فصلنامه علمی-ترویجی پدافند غیرعامل

سال سوم، شماره ۳، پاییز ۱۳۹۱، (پیاپی ۱۱): صص ۴۱-۵۰

تحمل‌پذیری نفوذ، رویکردی نوین برای توسعه سیستم‌های نرم‌افزاری

صادق بجانی[۱]، محمد عبداللهی ازگمی[۲]

تاریخ دریافت: ۹۱/۰۸/۱۴
تاریخ پذیرش: ۹۱/۱۰/۰۴

چکیده

امروزه شاهد حضور و تأثیر فراوان سیستم‌های نرم‌افزاری در سرعت، دقت و حجم پردازش‌ها هستیم. هرگونه نفوذ در سیستم‌های نرم‌افزاری، خسارات مؤثر و برخی اوقات آثار جبران‌ناپذیری را به دنبال دارد. روش‌های متعارف امنیتی به تنهایی توان مقابله با نفوذ به سیستم‌های نرم‌افزاری را ایجاد نمی‌کنند و همچنین وجود آسیب‌پذیری‌های انکارناپذیر و ناشناخته در فرآیند توسعه نرم‌افزار، آنها را در مقابله با نفوذ، از به‌کارگیری راهکارهای مؤثر بی‌نیاز نمی‌کند. در این شرایط، تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری به عنوان یک راه‌حل مقابله با نفوذ پذیرفتنی است. برای تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری در کنار روش‌های متعارف امنیتی، از تکنیک‌های مهم تحمل‌پذیری خطا استفاده می‌شود که نتیجه آن، فراهم کردن استمرار ارائه خدمات پیش‌بینی‌شده سیستم نرم‌افزاری، حتی در شرایط نفوذ است.

تعیین مؤلفه‌های مؤثر در تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری و میزان تأثیر هریک از مؤلفه‌ها، گامی مؤثر در تولید سیستم‌های نرم‌افزاری است که به صورت مشخص در اختیار طراحان نیست. در این مقاله با بهره‌گیری از دانش معماری سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ، از جمله SITAR, MAFTIA, SCIT، مؤلفه‌ها و عوامل مؤثر در تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری تعیین و ارائه می‌شوند تا زمینه‌های لازم جهت توسعه سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ فراهم گردد.

کلیدواژه‌ها: نفوذ، تحمل‌پذیری نفوذ، سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ و روش‌های متعارف امنیتی

۱- دانشجوی دکتری مهندسی کامپیوتر، دانشگاه جامع امام حسین(ع) sbejani@ihu.ac.ir- نویسنده مسئول

۲- استادیار و عضو هیئت دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران

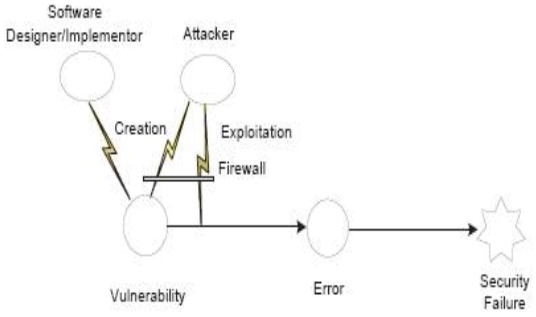
مقدمه

در سیستم‌های نرم‌افزاری، مشکلات در سه سطح خطا[1]، اشکال[2] و خرابی[3] قابل دسته‌بندی است. خطا عیب فیزیکی، کاستی[4] یا نقص است که در برخی از مؤلفه‌های سیستم اتفاق می‌افتد [۱]. از خطاهای رایج مولفه‌های نرم‌افزاری می‌توان اشکال‌های نرم‌افزاری را نام برد. خطاها در سطح فیزیکی سیستم رخ می‌دهند. اشکال، انحراف از درستی یا دقت در محاسبه است که در نتیجه یک خطا به‌وجود می‌آید و معمولاً سبب ایجاد حالت نادرست در حالات سیستم می‌شود. این مشکل در سطح اجرای سیستم نرم‌افزاری و سطح محاسباتی سیستم مطرح است [۲]. خرابی به معنی اجرا نشدن برخی از عملیات سیستم در زمان مقرر و برابر انتظار است [۲]. یک سیستم موقعی دچار خرابی می‌شود که عملکرد آن در یک بازه زمانی، از رفتار پیش‌بینی‌شده خود، تفاوت داشته باشد و حتی اگر عملکرد درست سیستم در خارج از زمان مقرر انجام گیرد، خرابی در آن سیستم رخ داده است. خرابی در سطح کل سیستم رخ می‌دهد. نفوذ[5]، حاصل یک حمله[6] موفق عمدی و بدخواهانه است که با استفاده از آسیب‌پذیری‌های سیستم ایجاد می‌شود [۲]. به عبارت دیگر، نفوذ، تحریک خارجی عمدی بدخواهانه یا خطای عملیاتی است که باعث وقوع حالت نادرست در سیستم می‌شود. نفوذ به صورت عمدی و تناوبی با هدف بهره‌برداری از نقاط آسیب‌پذیری سیستم رخ می‌دهد. با توجه به شکل (۱) می‌توان پذیرفت که هرنفوذ دارای دو عامل اصلی آسیب‌پذیری و حمله است [۳]. حمله در واقع یک خطای عمدی و بدخواهانه است که به‌دنبال بهره‌برداری از آسیب‌پذیری سیستم می‌باشد.

امروزه اکثر فعالیت‌های علمی، اقتصادی، تجاری، فرهنگی و سایر فعالیت‌ها با استفاده از سیستم‌های نرم‌افزاری انجام می‌گیرند. بنابراین سرعت، دقت و ارتباط پردازش‌های آنها غیرقابل انکار است و لازم است امنیت آنها تأمین شود. برای این منظور از روش‌های متعارف امنیتی و فنون تحمل‌پذیری خطا استفاده می‌شود. به چند دلیل از جمله دلایل زیر، امنیت سیستم‌های نرم‌افزاری به‌طور کامل تأمین نمی‌شود: ۱- مهارت و دانش نفوذگران، که همواره پیشاپیش متخصصین امنیت حرکت می‌کنند و سیستم‌های نرم‌افزاری مورد هجوم آنها قرار می‌گیرند؛ ۲- وجود آسیب‌پذیری‌های انکارناپذیر در محصولات نرم‌افزاری؛ ۳- عدم امکان حذف کامل آسیب‌پذیری‌های سیستم‌های نرم‌افزاری. این شرایط، زمینه لازم برای حمله مهاجمان را فراهم می‌کند؛ یعنی به‌کارگیری روش‌های متعارف امنیتی، تأمین کننده امنیت سیستم‌های نرم‌افزاری نمی‌باشند و لازم است راه حلی برای این مشکل پیدا شود. یک راه برای پوشش ضعف شیوه‌های مقابله با نفوذ، ایجاد زمینه لازم برای تحمل نفوذ در سیستم نرم‌افزاری است که در آن از تکنیک‌های تحمل‌پذیری خطا در کنار شیوه‌های متعارف امنیتی استفاده می‌شود. با استفاده از این کار، نرم‌افزار قابلیتی پیدا می‌کند که حتی در صورت رخداد نفوذ موفق در آن، قادر به استمرار ارائه خدمات خود باشد که این توانایی، همان تحمل‌پذیری نفوذ است.

این تحقیق به‌دنبال شناسایی وتعیین مؤلفه‌های مؤثر در تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری، تأثیر هرکدام در تأمین تحمل‌پذیری نفوذ، کاربرد و شرایط استفاده از هر مؤلفه می‌باشد. محصول این کار به عنوان راهکاری مناسب برای طراحان سیستم‌های نرم‌افزاری قابل استفاده خواهد بود.

Attack + Vulnerability → Intrusion → Error → Failure



شکل ۱- مراحل تحقق نفوذ و ایجاد خرابی در سیستم[۳]

۲- بیان مسئله و ضرورت تحقیق

سیستم‌های نرم‌افزاری باید در مقابل حملات مهاجمان مقاوم باشند و جنبه‌های مقاومت باید در فرآیند تولید، به آنها اضافه شود. علاوه بر آن، تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری نیز یک حقیقت انکارناپذیر است که بدون آن، سیستم‌های نرم‌افزاری دیر یا زود دچار حملات بدخواهانه مهاجمان قرار گرفته و ضررهای جبران‌ناپذیری ایجاد می‌شود. یک سوال مهم این است که طراحان سیستم‌های نرم‌افزاری با کدام تکنیک‌ها و ابزارها سیستم‌های نرم‌افزاری را تحمل‌پذیر نفوذ کنند؟ در این مقاله برای یافتن پاسخ مناسب، معماری‌های MAFTIA, SITAR و SCIT را که به‌منظور تولید سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ ارائه شده‌اند، مورد بررسی دقیق قرار داده و در آن معماری‌ها، مؤلفه‌های مؤثر در تحمل‌پذیری نفوذ و تأثیر هر یک در تأمین تحمل‌پذیری نفوذ، کاربرد و شرایط استفاده از هر مؤلفه، شناسایی و ارائه می‌شود.

سؤال اصلی تحقیق این است که چگونه می‌توان سیستم‌های

1- Fault
2- Error
3- Failure
4- Defect
5- Intrusion
6- Attack

نرم‌افزاری تحمل‌پذیر نفوذ تولید کرد؟ و برای رسیدن به پاسخ این سؤال، لازم است صفات اصلی سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ تعیین گردد و نیز مؤلفه‌هایی از معماری سیستم نرم‌افزاری تحمل‌پذیر نفوذ که در تحمل‌پذیری نفوذ سیستم مؤثر می‌باشند و اینکه هر مؤلفه مؤثر کدام ویژگی را تأمین می‌کنند و این کار را چگونه انجام می‌دهند، تعیین گردند. متغیر مستقل تحقیق، تحمل‌پذیری نفوذ و متغیر وابسته آن، معماری‌های تحمل‌پذیر نفوذ سیستم‌های نرم‌افزاری می‌باشد. این تحقیق برای رسیدن به پاسخ سؤال اصلی، به دنبال شناخت مؤلفه‌های مؤثر در معماری سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ و تبیین تأثیر استفاده از آنهاست.

در ادامه مقاله، راهکارهای مقابله با نفوذ، معرفی معماری سیستم‌های تحمل‌پذیر نفوذ، راهکار پیشنهادی برای توسعه سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ، معرفی یک نمونه کاربردی و در پایان، نتیجه‌گیری و پیشنهادها ارائه می‌شوند.

۳- راهکارهای مقابله با نفوذ

چهار راهکار مقابله با نفوذ در سیستم‌های متعارف نرم‌افزاری عبارت‌اند از [۵]:

۳-۱- پیشگیری از نفوذ[1] در مقابل خطاهای بدخواهانه یک توانمندی تدافعی است [۸]. استفاده از دیواره‌های آتش[2] و نرم‌افزارهای ضد ویروس و سایر تجهیزات دفاعی برای متوقف‌کردن سوءاستفاده از کلاس‌های شناخته‌شده آسیب‌پذیری، نمونه‌هایی از روش‌های پیشگیری از نفوذ محسوب می‌شوند[۱۰]. پیشگیری کامل از وقوع نفوذ غیرممکن است زیرا مدیریت تمام حمله‌ها به‌دلیل ناشناخته‌بودن حملات جدید یا ناشناخته بودن چگونگی شکل‌گیری حملات جدید، عدم امکان حذف کامل نقاط آسیب‌پذیر و جلوگیری از به‌وجود آمدن آنها عملاً غیرممکن است [۵].

۳-۲- تشخیص نفوذ[3] یک توانمندی تدافعی است که شامل مکانیزم‌هایی برای کشف نفوذ و ارسال پیام‌های هشدار می‌باشد [۵]. مکانیزم‌های یادشده، مبتنی بر امضاء یا مبتنی بر تشخیص ناهنجاری‌ها می‌باشند.

۳-۳- **پوشش نفوذ**[4] قابلیتی در نرم‌افزار است که علی‌رغم وجود نفوذ در سیستم، اجازه نمی‌دهد نتیجه نفوذ در خروجی سیستم اثرگذار باشد[۲]. برای دستیابی به این قابلیت، از فنون قطعه‌بندی، افزونگی و پخش استفاده می‌شود تا در صورت نفوذ به بخشی از سیستم، تنها امکان دسترسی به داده‌های غیر مهم فراهم شود. چون با استفاده از این فنون، داده‌ها به‌گونه‌ای تقسیم می‌شوند که هیچ بخشی از آنها به‌تنهایی شامل اطلاعات مهم نباشد و قرار دادن هر یک از بخش‌های داده در مکان‌های مختلف، دسترسی نفوذگر به مجموعه اطلاعات معنی‌دار را نیازمند به مصالحه[5] در آوردن تعداد بیشتری از گره‌ها می‌نماید.

۳-۴- تحمل‌پذیری نفوذ[6]، طی مراحل مختلف تولید سیستم، سعی بر این است که از ایجاد آسیب‌پذیری‌های جدید جلوگیری شود و آسیب‌پذیری‌های شناخته‌شده رفع یا مسدود شوند اما علی‌رغم همه این اقدامات، آسیب‌پذیری‌هایی همچنان باقی می‌مانند. آنها زمینه‌ساز بروز نفوذ هستند که نتیجه تحریک خارجی عمدی بدخواهانه یا خطای عملیاتی است[۵]. تحمل‌پذیری نفوذ، نوعی توانمندی محسوب می‌شود که بر خلاف روش‌های پیشگیری و تشخیص نفوذ، می‌تواند به صورت بالقوه برای مقابله با کلاس‌های ناشناخته آسیب‌پذیری مورد استفاده قرار گیرد [۹]. این روش به‌جای تشخیص و حدس حملات مشخص، با بررسی تغییرات رفتار سیستم نسبت به رفتار قابل انتظار عمل می‌کند و در شرایطی که حملات ناشناخته قبلی و حملات جدید سیستم را تهدید می‌کند، کارساز است.

۴- ویژگی سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ

طراحی سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ، به‌گونه‌ای است که صدمات ناشی از نفوذ، تحمل شده و در صورت امکان، سیستم به‌طور خودکار ترمیم[7] و بازیابی می‌شود. در صورت رخداد نفوذ موفق در سیستم‌های تحمل‌پذیر نفوذ، علاوه بر غیرفعال نشدن کامل آنها و از دست نرفتن تمام سرویس‌ها تا زمان تعمیر، تداوم فعالیت سیستم تحت نفوذ مهم است[۶] و به آن اولویت داده می‌شود. براساس[۵] سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ، پنج ویژگی زیر را توأماً دارا می‌باشند:

- **عدم مداخله مؤلفه‌های[8] سیستم در کارکرد یکدیگر:** برای تأمین این ویژگی نیاز است عملیات مؤلفه‌های سیستم، مستقل از سایر مؤلفه‌های سیستم باشد تا آسیب‌پذیری‌های آنها متفاوت باشند و در صورت نفوذ در یکی از مؤلفه‌های سیستم، راه برای نفوذ به سایر مؤلفه‌های سیستم باز نشود. وجود این ویژگی سبب محدودسازی نفوذ می‌شود.
- **حفظ جامعیت[9] داده‌ها:** از اهداف بدخواهانه حملات به سیستم‌های نرم‌افزاری، نقض جامعیت داده‌های سیستم است. این ویژگی سبب می‌شود با استفاده از مکانیزم‌های آزمون‌های پذیرش، نتایج پردازش سیستم نرم‌افزاری بررسی و اصلاح موارد

1- Intrusion Prevention
2- Fire Walls
3- Intrusion Detection
4- Intrusion Masking

5- Compromise
6- Intrusion Tolerance
7- Recovery
8- Componentes
9- Integrity

نقض جامعیت داده‌ها ممکن گردد.

- **محدودسازی نفوذ**[1]: وقتی در سیستم نرم‌افزاری، نفوذی، تشخیص داده شده و اعلام گردد، به‌منظور محدودسازی نفوذ و عدم گسترش آن، ابتدا محل رخداد نفوذ با استفاده از فنون خاصی تعیین شده و ارتباط آن محدوده با سایر نواحی سیستم قطع می‌گردد.
- **پایداری**[2]: پس از رخداد نفوذ، با استفاده از شناسایی حملات منع سرویس روی الگوریتم‌های پیکربندی مجدد و خنثی‌سازی آنها، پایداری سیستم تأمین می‌گردد.
- **بازیابی‌نفوذ**[3]: بازیابی داده‌ها و مسدودسازی آسیب‌پذیری‌های منجر به نفوذ، در سیستم‌های تحمل‌پذیر نفوذ انجام می‌گیرد. با این کار، سیستم نرم‌افزاری قابلیتی پیدا می‌کند که علی‌رغم مورد نفوذ قرار گرفتن، توانایی خودترمیمی و پوشش نفوذ را داشته و به‌راحتی قادر به استمرار ارائه سرویس‌های خود می‌شوند. در واقع با استفاده از فنون خاص و روش‌های امنیتی، بقاپذیری سیستم نرم‌افزاری تقویت می‌شود [۹].

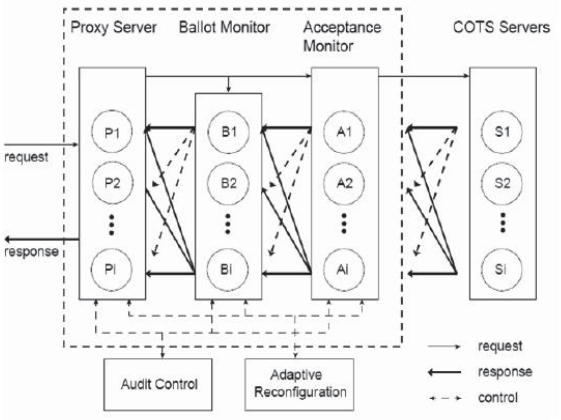
۵- معرفی چند معماری سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ

۵-۱- معماری SITAR[4]

این معماری برای ایجاد سیستم‌های تحمل‌پذیر نفوذ در سرویس‌های توزیع‌شده، یک چارچوب محسوب می‌شود [۴]. بر این معماری جهت ایجاد کلاس خاصی از سرویس‌های توزیع شده در شبکه تأکید می‌شود و همچنین از فنون تحمل‌پذیری خطا، از جمله افزونگی و تنوع طراحی و نحوه مدیریت حملات خارجی و مولفه‌های مصالحه‌ای استفاده می‌شود. مؤلفه مصالحه‌ای ممکن است هر رفتار دلخواه و غیرقابل پیش‌بینی شده داشته باشد و دارای پیچیدگی خاصی است. شکل (۲) نمای کلی معماری SITAR را نشان می‌دهد. بخش تحمل‌پذیری نفوذ معماری، درون کادر نقطه‌چین نشان داده شده است.

در این معماری فرض اصلی این است که سرویس‌دهنده‌های مؤلفه‌های تجاری، در مقابل نفوذ آسیب‌پذیر هستند و قابلیت‌های این معماری است که آنها را در مقابل نفوذ، تحمل‌پذیر می‌نماید.

سرویس‌دهنده‌های نماینده[5] به‌عنوان نقاط دسترسی عمومی، دسترسی به سرویس‌های تحمل‌پذیر نفوذ را فراهم می‌کنند[۴]. درخواست‌های کاربران با توجه به نوع سرویس درخواستی، به یکی از



شکل ۲- نمای کلی معماری SITAR [۴]

سرویس‌دهنده‌های نماینده وارد می‌شوند. در این معماری سرویس درخواستی، توسط سرویس‌دهنده نماینده دریافت می‌شود و آن را برای سرویس‌دهنده‌های مؤلفه‌های تجاری[6] مورد نظر ارسال می‌کند و ضمناً در حین ارسال، سرویس‌دهنده‌های کنترلِ پذیرش و کنترلِ رأی را مطلع می‌کند. پاسخ‌های آماده‌شده توسط سرویس‌دهنده‌های مؤلفه‌های تجاری، ابتدا توسط سرویس‌دهنده‌های کنترل پذیرش دریافت و مورد بررسی قرار می‌گیرند. این سرویس‌دهنده‌ها پس از اعمال آزمون‌های درستی‌یابی روی پاسخ‌های دریافتی، پاسخ‌های دریافتی و نتیجه این آزمون‌ها را به سرویس‌دهنده‌های کنترل رأی ارسال می‌نمایند. سرویس‌دهنده‌های کنترل رأی وظیفه دارند در صورت وجود نفوذ در سرویس‌دهنده‌های مؤلفه‌های تجاری، آن را تشخیص داده و به دنبال آن، واحد پیکربندی مجدد را فعال سازند. برای تشخیص نفوذ می‌توان از آزمون پذیرش استفاده کرد. آزمون پذیرش با هدف بررسی منطقی بودن نتیجه اعلامی از سوی سرویس‌دهنده‌های مؤلفه‌های تجاری انجام می‌گیرد. این کار توسط برنامه‌نویس یا توسعه‌دهنده نرم‌افزار قابل تأمین است. آزمون پذیرش دربردارنده مجموعه‌ای مرتب از فعّالیت‌ها است که در صورت قابل قبول نبودن حالت سیستم، باعث بروز استثنا می‌شود و اگر در حین آزمون پذیرش این حالت رخ دهد، نشان‌دهنده خرابی یا سازش در آن خواهد بود. آزمون پذیرش، یک معیار تشخیص خطا است که توسط برنامه‌نویس در یک واحد نرم‌افزاری قرار داده می‌شود. بررسی‌های تنظیم وقت[7]، بررسی‌های کدگذاری، بررسی‌های برگشت، بررسی معقول بودن و بررسی ساختاری، از آزمون‌های پذیرش می‌باشند. در معماری SITAR، سه سطح دفاعی به شرح زیر ایجاد می‌شود[۴]:

1- Intrusion Containment
2- Durability
3- Intrusion Recovery
4- Scalable Intrusion- Tolerant Architecture for Distributed Services
5- Proxy Servers

6- (COTS) Commercial Of The Shelf
7- Timing Check

۱- درستی‌یابی درخواست‌های ورودی
۲- تست پذیرش پاسخ‌ها
۳- رأی‌گیری مخفی اکثریتی

به‌دلیل اینکه ممکن است هرکدام از مؤلفه‌های منفرد به مصالحه درآید، از یک واحد پیکربندی پشتیبان برای جلوگیری از نقاط خرابی منفرد استفاده می‌شود. معماری‌های قابل پیکربندی مجدد امکان می‌دهد که طیف وسیعی از راهبردهای تحمل‌پذیری خطا و نفوذ در سیستم وجود داشته باشد. این امر امکان پشتیبانی از سطوح مختلف امنیتی را به صورت همروند میسر می‌نماید. از آنجایی که همه مؤلفه‌های این معماری، قابلیت پیکربندی مجدد را دارند، واحد پیکربندی مجدد تضمین می‌کند که پیکربندی کل سیستم، سطح مناسبی از امنیت را ارائه نماید.

۵-۱-۱- مکانیزم‌های امنیتی در SITAR

برای فراهم‌سازی احتمال تشخیص نفوذ، فاز تشخیص نفوذ در زیرسیستم‌هایی به شرح زیر انجام می‌گیرد:

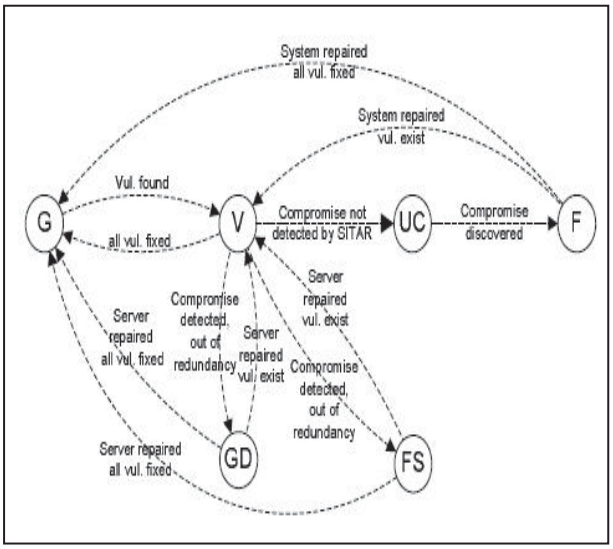
۱- تشخیص درخواست بدخواهانه توسط واحد کنترل پذیرش
۲- اجرای تست پذیرش روی پاسخ‌های اجرای درخواست توسط واحد کنترل پذیرش
۳- اجرای رأی‌گیری و ایجاد توافق توسط واحد کنترل رأی
۴- ممیزی اتوماتیک توسط واحد کنترل ممیزی

۵-۱-۲- مدل بنیادی رفتار امنیتی سیستم SITAR

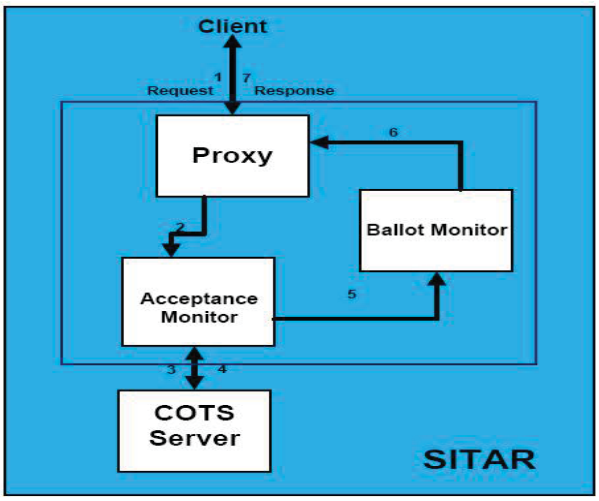
برای مدل‌سازی رفتار امنیتی سیستم تحمل‌پذیر به منظور ارزیابی صفات امنیتی، مهم این است که رفتار دو موجودیت اصلی یعنی «مهاجم»[1] و «پاسخ سیستم به حمله» را مدل نمائیم. رفتار امنیتی سیستم تحمل‌پذیر نفوذ می‌تواند با استفاده از تعریف حالات مرتبط و مختلف امنیتی و تعاملات داخلی بین آنها و با استفاده از دیاگرام گذر حالات بیان شود. شکل (۳) دیاگرام گذر حالت اولیه سیستم SITAR را نشان می‌دهد. در شکل (۴) مسیر حرکت درخواست کاربر و مسیر حرکت پاسخ با اعداد مشخص شده است.

۵-۲- معماری SCIT[2]

فرض مبنایی این معماری این است که تمامی سیستم‌های نرم‌افزاری منعطف بوده و مکانیزم‌های تشخیص نفوذ جهت تشخیص تمام حملات کافی نمی‌باشند[۵]. دلیل این فرض، ناکارآمدی مکانیزم‌های تشخیص نفوذ در کشف حملات پیچیده روی آسیب‌پذیری‌های ناشناخته می‌باشد. می‌توان گفت رهیافت اصلی معماری، پاکسازی متناوب سرویس‌دهنده‌های سیستم بدون توجه به آلودگی آنها به‌عنوان دفاع در عمق می‌باشد.



شکل ۳- دیاگرام گذر حالت اولیه سیستم SITAR [۴]

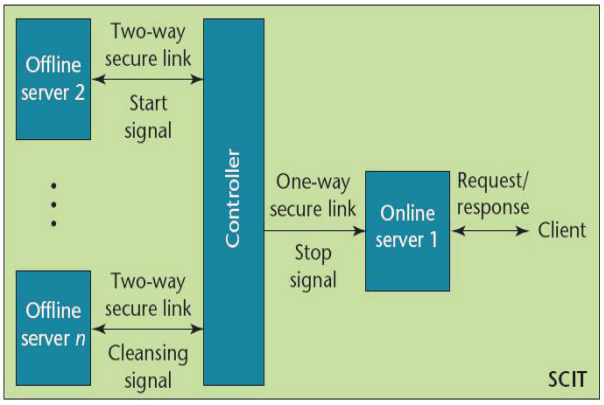


شکل ۴- مسیر حرکت درخواست و حصول نتیجه در معماری SITAR

در این مدل، بدون توجه به وجود یا عدم وجود مصالحه در مولفه‌ها، عملیات پاکسازی را برای بازگرداندن وضعیت سیستم به حالت سالم انجام می‌دهد و از هیچ مکانیزمی برای تشخیص و پیشگیری نفوذ استفاده نمی‌شود و از پاکسازی متناوب سیستم به‌عنوان توسعه‌ای از رهیافت دفاع در عمق استفاده می‌گردد[۵].

میزان کارایی و اثربخشی این روش، به سرعتِ پاکسازی سیستم‌ها بستگی دارد و حمله‌کننده فرصت کوتاهی برای رخنه در سیستم دارد. شکل (۵) نمای سطح بالایی از خوشه SCIT را نشان می‌دهد.

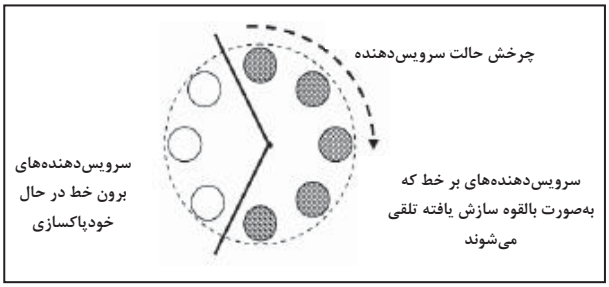
1- Attacker
2- Self Cleansing Intrusion Tolerance



شکل ۵- نمای سطح بالای SCIT [۵]

معماری SCIT دارای سه پارامتر قابل تنظیم: پنجره زمان آشکارسازی، زمان‌اصلاح[1] و تعداد گره‌ها درخوشه[2] می‌باشد. این پارامترها به هم مرتبطاند. مثلاً برای تعیین مقداری برای W، لازم است حداکثر زمان لازم برای پاک‌کردن و همچنین تعداد ندهای قابل دسترس، معلوم باشند.

شکل (۶) وضعیت سرویس‌دهنده را نشان می‌دهد. حالت هر سرویس‌دهنده در این خوشه به‌طور متناوب میان دو حالت برخط برای سرویس‌دهی به مشتری و برون از خط برای انجام عملیات پاکسازی تغییر می‌کند.



شکل ۶- وضعیت سرویس‌دهنده در معماری SCIT [۷]

استفاده از اصلاح چرخشی سبب می‌شود در هر لحظه سرویس‌دهنده موجود در یک خوشه، یکی از سه حالت: خارج از خطوط در حال اصلاح، برخط اولیه یا برخط در حال نسخه پشتیبان را داشته باشد. طول مدتی که سرویس‌دهنده تحمل‌پذیر نفوذ خوداصلاح در اینترنت در معرض پایش است، زمان آشکاری[3] خوانده می‌شود. میزان کارایی و اثربخشی این معماری، به طول زمان آشکاری و سرعت پاکسازی سیستم بستگی دارد[۷].

در این معماری کنترل‌کننده، مؤلفه اصلی است که چرخش را برای ورود و خروج عملیات اصلاح، مدیریت می‌کند.

یک خوشه تحمل‌پذیری نفوذ خوداصلاح، شامل مجموعه‌ای از سرویس‌دهنده‌های به‌هم متصل است که به‌صورت هماهنگ برای ارائه سرویس‌های از پیش‌تعیین‌شده با همدیگر همکاری می‌کنند.

در کاربردهایی با دسترس‌پذیری بالا از میان مجموعه سرویس‌دهنده‌های اولیه فعال، یکی از سرویس‌دهنده‌ها به‌عنوان سرویس‌دهنده اصلی و بقیه به‌عنوان سرویس‌دهنده‌های پشتیبان عمل می‌کنند. وظیفه فرآیند چرخش، تغییردادن متناوب نقش سرویس‌دهنده‌ها در طول زمان است. هر سرویس‌دهنده پس از خروج از خوشه تحمل‌پذیری نفوذ خوداصلاح، برای فعال‌سازی رویه پاکسازی، راه‌اندازی مجدد می‌شود که وظیفه این رویه، بازگرداندن سیستم به یک حالت خوش‌تعریف و سالم است.

میزان تحمل‌پذیری نفوذ سیستم‌هایی که بر اساس این معماری می‌باشند، بر پایه اینکه چه اندازه توانایی محدودسازی خسارات را دارند، سنجیده می‌شود[۴و۵].

۵-۳- معماری MAFTIA[4]

MAFTIA جهت ساخت سیستم‌های تحمل‌پذیر نفوذ، از ترکیب مکانیزم‌های مختلف استفاده می‌کند. به دلیل اینکه سیستم‌های تحمل‌پذیر نفوذ باید قادر به ارائه سرویس‌های امن حتی در حضور نفوذها باشند، «دفاع در عمق» یک راهبرد مورد نیاز برای اجتناب از تبدیل شدن هر یک از مؤلفه‌های سیستم به یک «نقطه شکست واحد[5]» محسوب می‌شود که در معماری MAFTIA استفاده شده است. پنج توانایی MAFTIA برای ایجاد راهبرد دفاع در عمق و تحمل‌پذیری نفوذ به شرح زیر می‌باشند[۶]:

۱) تشخیص نفوذ: هدف، تشخیص انواع مختلف سوء استفاده و انواع ناهنجاری‌ها می‌باشد.

۲) قراردادهای ارتباط گروهی مبتنی بر ساختارهای متفاوت عمومی‌سازی شده[6]: هدف، ترکیب کلاسیک و متعارف فنون تحمل‌پذیری خطا با فنون تنوع طراحی و پیاده‌سازی است.

۳) فنون رمزنگاری[7]: هدف این فنون، انتقال امن داده‌ها در کانال‌های فیزیکی نا امن و ذخیره‌سازی امن داده‌ها در رسانه‌های ذخیره‌سازی می‌باشد.

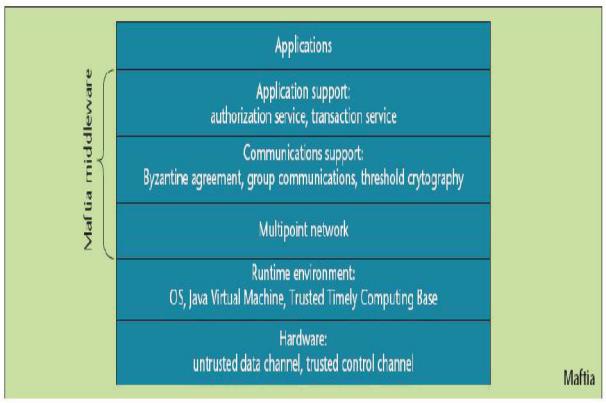
۴) جداسازی و پخش داده[8]: به‌منظور ایجاد سختی در تفسیر داده‌ها است که در صورت دستیابی غیرمجاز به داده‌ها، داده‌های قابل استفاده به‌دست نیاید.

1- Tcleansing
2- N-total
3- Exposure Time

4- Malicious and Accidental Fault Tolerance for Internet Applications
5- Single Point of Failure
6- Group Communication Protocols Based on Generalized Adversary Structure
7- Cryptographic Techniques
8- Data Fragmentation and Scatrring

۵) کنترل دسترسی[1]: به‌منظور قانونمندسازی دسترسی به منابع براساس ملاحظات «حداقل مجوز دسترسی» و «نیاز به آگاهی» می‌باشد.

هدف اصلی در معماری MAFTIA، استفاده از مکانیزم‌های تحمل‌پذیری برای ایجاد کاربردهای توزیع‌شده اتکاپذیر در مقیاس بزرگ و امکان ارائه سرویس‌های الکترونیکی در جوامع اطلاعاتی، توسعه قابلیت‌های سیستم‌عامل و شفاف‌سازی ناهمگونی میان سیستم‌های عامل با ارائه یک APIهمگون و چارچوبی برای ترکیب پروتکل‌ها است. در این معماری، خطاهای تصادفی و حملات بدخواهانه، تحمل‌پذیر می‌باشند. شکل (۷) ساختمان سیستم با معماری MAFTIA را نشان می‌دهد. در این معماری فرض این است که اغلب سخت‌افزارها غیر قابل اعتماد هستند[۶]. لایه‌های میان‌افزار براساس مکانیزم‌های پشتیبان زمان اجرا، اجرا می‌شوند و برنامه‌های کاربردی که روی MAFTIA اجرا می‌شوند، از تجرید ایجادشده توسط میان‌افزار و سرویس‌های کاربردی، برای اجرای امن عملیات میان چند میزبان و یا دسترسی امن کاربران از راه دور با وجود خطاهای بدخواهانه استفاده می‌کنند.



شکل ۷- ساختمان معماری MAFTIA

براساس [۶] رویکرد تحمل‌پذیری نفوذ MAFTIA به سه دلیل جذاب است:

۱- پروتکل‌های گروهی و سایر سرویس‌های امنیتی قابل اعتماد را ارائه می‌کند.

۲- معماری بین لایه‌های مختلف سرویس‌های امنیتی بر پایه اعتماد است.

۳- این معماری برای توانمندی‌های تشخیص، به رأی‌گیری و تسهیم امنیت به پروتکل‌های توزیع شده اعتماد می‌کند. این پروتکل‌های پشتیبانی توسط سخت‌افزار مورد اعتماد و محیط زمان اجرا در سطوح پایین‌تر تقویت شده است جدول (۱) به طور مختصر ویژگی‌های سه معماری MAFTIA, SCIT و SITAR را بیان می‌کند [۴].

جدول ۱- مشخصات معماری سیستم‌های تحمل‌پذیر نفوذ (SITART, SCIT, MAFTIA) [۴و۶]

معماری / عملکرد	SITAR	MAFTIA	SCIT
بازدید از payload دارد؟	بلی	بلی	نه
الگوریتم رای‌گیری استفاده می‌شود؟	بلی ، برای تشخیص خطا و باقی بودن حملات	بلی ، برای تشخیص خطا و باقی بودن حملات	نه
قطعیت وجود دارد؟	نه	نه	بلی
میزان کارایی	توجه به زمان پاسخ	توجه به زمان پاسخ	توجه روی سیکل‌های محاسباتی برای سرویس‌دهنده نمونه جدید
اجرای الگوریتم سیستم تحمل نفوذ	در جریان داده‌ای برنامه	در جریان داده‌ای برنامه	خارج از اندازه
فن تنوع استفاده می‌شود؟	استفاده از فن تنوع لازم است	استفاده از فن تنوع لازم است.	اختیاری
فن ترمیم استفاده می‌شود؟	فن ترمیم بر اساس عمل تشخیص نفوذ اجرا می‌شود.	فن ترمیم بر اساس عمل تشخیص نفوذ اجرا می‌شود. عین خطا مطابق با حالت سالم پوشش داده می‌شود.	فن ترمیم متناوب (زمانی) به‌وسیله کنترل‌کننده بر مبنای کپی اصلی اجرا می‌شود.

MAFTIA براساس پروتکل‌های زمانی بیزانتین[2] و بر پایه مؤلفه‌های قابل اعتماد که TTCB[3] نامیده می‌شوند، می‌باشد. TTCB یک هسته امنیتی توزیع‌شده است که برای کنترل اختصاصی شبکه جهت فراهم‌سازی پشتیبانی قوی و محکم در مقابل خرابی مورد استفاده قرار می‌گیرد. هر مؤلفه TTCB روی هر میزبان MAFTIA که به دارا بودن طراحی دارای خرابی پنهان تظاهر می‌کند، نصب می‌شود. یک راه و رویکرد برای حصول آن، به‌کارگیری فنون جدا از هم (از نظر فیزیکی) و سخت‌افزارهای مخصوص می‌باشد. TTCB می‌تواند از

1- Access Control

2- Byzantine Protocols
3- Timly Trusted Computing Base

اجرای صحیح پروتکل‌های زمانی بیزانتین حتی در حضور خطاهای بدخواهانه پشتیبانی کند.

سرویس‌هاس TTCB به دو دسته: سرویس‌های مرتبط با امنیت[1] و سرویس‌های مرتبط با زمان[2] تقسیم می‌شوند[۶].

از سرویس‌های مرتبط با امنیت می‌توان سرویس احراز هویت محلی، سرویس احراز هویت توزیع‌شده و سرویس تولید شماره تصادفی امن را نام برد.

از سرویس‌های مرتبط با زمان می‌توان سرویس اجرای به موقع و مطمئن، اندازه‌گیری مدت زمان مطمئن اجرای سرویس، کشف خطای تنظیم وقت مطمئن و مهر زمان قطعی مطمئن را نام برد.

۶- راهکار پیشنهادی برای توسعه سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ

با توجه به معماری‌های معرفی‌شده در جهت اهداف مقاله، رویکرد مورد نظر در قالب فنون مؤثر برای توسعه سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ، مؤلفه‌ها و کاربرد هر یک از آنها برای طراحان سیستم‌های نرم‌افزاری به شرح زیر پیشنهاد می‌شود که متناسب با نیاز می‌تواند مورد انتخاب و استفاده قرار گیرد:

۶-۱- راهبرد اصلی طراحی

استفاده از ادغام روش‌های متعارف امنیتی مانند جلوگیری از نفوذ، تشخیص نفوذ، رفع نفوذ و فنون تحمل‌پذیری خطا به‌عنوان راهبرد اصلی طراحی سیستم‌های تحمل‌پذیری نفوذ مورد استفاده قرار گیرد. هر یک از فنون تک‌نگارشی و چندنگارشی (که از انواع فنون تحمل‌پذیری خطا می‌باشند) را می‌توان متناسب با کاربرد سیستم نرم‌افزاری انتخاب و در جای خود استفاده کرد. فنون تک‌نگارشی برای کاربردهای خاص، و فنون چندنگارشی برای کاربردهای عام مناسب می‌باشند.

۶-۲- تأمین دسترسی غیر مستقیم کاربران به مؤلفه‌های پردازشی

برای جلوگیری از دسترسی مستقیم کاربران به مؤلفه‌های سرویس‌دهنده سیستم نرم‌افزاری، حتماً از یک واسط مناسب مانند سرویس‌دهنده‌های نماینده در معماری SITAR استفاده شود و اجازه ورود درخواست به بخش پردازشی سیستم نرم‌افزاری بدون بررسی و اطمینان از بدخواهانه نبودن آن داده نشود. به این وسیله تا حدی امنیت سیستم رعایت می‌شود.

۶-۳- تأمین احراز هویت متقاضی سرویس برای جلوگیری از نفوذ تکراری

از فن مناسب برای احراز هویت متقاضی استفاده گردد. این امر می‌تواند در جلوگیری از نفوذ تکراری مؤثر باشد.

۶-۴- مؤلفه تکراری

ماهیت سیستم نرم‌افزاری هدف، می‌تواند خاص منظوره و یا عام منظوره باشد. با توجه به ماهیت سیستم، از فنون مختلف تحمل‌پذیری خطا جهت تکمیل مقابله با نفوذ استفاده می‌شود. در سیستم‌های نرم‌افزاری با کاربرد خاص از فن مؤلفه پشتیبان (در این فن، در کنار هر مؤلفه، یک نسخه مشابه آن مؤلفه برای پشتیبانی استفاده می‌شود) می‌توان بهره‌برداری نمود و در سیستم‌های نرم‌افزاری با کاربرد عام از فن مؤلفه تکراری (فنی که وجود هر تعداد مؤلفه افزونه را مجاز می‌داند) استفاده می‌شود. ایده داشتن مؤلفه‌های تکراری در سیستم، مستلزم پذیرش هزینه بالایی است ولی در تأمین تحمل‌پذیری نفوذ، اثربخش‌تر است.

۶-۵- ثبت سوابق

از مؤلفه ثبت سوابق حملات موفق، برای تشخیص حملات موفق تکراری می‌توان استفاده کرد. همچنین می‌توان عملکرد هر مؤلفه را به‌منظور ممیزی مورد استفاده قرار داد. این رفتار برای شناسایی مؤلفه‌های مصالحه‌ای (مؤلفه‌های مورد نفوذ قرار گرفته) در سیستم مورد استفاده قرار می‌گیرد.

۶-۶- تست‌پذیرش

از مؤلفه تستِ پذیرش در شرایط استفاده از فن برنامه‌نویسی چندنگارشی، به‌منظور تشخیص نفوذ می‌توان استفاده نمود. تست پذیرش و برنامه‌نویسی چندنگارشی هر یک از فنون تحمل‌پذیری خطا می‌باشد. تست پذیرش می‌تواند مانع از حملات به جامعیت داده‌های سیستم باشد. البته با توجه به افزونگی مورد استفاده، لازم است تعادل بین هزینه و کارایی مورد بررسی قرار گیرد.

۶-۷- فن رأی‌گیری

برای تعیین نتیجه نهایی اجرای درخواست در معماری‌هایی که در آنها مؤلفه‌های پردازشی تکراری مورد استفاده قرار می‌گیرند، از مکانیزم‌های رأی‌گیری استفاده می‌شود. هر مکانیزم رأی‌گیری، از الگوریتم‌های رأی‌گیری خاصی استفاده می‌کند و هر الگوریتم رأی‌گیری دارای مزایا و معایب خاص خود است.

۶-۸- پیکربندی مجدد

تأمین قابلیت مدیریت مؤلفه‌های مصالحه‌ای: پیش‌بینی روش پیکربندی مجدد مؤلفه‌های مصالحه‌ای و زمینه‌های مناسب مکانیزم

1- Security Related Services
2- Time Related Services

انجام کار و امکانات موجود برای اجرای آن می‌تواند در پوشش تحمل‌پذیری نفوذ بسیار مؤثر باشد.

۶-۹- پیکربندی کل سیستم

تأمین قابلیت مدیریت مؤلفه‌های مصالحه‌ای: اگر همه مؤلفه‌های سیستم، قابلیت پیکربندی مجدد را داشته باشند، واحد پیکربندی مجدد تضمین می‌کند که پیکربندی کل سیستم، سطح مناسبی از امنیت را فراهم نماید. بنابراین، مانند معماری SITAR اگر بتوان طراحی سیستم را طوری انجام داد که همه مؤلفه‌های سیستم، قابلیت پیکربندی مجدد را داشته باشند، در جهت تحمل‌پذیری نفوذ سیستم گام اساسی برداشته شده است.

۶-۱۰- فنون برنامه‌نویسی چندنگارشی

استفاده از فنون برنامه‌نویسی چندنگارشی به‌عنوان مصداقی برای مؤلفه‌های تکراری از فنون مهم در تأمین تحمل‌پذیری نفوذ می‌باشد. برای پیاده‌سازی آن می‌توان از فنون دیگری مانند تنوع طراحی بهره برد.

۶-۱۱- ایجاد ارتباط منطقی

پیش‌بینی ارتباط منطقی بین مؤلفه تشخیص نفوذ و مؤلفه پیکربندی، از گام‌های مهم رسیدن به هدف اصلی است.

۶-۱۲- فن زوج پردازنده

برای تأمین دسترس‌پذیری بالا در سیستم‌نرم‌افزاری می‌توان از فن زوج‌پردازنده استفاده نمود. فن زوج پردازنده از فنون تحمل‌پذیری خطا می‌باشد. برای استفاده از این فن و بهره‌برداری از نتایج آن، لازم است علاوه بر مؤلفه اصلی و تکرار آن، هر یک از آنها روی یک پردازنده مستقل اجرا شوند. این فن در سیستم‌هایی که مأموریت بحرانی باشند کاربرد بیشتری دارد.

۶-۱۳- فنون تنوع طراحی

از جمله فنون مهم در کاهش آسیب‌پذیری‌های یکسان مؤلفه‌های تکراری، استفاده از فنون تنوع طراحی است. استفاده از این فن می‌تواند وجود مؤلفه‌های تکراری سیستم نرم‌افزاری را هدفمند و در راستای اهداف خاص قرار دهد. از این فن می‌توان برای درستی‌یابی نتیجه پردازش مؤلفه‌ها استفاده نمود.

۶-۱۴- تأمین قابلیت عدم حضور مؤلفه با خاصیت نقطه شکست واحد بودن

به‌عنوان یک سیاست بازدارنده، باید تمهیدات لازم پیش‌بینی شود تا در صورت وجود مؤلفه‌ای به‌عنوان نقطه شکستِ واحد در سیستم، این مؤلفه از حالت نقطه شکست واحد بودن خارج گردد. به این معنی که علی‌رغم به‌کارگیری روش‌ها و فنون متعدد و مؤثر در تأمین تحمل‌پذیری نفوذ، وجود حتی یک مؤلفه که نقطه شکست واحد باشد می‌تواند در شکست سیستم بسیار مؤثر باشد. توجه به این مهم سبب می‌شود سیستم توسعه داده‌شده، بقاپذیری بالایی داشته باشد.

۶-۱۵- پایداری سیستم نرم‌افزاری پس از رخداد نفوذ

شناسایی حملات منع سرویس روی الگوریتم‌های پیکربندی مجدد و خنثی‌سازی آنها انجام گرفته و سبب متوقف نشدن اجرای الگوریتم‌های پیکربندی مجدد و رفع نفوذ و در نهایت، موجب پایداری سیستم نرم‌افزاری پس از رفع نفوذ می‌شود.

۶-۱۶- ممانعت از گسترش محدوده نفوذ

وقتی در سیستم نرم‌افزاری، نفوذی تشخیص داده شد و اعلام گردید، با استفاده از فنون خاصی، ابتدا محدوده رخداد نفوذ تعیین شده و ارتباط آن محدوده با سایر نواحی سیستم قطع می‌گردد تا نفوذ و اثرات آن در سیستم منتشر نگردد. از ابزار یادشده می‌توان ماتریس وابستگی آسیب‌پذیری و گراف نفوذ را نام برد.

۶-۱۷- کشف مؤلفه مصالحه‌ای

با استفاده از فنون آزمون پذیرش، از خدشه‌دار شدن جامعیت داده‌های سیستم جلوگیری شده و اطمینان از صحت پاسخ پردازش درخواستی حاصل می‌گردد و همچنین وقتی که فن برنامه‌نویسی چندنگارشی مورد استفاده قرار گرفته باشد برای تعیین مؤلفه مصالحه‌ای، از تست پذیرش می‌توان استفاده کرد. جدول (۲) به‌صورت مختصر فنون و آثار استفاده از آنها را بیان می‌کند.

جدول ۲- فنون تحمل‌پذیری خطا و اثر استفاده آنها

فن / روش	اثر استفاده
احراز هویت	افزایش امنیت سیستم
ثبت سوابق	شناسایی حملات موفق تکراری
تست پذیرش	اطمینان از صحت پاسخ پردازشی
رأی‌گیری	تعیین نتیجه نهایی پردازش
زوج پردازنده	افزایش تحمل‌پذیری نفوذ و دسترس‌پذیری بالا
پیکربندی مجدد	افزایش تحمل‌پذیری نفوذ
تنوع طراحی	کاهش احتمال خطاهای طراحی مشترک
برنامه‌نویسی تک‌نگارشی	افزایش تحمل‌پذیری نفوذ
برنامه‌نویسی چندنگارشی	افزایش تحمل‌پذیری نفوذ
پیشگیری از نفوذ	افزایش امنیت سیستم
تشخیص نفوذ	افزایش امنیت سیستم
حذف نفوذ	افزایش امنیت سیستم
مؤلفه‌های تکراری	افزایش تحمل‌پذیری نفوذ
اعتبارسنجی مبداء درخواست	شناسایی حملات منع سرویس توزیع شده

۷- ارائه یک نمونه کاربردی

براساس [۱۱] یک معماری چندلایه‌ای برای سرویس‌های وب تحمل‌پذیر نفوذ مطابق شکل (۸) ارائه شده است که در آن، از فنون تحمل‌پذیری خطا، فن تک‌نگارشی و تطبیق آن با شرایط خرابی بدخواهانه استفاده شده است. این معماری برای کاربردهای سرویس‌های اختصاصی مناسب است. در این معماری، سرویس وب فقط دارای یک نسخه پشتیبان است که به‌منظور ترمیم سرویس وب مصالحه‌ای مورد استفاده قرارمی‌گیرد.

شکل ۸- معماری چندلایه‌ای سرویس وب تحمل‌پذیر نفوذ [۱۱]

۸- نتیجه‌گیری و پیشنهادها

برخلاف توانایی‌های ایجاد شده برای برقراری امنیت سیستم‌های نرم‌افزاری، نفوذگران همواره پیشاپیش متخصصین امنیت حرکت می‌کنند و سیستم‌های نرم‌افزاری مورد هجوم قرار می‌گیرند. علاوه بر آن، آسیب‌پذیری‌های انکارناپذیر در محصولات نرم‌افزاری و امکان حذف کامل آسیب‌پذیری‌های سیستم‌های نرم‌افزاری ممکن نیست. یک راه برای پوشش ضعف شیوه‌های مقابله با نفوذ، ایجاد زمینه لازم برای تحمل آنها در سیستم نرم‌افزاری است؛ به این معنی که نرم‌افزار قابلیتی پیدا کند که حتی در صورت رخداد نفوذ موفق در آن، قادر به استمرار ارائه خدمات خود باشد. تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری با ترکیب و تلفیق روش‌های متعارف امنیتی و فنون تحمل‌پذیری نفوذ حاصل می‌شود. تأمین تحمل‌پذیری نفوذ موجب می‌شود سیستم نرم‌افزاری قابلیتی پیدا کند که علی‌رغم مورد نفوذ قرار گرفتن، توانایی خودترمیمی و پوشش نفوذ را می‌یابد. در مقاله حاضر، راهکارهایی مناسب برای توسعه سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ ارائه گردید. راهکارهای مذکور حاصل بررسی سه معماری برتر سیستم‌های نرم‌افزاری تحمل‌پذیر نفوذ می‌باشند. طراح سیستم نرم‌افزاری جدید می‌تواند با توجه به شرایط کاربردی سیستم نرم‌افزاری، برخی از مؤلفه‌های تحمل‌پذیر نفوذ را به‌طور مناسب انتخاب و در توسعه سیستم مورد استفاده قرار دهد. در آینده قصد داریم با توکل به خداوند متعال، به موارد زیر پرداخته و نتایج را در قالب مقالاتی مناسب ارائه نمائیم.

- شناسایی فنون جدید مقابله با نفوذ در سیستم‌های نرم‌افزاری
- تأثیر استفاده از فنون مختلف تحمل‌پذیری خطا، در جهت تحمل‌پذیری نفوذ سیستم‌های نرم‌افزاری
- تحمل‌پذیری نفوذ در سرویس‌های وب
- تحمل‌پذیری نفوذ در سیستم‌های نرم‌افزاری مبتنی بر وب

مراجع

1. Agirdas Avi_zienis, Fellow, Jean-Claude Laprie, Brian Randell, and Carl andwehr," Basic Concepts and Taxonomy of Dependable ",IEEE transactions on dependable and secure computing, Vol. 1, No. 1, january-march (**2004**).
2. E. Dunrova," fault tolerant design: an introduction", kluwer academic publishers, (**2008**).
3. Dazhi Wang, Bharat B. Madan, "Security Analysis of SITAR Intrusion Tolerance System", U.S. Department of Defense Advanced Research Projects Agency (DARPA), (**2007**).
4. a. A. S. Quyen L. Nguyen, "Comparative Analysis of Intrusion-Tolerant System Architectures" IEEE Security and Privacy, (**2011**).
5. A. K. B. a. A. K. sood, "Securing Web Servers Using Intrusion Tolerance (SCIT)," in Proc of the second International Conference in Dependability, (**2009**).
6. Ian Welch, John Warne, Peter Ryan, "Architectural Analysis of MAFTIA's Intrusion Tolerance Capabilities", MAFTIA deliverable D99,Public document, February 3rd (**2003**).
7. J. W. Ian Welch, Peter Ryan, Robert Stroud, "Architectural Analysis of MAFTIA's Intrusion Tolerance Capabilities," (**2003**).
8. J. C. Agustin Orfil, Arturo Ribagord, "Autonomous decision on intrusion detection with trained BDI agents" Computer communications, Vol. 31, pp. 1803-1813, (**2008**).
9. R. R. e. a. Obelheiro, "How Practical are Intrusion-Tolerant Distributed Systems? " (**2006**).
10. K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS) " pp. 800-94, (**2007**).
11. Z. Aghajani and M. Abdollahi Azgomi, " A Multi-Layer Architecture for Intrusion Tolerant Web Services ," International Journal of u- and e- Service, Science and Technology , SERSC, ISSN: 2005-4246, Vol. 1, No. 1, pp. 73-80, (**2008**).

Intrusion Tolerance, Modern Approach for Software Systems Development

S. Bejani[1]

M. Abdollahi[2]

Abstract

Today, we see the effects of using software systems in speed and accuracy of processes. Every intrusion in software systems, has irrevocable loss. The security traditional methods can not defend against intrusions, and using intrusion tolerance techniques is necessary and acceptable. For software intrusion tolerance, security traditional approach and fault tolerant techniques are useful. Using intrusion tolerant techniques are means that provide continuity of predicted software system services even at the presence of intrusion.

Determining the effective parameters in intrusion tolerance and the scope of impacts of each of these parameters are considered an effective step towards producing software systems which are not accessible to designers.

In this paper, we exploit intrusion tolerant software systems architecture as SCIT, MAFTIA, SITAR and their components. Our methods exploit intrusion tolerant components that are useful for software developers.

Key Words: *Intrusion, Intrusion Tolerance, Intrusion Tolerance Software Systems, Conventional Security Methods*

1- Imam Hossein University- Doctoral Candidate of Computer Engineering- Writer in Charge (sbejani@ihu.ac.ir)

2- Iran's Science and Technology University- Faculty of Computer Engineering- Assistan Professor and Academic Member