مخصوص این قرارداد را تشخیص می‌دهند. برای رسیدن به این هدف، WAF کل ترافیک HTTP را موشکافی نموده و سپس آن را برای تشخیص حملات بررسی می‌کند.

محصول WAF به‌طور کامل، ترافیک رمز شدهٔ SSL را بازرسی می‌کند. برای این منظور این محصولات به عنوان یک نمایندهٔ[3] قرارداد SSL در شبکه قرار گرفته و ترافیک را رمزگشایی، بازرسی و سپس رمزنگاری می‌کنند.

گرچه در مفاهیم شبکه‌ای بالاترین لایه در پشتهٔ قراردادهای شبکه‌ای[4]، لایهٔ کاربرد[5] است، در زمانی که مفاهیم برنامه‌نویسی شبکه مطرح می‌شود، و به‌طور خاص در برنامه‌نویسی وب، لایه‌ای بالاتر از لایهٔ کاربرد نیز وجود دارد. البته این لایه وابسته به نوع برنامهٔ کاربردی وب متفاوت است. برای نمونه مفاهیمی نظیر مدیریت نشست، Web Service، اعتبارسنجی ورودی[6] و ... همگی بالاتر از سطح قرارداد HTTP قرار می‌گیرند. بخش زیادی از قابلیت‌های صافی کردن در WAF مربوط به بازرسی در لایه‌ای بالاتر از لایهٔ کاربرد است. برای نمونه WAF می‌تواند مدیریت نشست را در یک برنامهٔ کاربردی بررسی نماید. برخی از قابلیت‌های امنیتی محصول WAF که خاص مفاهیم لایهٔ بالاتر کاربرد می‌باشد، شامل کاهش حملات جستجوی کامل[7]، حفاظت از کوکی‌ها[8]، کاهش حملات مبتنی بر نشست، حفاظت فیلدهای مخفی در فرم‌ها و غیره می باشد. محصول WAF علاوه بر کشف آسیب‌پذیری، امکان ممانعت از تهدید را نیز دارد. یک محصول WAF می‌تواند چندین برنامهٔ کاربردی را به‌طور مداوم حفاظت نماید.

با توجه به گزارش ارائه شده در [۲۲] دیواره‌های آتش برنامهٔ کاربردی وب بدون مکانیزم آموزش درصد پایینی از حملات را شناسایی و جلوگیری می‌کنند و موارد شناسایی شده توسط آنها دارای میزانی از خطا است. بنابراین برای بهبود عملکرد دیواره آتش برنامهٔ کاربردی وب در مقابل آسیب‌پذیری‌ها بهتر است این سامانه در ترکیب با یک پویشگر آسیب‌پذیری برنامه‌های کاربردی وب، به‌منظور تولید قوانین بازدارنده استفاده شود [۲۳].

۳. روش پیشنهادی

برای بیان توانمندی هر کدام از راه‌کارهای مطرح شده از مکانیزم طیف لیکرت به‌عنوان ارزیاب توانمندی استفاده شده است. جدول شماره ۱ بیانگر نگاشت یک معیار عددی به معیار کمی در طیف لیکرت می‌باشد[۲۴و۲۵]:

جدول ۱. طیف لیکرت

معیار	وزن
کم	۱
نسبتا کم	۳
متوسط	۵
زیاد	۷
خیلی زیاد	۹

سعی شده است با استفاده از این طیف، مقادیر کیفی به دست آمده در جدول ۲ را به یک مقدار کمی نگاشت کرد. در مدل مارکوف پیشنهادی از این مقادیر استفاده می‌گردد.

با توجه به مطالب فوق، معیارهای ارزیابی لازم در جهت استفاده از راه‌کارهای امنیتی بیان شده در مقاله را در دوره‌های مختلف عمر یک نرم‌افزار در جدول ۲ ارائه می‌دهیم. وزن هر کدام از معیارهای انتخابی، با بررسی‌ها و آزمایش‌های انجام شده و مقایسه نتایج آن ها به‌دست آمده است [۲۰ و۲۲]. با توجه به اینکه توانمندی هر کدام از معیارها می‌تواند در یک بازه نوسان داشته باشد، از طیف لیکرت برای بیان توانمندی و مقایسه استفاده گردیده است. در مواردی که معیار انتخاب شده در راه‌کار مورد نظر امکان اجرا نداشته یا با شرایطی امکان چشم‌پوشی را دارد، از مقدار کم برای بیان اهمیت معیار مورد

1- Session
2- Embeded
3- Proxy
4- Network Stack Protocol
6- Input Validation
7- Brute Force
8- Cookies

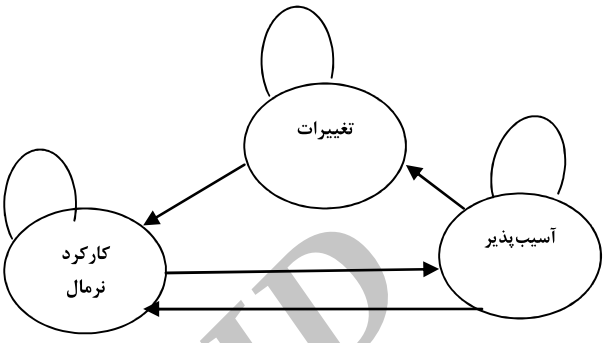
جدول ۲. معیارهای مربوط به ارزیابی راه‌کارهای امن‌سازی برنامه‌های کاربردی وب

ردیف	معیارهای ارزیابی		بازبینی کد	پویشگر	NIDS/IPS	WAF
۱	زمان مورد استفاده	توانمندی کشف آسیب‌پذیری در زمان تغییر کد	خیلی زیاد	متوسط	کم	زیاد
		واکنش سریع به حمله	نسبتاً کم	نسبتاً کم	متوسط	خیلی زیاد
۲	مهارت‌های مورد نیاز	آشنایی با زبان‌های برنامه‌نویسی	خیلی زیاد	نسبتاً کم	کم	متوسط
		آشنایی با حملات	خیلی زیاد	متوسط	زیاد	خیلی زیاد
		آشنایی با پروتکل‌های مربوط	زیاد	متوسط	زیاد	خیلی زیاد
۳	زمان و هزینه	نصب و راه‌اندازی	کم	متوسط	زیاد	زیاد
		آموزش	خیلی زیاد	کم	متوسط	متوسط
		تنظیمات دوره‌ای	کم	نسبتاً کم	متوسط	متوسط
		سرعت تست نرم‌افزار	متوسط	زیاد	کم	کم
۴	مزایا	تشخیص حملات فعال	کم	کم	نسبتاً کم	خیلی زیاد
		جلوگیری از حملات فعال	کم	کم	زیاد	خیلی زیاد
		Zero Day Detection	خیلی زیاد	کم	کم	زیاد
		بررسی کانال رمزشده	کم	کم	کم	خیلی زیاد
		اعمال سیاست‌های مستقیم امنیتی	کم	کم	متوسط	زیاد
		محافظت هم‌زمان از چندین برنامهٔ‌کاربردی وب	متوسط	متوسط	زیاد	خیلی زیاد
		در دسترس نبودن کد (جعبه سیاه)	کم	متوسط	متوسط	خیلی زیاد
		تشخیص‌حملات منطقی	خیلی زیاد	کم	کم	کم
		قابلیت همبستگی با سایر ابزارهای امنیتی	کم	متوسط	زیاد	زیاد
		قابلیت یادگیری/ تجربه‌نگاری	زیاد	کم	کم	متوسط
۵	معایب	False Positive	کم	متوسط	زیاد	نسبتاً کم
		False Negative	کم	متوسط	زیاد	نسبتاً کم
		انسان خبره در دسترس	نسبتاً کم	زیاد	متوسط	متوسط
		سربار در بلندمدت	کم	نسبتاً کم	متوسط	زیاد
		عدم توانایی در پوشش نقصان‌های معماری نرم‌افزار	کم	خیلی زیاد	خیلی زیاد	خیلی زیاد

نظر استفاده می‌شود. به عنوان مثال زمان و هزینه مورد نیاز نصب و راه‌اندازی در روش بازبینی کد، قابل چشم‌پوشی بوده ولی با توجه به انتخاب طیف لیکرت به عنوان معیار ارزیابی می‌توان مقدار کم را به آن نسبت داد.

برای انتخاب راه‌کار مناسب در شرایط مختلف از مدل مارکوف استفاده می‌کنیم. برای به‌دست آوردن مدل مارکوف، فرض شده که برنامهٔ کاربردی وب در حال استفاده است. بدین منظور که مدل ارائه شده، برای انتخاب راه‌کار مناسب در زمان استفاده

برنامهٔ کاربردی است و شرایط تولید برنامهٔ کاربردی مد نظر نمی‌باشد. برای یک برنامهٔ کاربردی در زمان استفاده، حالات کارکرد نرمال، آسیب‌پذیر و تغییرات در برنامه، مطابق شکل متصور می‌باشد.



شکل ۴. مربوط به حالات مختلف یک برنامهٔ کاربردی در زمان استفاده

همان‌طور که در شکل۴ نمایش داده شده است، وقتی یک برنامهٔ کاربردی که در حالت کارکرد نرمال است با گزارش یک آسیب‌پذیری به حالت آسیب‌پذیر می‌رود، در حالت آسیب‌پذیر در صورتی که کد برنامهٔ کاربردی برای رفع آسیب‌پذیری در دسترس باشد برنامه به حالت تغییرات منتقل شده و پس از رفع آسیب‌پذیری مجددا به حالت کارکرد نرمال باز می‌گردد.

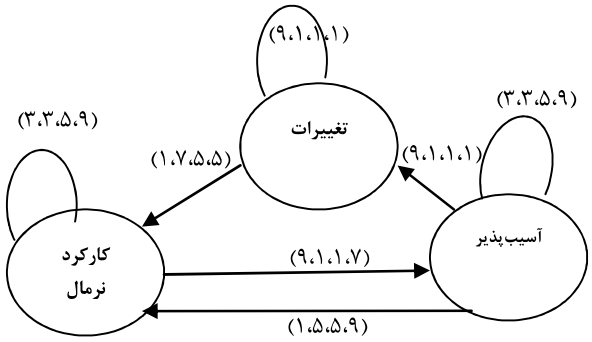
در صورت در دسترس نبودن کد برنامه، باید از راه کارهای امن‌سازی که نیاز به کد برنامه ندارند استفاده کرد و با رفع آسیب‌پذیری، برنامه را دوباره به حالت کارکرد نرمال بازگرداند.

حال با استفاده از پارامترهای مطرح شده در جدول ۳ و با استفاده از شکل ۴، مدل مارکوف را مربوط به انتخاب راه‌کار مناسب در شرایط مختلف چرخهٔ استفاده یک برنامهٔ کاربردی ارائه می‌دهیم (شکل ۵).

جدول ۳. نحوهٔ محاسبهٔ امتیاز مربوط به هر کدام از روش‌ها

SOLUTION1	WAF+SC=9+3=12
SOLUTION2	IDS+SC=5+3=8

در مدل ارائه شده چهارتایی، (CA,SC,IDS,WAF) به ترتیب بیانگر امتیاز کسب شده استفاده از بازبینی کد، پویشگر، سامانه تشخیص و جلوگیری از نفوذ و دیوارهٔ آتش برنامهٔ کاربردی برای امن‌سازی در طیف لیکرت می‌باشد. مقادیر ارائه شده برای هرکدام از چهارتایی‌ها از نگاشت وزن‌های ارائه شده در جدول ۲، با معیارهای مطرح طیف لیکرت در جدول ۱ به‌دست آمده است.



شکل ۵. مدل مارکوف؛ انتخاب راه‌کار مناسب امن‌سازی برنامهٔ کاربردی وب

باتوجه به مدل ارائه شده می‌توان از روی چهارتایی مربوط به زمان استفاده یعنی (۳،۳،۵،۹) نتیجه گرفت که ترکیب استفاده از پویشگر با دیوارهٔ آتش برنامهٔ کاربردی وب یا ترکیب پویشگر با سامانه تشخیص و جلوگیری نفوذ به‌ترتیب می توانند راه‌کارهای مناسبی برای امن‌سازی برنامهٔ کاربردی وب باشند.

همچنین می‌توان این نتیجه را گرفت که در صورت وقوع آسیب‌پذیری با شرایط در دسترس نبودن کد برنامه، راه‌کار مناسب استفاده از دیوارهٔ آتش برنامهٔ کاربردی وب می‌باشد و در شرایطی که کد برنامه در دسترس باشد بازبینی کد به عنوان راه کار مناسب می‌تواند انتخاب شود. همان‌طور که در مدل مشاهده می شود، در شرایط مختلف و با توجه به معیارهای معایب، مزایا و هزینهٔ زمان می‌توان گزینه‌های دیگر را با اولویت خاص انتخاب کرد.

آشنایی با زبان‌های برنامه‌نویسی تقریبا برای تمامی روش های مطرح لازم می‌باشد ولی در صورت استفاده از روش‌هایی که نیاز بالایی به این پارامتر دارند، باید محدودیت‌ها و معایب مربوط به آن روش مانند کمبود نیروی انسانی متخصص را مد نظر داشت.

زمان و هزینه مورد نیاز برای هر کدام از راه‌کارهای مطرح، از دو نگاه مورد بررسی قرار گرفته است. اولی مربوط به هزینه و زمان مورد نیاز برای نصب و راه‌اندازی و آموزش در جهت استفادهٔ بهینه از راه‌کار و دومی مربوط به هزینه و زمان مورد نیاز برای به‌روزرسانی هر کدام از راه‌کارهای مطرح که در قالب تنظیمات دوره‌ای آورده شده است.

در بخش مزایا توانایی راه‌کارهای مختلف در شناسایی حملات ناشناخته، محافظت از چندین برنامه در آن واحد، میزان همبستگی با

1. Brute Force
2. Cookies

ابزارهای امنیتی دیگر (به عنوان مثال دیواره‌های آتش برنامهٔ کاربردی وب برای محافظت از بارگذاری فایل‌های آلوده در سرور از آنتی‌ویروس‌ها کمک می‌گیرند)، توانایی یادگیری و میزان وابستگی به در دسترس بودن کد برنامهٔ مد نظر قرار گرفته است. بخش معایب از دو منظر کلی عملکرد(False Negative ,False Positive) سربار و پوشش نقصان‌های معماری نرم‌افزار) و هزینه (انسان خبره) انتخاب شده‌اند.

۴. ارزیابی روش پیشنهادی

برای ارزیابی روش پیشنهادی از ابزارهای متن باز Mode Security، W3af و Snort استفاده گردیده است. برای این منظور ابزارهای مطرح در معماری شکل ۶ استقرار یافتند.

پس از استقرار معماری، حملات مطرح در حوزهٔ برنامه های کاربردی وب بر روی ۳۱۵ وب سایت مورد آزمون قرار گرفت. نتایج حاصل از این ارزیابی در جدول ۴ ارائه گردیده است. همان‌طور که مشاهده می‌شود محصولات دیوارهٔآتش و سامانه‌های تشخیص نفوذ، به تنهایی در مقابل حملات تا حدی موفق بوده‌اند. در صورتی که ترکیب این محصولات با پویشگرهای آسیب‌پذیری درصد موفقیت چشمگیری را نشان می‌دهد. به عنوان مثال دیوارهٔ آتش Mode Security به تنهایی از بین ۱۱۹ سایت آسیب‌پذیر توانست ۴۴ سایت را به درستی آسیب‌پذیر تشخیص دهد. در صورتی که در ترکیب با پویشگر آسیب‌پذیری W3af این تعداد به ۹۰ سایت افزایش یافت.

جدول ۴. نتایج حاصل از ارزیابی روش پیشنهادی

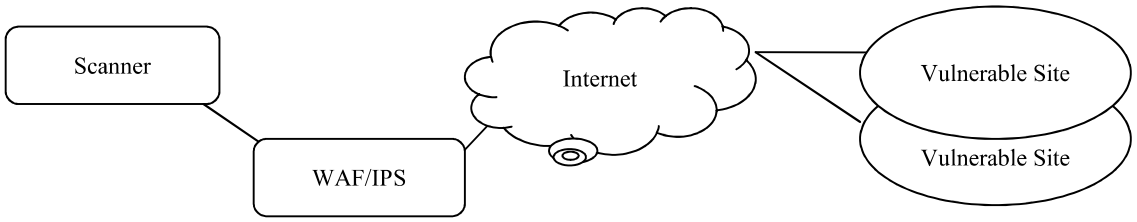
		WAF		IDS	
		Mode Security	W3af	Snort	W3af
تعداد کل سایتهای مورد ارزیابی ۳۱۵	تشخیص صحیح	۴۴	۹۰	۳۱	۸۶
	درصد تشخیص صحیح	٪۳۶.۹۷	٪۷۵.۶۳	٪۲۶.۰۵	٪۷۲.۲۶
تعداد سایت‌های آسیب‌پذیر ۱۱۹	تشخیص نادرست	۷۵	۲۹	۸۸	۳۳
	درصد تشخیص نادرست	٪۶۳.۰۲	٪۲۴.۳۶	٪۷۳.۹۵	٪۲۷.۷۳

پس می‌توان نتیجه گرفت نتایج به‌دست آمده از ارزیابی منطبق با مدل مارکوف پیشنهادی بوده و جمع مقادیر مربوط به چهارتایی‌ها در شرایط مختلف میزان کارا بودن آن راه‌کار را نشان می‌دهد.

۵. نتیجه‌گیری

در این مقاله ابتدا راه‌کارهای مختلف امن‌سازی برنامه های کاربردی وب مطرح و در شرایط مختلف، چرخهٔ حیات سامانه با هم مقایسه گردیدند. سپس با توجه به معیارهای تدوین شده و مدل مارکوف پیشنهادی نحوهٔ انتخاب راه‌کار مناسب برای امن سازی برنامه‌های کاربردی ارائه گردید و مورد ارزیابی قرار گرفت. نتایج حاصله نشان داد که در زمان استفاده از برنامهٔ کاربردی وب برای انجام واکنش سریع به حمله، مهارت‌های آشنایی با زبان‌های برنامه نویسی، آشنایی با حملات و آشنایی با پروتکل‌های مربوطه، راه کار مناسبی برای تأمین امنیت برنامهٔ کاربردی، استفاده از دیوارهٔ آتش برنامهٔ کاربردی وب با ترکیب پویشگرهای آسیب‌پذیری می‌باشد. در صورت استفاده از این گزینه حملات Zero Day و حملات فعال با درصد احتمال بالایی قابل شناسایی می‌باشند. این روش به عنوان یکی از راه‌کارهای مناسب برای امن‌سازی برنامه‌های کاربردی وب که کد آنها در دسترس نمی‌باشد مطرح است. معایب مهم این روش عدم توانایی پوشش نقصان‌های مربوط به معماری برنامه‌های کاربردی وب است. در صورت کشف آسیب‌پذیری در دورهٔ استفاده از نرم‌افزار نیز با توجه به پارامترهای مزایا، معایب، هزینه و زمان بیان شده در جدول ارزیابی، دیوارهٔآتش برنامهٔ کاربردی وب گزینه مناسب‌تری می‌باشد.

روش پیشنهادی جواب‌گوی یکی از مهم‌ترین نیازهای کشور در حوزهٔ پدافند سایبری برای مدل کردن تهدیدات و نحوهٔ امن سازی برنامه‌های کاربردی وب می‌باشد.



شکل ۶. معماری استقرار ابزارها برای ارزیابی روش پیشنهادی

۵. مراجع

[1] Web Application Security Consortium., WEB APPLICATION SECURITY STATISTICS, http://www. Web apps ec.org/. s.l. : (WASC), 2011.

[2] WhiteHat Security., 11th Edition Website Security Statistic Report. s.l. : WhiteHat Security, Inc., 2011.

[3] Scarfone. K, Mell. P, Guide to Intrusion Detection and Prevention Systems (IDPS). s.l: Computer Security Resource Center, 800-94, 2007.

[4] Kruegel. Ch, Valeur. F, Vigna. G, INTRUSION DETECTION AND CORRELATION Challenges and Solutions. s.l: Springer Science + Business Media, Inc, 2005.

[5] What is the Difference Between IPS, IDS and Internal Security? checkpoint. [Online] Check Point Software Technologies, http://www.checkpoint.com/securitycafe/readingroom/internal_security/ ips_ids_ internal_ security. html, 2011.

[6] McMillan. J, Intrusion Detection FAQ: What is the difference between an IPS and a Web Application Firewall? sans.org. [Online] http://www.sans.org/security-resources/idfaq/ips-web-app-firewall.php, November 2009.

[7] OWASP German Chapter., Best Practices:Use of Web Application Firewalls. s.l. : Open Web Application Security Project(OWASP), 2011.

[8] Web application Security Consortium,Web Application Security Scanner Evaluation Criteria,Version 1.0, 2009

[9] http://samate.nist.gov

[10] Suto. L, Analyzing the accuracy and time costs of web application security scanners, February 2011.

[11] Fong. E, Gaucher. R, Okun. V, Black. P. E, and Dalci. E, Building a test suite for web application scanners. Hawaii International Conference on System Sciences, 2008

[12] Antunes. N, Vieira. M, Detecting SQL Injection Vulnerabilities in Web Services. In Dependable Computing, 2009. LADC '09. Fourth Latin-American Symposium on, pages 17–24, Sept. 2009.

[13] Fonseca. J, Vieira. M, Madeira. H, Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks. In Dependable Computing, 2007.PRDC 2007. 13th Pacific Rim International Symposium on, pages 365–372, Dec. 2007

[14] Vieira. M, Antunes. N, Madeira. H, Using web security scanners to detect vulnerabilitiesin web services. In Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIPInternational Conference on, pages 566–571, 29 2009-July 2 2009.

[15] http://www.owasp.org

[16] State of the Art: Automated Black-Box Web Application Vulnerability Testing, Jason Bau, Elie Bursztein, Divij Gupta, John Mitchell ,Stanford University Stanford, CA,2009

[17] D. Rhoades. WebMaven. Available at: http://www.mavensecurity .com/WebMaven.php, 2002

[18] McAfee, Inc. Hacme Bank v2.0. Available at: http://www.foundstone.com/us/resources/proddesc/hacmebank.htm, May 2006.

[19] Using a Web Server Test Bed to Analyze the Limitations of Web Application Vulnerability Scanners, David A. Shelly, Blacksburg, Virginia ,2010

[20] Barnett. R, "WAF Virtual Patching Challenge: Securing WebGoat with ModSecurity."s.l: Blackhat DC Conference, 2009.

[21] Internet Threat Report, H3: Symantec, 2007.

[22] Suto. L, "Analyzing the Effectiveness of Web Application Firewalls." 2011.

[23] Lemos. R, Time To Automate Web Defenses?, Http://www.darkreading.com/vulnerability-management/167901026/security/attacks-breaches/231901651/time-to-automate-web-defenses.html. Oct 25, 2011.

[24] Likert. R, "A Technique for the Measurement of Attitudes," Archives of Psychology, p. 140:1–55, 1932.

[25] Wuensch. K. L, "What is a Likert Scale? and How Do You Pronounce 'Likert?," East Carolina University, 2009.

[26] Bryant. Ch, The Key Difference Between IPS And IDS. Streetdirectory. [Online] 2011.http://www.Streetdirectory.com/travel_guide/149387/networking/the_key_difference _ between_ips_and_ids. html.

[27] Palka. D, Zachara. M, "Learning Web Application Firewall - Benefits and Caveats." 2011.[28] A. Robert, R. C. Barnett and etc., Web Application Firewall Evaluation Criteria. s.l.: Web Application Security Consortium, 2009.

[28] Barnett. R, "XSS Street-Fight:The Only Rule Is There Are No Rules." s.l. : Blackhat Dc Conference, 2010.