امین رای

گسترش فناوری پارس امین رای

نـام سنـد طراحی معماری امنیتی برای سیستم سادا	
تهیه کننده: گسترش فناوری پارس امین رای **طبقه بندي:** محرمانه	**تاریخ نشـر:** ۱۴۰۲/۰۶/۲۶
کد سند: AMR_SADA_Security_Architecture_14020626_v1.0	**تاریخ آخرین بازنگري:** ۱۴۰۲/۰۷/۰۳

به‌نام خدا

فهرست مطالب

عنوان	صفحه
فصل ۱- مقـدمــه	**۴**
فصل ۲- نکات طراحی معماری امنیتی نرم‌افزار	**۵**
۱-۲- اصول طراحی امن	۵
۲-۲- سرویس‌های امنیتی مشترک	۸
فصل ۳- معماری امنیتی سادا	**۹**
۱-۳- هدف سند	۹
۲-۳- مخاطبین سند	۹
۳-۳- دیاگرام معماری امنیتی سادا	۹
۴-۳- تشریح معماری امنیتی سادا	۹

فصل ۱- مقــدمــه

معماری امنیتی یک نرم‌افزار دربرگیرنده تمام عناصری است که مکانیزم‌های امنیتی موردنیاز را در سطح طراحی معمارانه آن مشخص می‌کنند. این مکانیزم‌ها را می‌توان در طی فاز تحلیل و گردآوری نیازمندی‌های امنیتی نرم‌افزار، شناسایی کرد. سند نیازمندی‌های امنیتی سادا در یک فایل تحت عنوان AMR_SADA_Security _Requirements_14020611_v1.0 ارائه شده است. در سند پیش رو که توسط شرکت «امین رای» تهیه شده است، معماری امنیتی سیستم سادا ترسیم و تشریح شده است.

در ادامه ابتدا در فصل اول، فرایند و نکات طراحی معماری امنیتی یک نرم‌افزار آورده شده است، سپس معماری امنیتی سیستم سادا در فصل دوم تشریح شده است.

فصل ۲– نکات طراحی معماری امنیتی نرم‌افزار

در فرایند توسعه نرم‌افزارها، پس از تحلیل و شناسایی نیازمندی‌های کارکردی و غیرکارکردی نرم‌افزار، نوبت به طراحی نرم‌افزار می‌رسد. در این مرحله نیازمندی‌های شناسایی شده برای نرم‌افزار در قالب یک معماری، به تصویر کشیده می‌شوند. معماری نرم‌افزار، مشخص‌کننده عناصر و اجزای تشکیل‌دهنده نرم‌افزار، نحوه سازمان‌دهی و ارتباطات بین آنها است. این اجزا و ارتباطاتشان، اهداف تعیین شده در نیازمندی‌های شناسایی شده را برآورده خواهند کرد. بنابراین معماری نرم‌افزار، نقشه راهی برای توسعه‌دهندگان در فاز پیاده‌سازی نرم‌افزار خواهد بود و در طراحی آن باید دقت کافی وجود داشته باشد. درواقع درصد قابل توجهی از خطاهایی که بعدا در تست نرم‌افزار شناسایی می‌شوند، به دلیل وجود اشتباهات و نقص‌ها در طراحی معماری نرم‌افزار بوده است. حال اگر این اشتباهات و نقص‌ها مربوط به عدم توجه کافی به برآورده شدن نیازمندی‌های امنیتی در معماری باشند، در نهایت منجر به آسیب‌پذیر شدن نرم‌افزار در برابر انواع حملات خواهند شد.

بنابراین طراحی معماری یک نرم‌افزار باید طراحی معماری امنیتی برای آن باشد. به عبارت دیگر، معماری نرم‌افزار باید دربرگیرنده تمام اجزایی باشد که کارکردهای امنیتی موردنیاز را فراهم کنند. برای رسیدن به این معماری، ابتدا باید نیازمندی‌های امنیتی نرم‌افزار در فاز تحلیل نیازمندی‌ها، شناسایی و سپس در ترکیب مولفه‌ها و ارتباطاتشان، دیده شوند. علاوه بر این، در معماری امنیتی نرم‌افزار، تکنولوژی‌های امنی که برای توسعه، استقرار و عملیاتی شدن نرم افزار موردنیاز هستند نیز شناسایی می‌شوند. در طراحی معماری امنیتی نرم‌افزار، نکاتی وجود دارد که ادامه این بخش، به آنها اشاره شده است.

۲-۱-اصول طراحی امن

در این بخش برخی از مهمترین اصولی که در طراحی معماری امنیتی یک نرم‌افزار و به طور کلی طراحی امن آن، بهتر است مورد توجه قرار گیرند، آورده شده است.

- **Least privileges**: یکی از مهمترین اصولی که در طراحی امن یک سیستم باید در نظر گرفته شود، اصل حداقل دسترسی یا همان Least privileges است که هدف آن توجه به این نکته است که یک subject که می‌تواند یک شخص یا یک سرویس و یا یک فرایند باشد، فقط باید حقوق و مجوزهای دسترسی ضروری را که برای انجام و تکمیل وظیفه جاری خود به آنها نیاز دارد، در اختیار داشته باشد و نه بیشتر. اعمال این اصل، میزان صدمات ناشی از به خطر افتادن سیستم را کاهش خواهد داد. برای این منظور باید سطوح مختلفی از دسترسی‌ها را برای وظایف مختلف در نظر گرفت و حتی اگر یک subject وظایف مختلفی در یک سیستم داشته باشد، آنها را با سطوح متناسب خود بتواند اجرا کند و امکان اجرای این وظایف با بالاترین سطح دسترسی هرگز فراهم نباشد.
- **Defense in depth**: یکی از اولین اصولی که در طراحی امن مطرح می‌شود، اصل دفاع در عمق یا همان Defense in depth است. این اصل که با نام layered security نیز شناخته می‌شود، به

وجود چندین مکانیزم امنیتی برای دفاع از سیستم در برابر حملات اشاره می‌کند؛ به طوری که اگر یک مکانیزم در دفاع از سیستم شکست بخورد و دور زده شود، مکانیزم بعدی در برابر حمله مقاومت کند.

- **Separation of duties**: در این اصل، تایید تکمیل یک کار نیازمند برآورده شدن دو یا چند شرط است که هریک از آنها به تنهایی برای تکمیل آن کافی نیستند. در تعبیری دیگر، اصل تفکیک وظایف یا همان Separation of duties به این موضوع اشاره می‌کند که برای تکمیل یک کار، باید چندین نفر مشارکت کنند. برای این منظور، آن کار باید به چندین آیتم تقسیم شود و هر بخش از آن توسط یک نفر انجام شود. در این صورت هیچ کس نمی‌تواند به تنهایی از سیستم سوءاستفاده کند و آن را به خطر بیندازد.
- **Fail secure**: با توجه به اینکه امکان شکست یا failure در تمام سیستم‌ها وجود دارد، اصل Fail secure که به آن Fail safe هم گفته می‌شود به این موضوع اشاره می‌کند که وقتی سیستم دچار شکست شود، باید همچنان در وضعیت امنی باقی بماند. به این معنا که ویژگی‌هایی از سیستم که مرتبط با سه اصل محرمانگی، صحت و دسترس‌پذیری در امنیت هستند، همچنان باید حفظ شوند. در این راستا باید تمام مسیرهایی که منجر به قرار گرفتن سیستم در وضعیت شکست خواهند شد، شناسایی و نحوه مدیریت آنها برای بازگشت سیستم به وضعیت پایدار و امن مشخص شود.
- **Secure by default**: این اصل که معمولا در کنار اصل fail secure آورده می‌شود، به مفهوم اعطای صریح دسترسی اشاره می‌کند. به این معنا که به صورت پیش‌فرض یک subject از دسترسی به یک object (منبع)، منع شده است مگر آنکه به صورت صریح این دسترسی اعطا شده باشد. درواقع بهتر است دسترسی‌های سیستم به صورت پیش‌فرض بسته باشند، مگر آنکه به صورت صریح از طریق مجوزهایی برای کاربران مشخص باز شده باشند.
- **Complete Mediation**: این اصل به این موضوع اشاره می‌کند که مکانیزم مجازشماری سیستم هرگز نباید دور زده شود. به عبارت دیگر، با هر دسترسی subject به object و عملی که می‌خواهد روی آن انجام دهد، مجاز بودن آن باید بررسی شود؛ حتی اگر این دسترسی بارها و بارها اتفاق بیفتد. بنابراین باید تمام موقعیت‌هایی که احتمال دور زده شدن مکانیزم مجازشماری سیستم وجود دارد، شناسایی و از وقوع آنها ممانعت شود.
- **Economy of Mechanism**: این اصل به موضوع سادگی طراحی سیستم اشاره می‌کند. درواقع پیچیدگی و امنیت متضاد یکدیگر هستند. هرچه یک سیستم پیچیده‌تر باشد، فهم آن و در نتیجه امنسازی آن سخت‌تر خواهد شد. زیرا در یک سیستم پیچیده، متغیرهای زیادی وجود دارند که در امنیت تاثیر خواهند گذاشت. در نتیجه آسیب‌پذیری‌ها و حملاتی که متوجه سیستم خواهند شد، بیشتر می‌شوند و منابعی که باید مورد محافظت قرار گیرند و همچنین راهکارهای ممکن برای حفاظت از آنها هم بیشتر خواهد شد. بنابراین کار امنسازی سخت‌تر خواهد شد. به عنوان مثال،

نرم‌افزاری که دو میلیون خط کد دارد نسبت به نرم‌فزاری که ۴ هزار خط کد دارد، به احتمال زیاد نقاط آسیب‌پذیر بیشتری خواهد داشت که یافتن همه آنها و رفعشان کار راحتی نخواهد بود.

- **Avoid security by obscurity**: پنهان‌سازی یا همان obscurity الزاما منجر به امنیت نخواهد شد. به عنوان مثال، مخفی کردن کلیدها و رمزها در سورس کد بدون رمزنگاریشان و مثلا با انکد کردن آنها، به معنای حفظ امنیت آنها نخواهد بود. زیرا امکان مهندس معکوس نرم‌افزار و دستیابی به آنها فراهم خواهد شد.
- **Open design**: این اصل به این نکته اشاره می‌کند که امنیت یک سیستم نباید وابسته به پیاده‌سازی آن باشد. به عنوان مثال، یک مکانیزم رمزنگاری درست نباید وابسته به الگوریتم رمزنگاری باشد. زیرا این الگوریتم‌ها معمولا به صورت عمومی منتشر می‌شوند و در صورتی که آسیب‌پذیری برای آنها شناسایی شود و یا از رده خارج شوند، باید به راحتی قابل جایگزین شدن باشند. درواقع امنیت داده‌های رمزشده فقط باید به امن بودن کلید رمزنگاری وابسته باشد.
- **Psychological acceptability**: کاربران یک بخش مهم از سیستم و امنیت آن هستند. بنابراین امنیت به گونه‌ای در سیستم در نظر گرفته شود که توسط کاربران پذیرفته شود. درواقع اگر کاربر احساس کند که امنیت مخل کار او خواهد بود، به راحتی آن را کنار می‌گذارد. بنابراین مکانیزم‌های امنیتی باید برای کاربر کاملا شفاف باشند و به راحتی هم قابل استفاده باشند. درواقع سهولت استفاده و شفافیت، دو موضوع اصلی است که در این اصل به آن پرداخته می‌شود.
- **Weakest Link**: از نظر تئوری تمام سیستم‌ها می‌توانند شامل یک لینک ضعیف یا به عبارت بهتر یک نقطه شکست امنیتی باشند. طبیعتا، مهاجمین در سیستم‌ها سعی می‌کنند این نقاط شکست را بیابند و از طریق آنها به سیستم نفوذ کنند. پس بهتر این نقاط در سیستم شناسایی و بهبود یابند.
- **Leverage existing components**: بهره‌گیری از مولفه‌های موجود مزایای زیادی برای تیم‌ها به ارمغان می‌آورد. از جمله اینکه نیازی به توسعه مولفه‌های جدید وجود نخواهد داشت. در نتیجه احتمال وجود آسیب‌پذیری‌های جدید نیز کاهش خواهد یافت.
- **Least common mechanism**: هر چند در اصل قبلی (Least existing components) به استفاده از توابع موجود اشاره شده است اما در حالتی که این توابع بین کاربران یا فرایندهایی که از سطوح دسترسی مختلفی برخوردار هستند، به اشتراک‌گذاری آنها چندان مناسب نخواهد بود. زیرا ممکن است به صورت ناخواسته منجر به ایجاد یک مسیر برای به اشتراک‌گذاری اطلاعات بین آنها شود.
- **Single point of failure**: در این اصل به این موضوع اشاره می‌شود که باید تمام نقاط شکست سیستم، شناسایی و تحلیل شوند و این اطمینان حاصل شود که هیچ یک از این نقاط منجر به از کار افتادن کل سیستم نخواهد شد.

۲-۲-سرویس‌های امنیتی مشترک

با گسترش نرم‌افزارهای در حال توسعه در تیم‌ها، وجود مولفه‌ها و سرویس‌های امنیتی که یکبار توسعه داده شده و در محصولات مختلف مورداستفاده قرار می‌گیرند، ضروری خواهد شد. درواقع چون بسیاری از کارکردهای امنیتی همانند احرازهویت، مجازشماری، رویدادنگاری و پایش رویدادها، مدیریت کلیدها و فایروالینگ اپلیکیشن‌ها، سرویس‌هایی هستند که معمولا در تمام نرم‌افزارها موردنیاز هستند، بهتر است این کارکردها یکبار توسعه داده شده و در ترکیب معماری نرم‌افزارهای مختلف استفاده شوند. در این صورت نیازی نخواهد بود که به ازای هر نرم‌افزار، این سرویس‌ها به صورت مجزا طراحی، پیاده‌سازی و تست شوند. به عنوان مثال، در این لینک، فهرست سرویس‌های امنیتی AWS آورده شده است.

در سطوح بلوغ بالاتر برای امنیت نرم‌افزار، تیم‌ها می‌توانند با استفاده از سرویس‌های امنیتی موجود، برای موضوعات و حوزه‌های مختلف، انواع معماری‌های مرجع امنیتی[1] ارائه کنند که در آنها ترکیب سرویس‌های امنیتی با دیگر مولفه‌ها برای موضوع یا حوزه موردنظر مشخص شده است. به عنوان مثال، در این لینک می‌توان نمونه‌ای از معماری‌های مرجع امنیتی AWS را مشاهده کرد.

[1] Security Reference Architecture

فصل ۳– معماری امنیتی سادا

همانطور که در فصل قبل به آن اشاره شد، معماری امنیتی یک نرم‌افزار مشخص‌کننده طراحی معماری آن و مجموعه تکنولوژی‌های امن قابل استفاده برای توسعه، استقرار و عملیاتی شدن نرم‌افزار است. در این فصل، معماری امنیتی سادا در این دو حوزه مشخص شده است. در این راستا ابتدا دیاگرام معماری و تشریح اجزای مرتبط با امنیت در آن آورده شده است. فهرست تکنولوژی‌های امن مورداستفاده برای توسعه و استقرار سادا، در یک فایل مجزا ارائه می‌شود.

۳-۱- هدف سند

هدف از تدوین این سند، تشریح معماری امنیتی سیستم سادا است که در آن اجزای مرتبط با امنیت در معماری سادا مشخص و تشریح شده‌اند.

۳-۲- مخاطبین سند

مخاطبین این سند، تیم طراحی، توسعه، استقرار و عملیات سیستم سادا هستند.

۳-۳- دیاگرام معماری امنیتی سادا

دیاگرام معماری امنیتی سادا در یک فایل پاورپوینت با عنوان SADA_Security_Architecture_14020701_v1.0 آورده شده است.

۳-۴- تشریح معماری امنیتی سادا

در این بخش، به شرح معماری امنیتی سادا یعنی توصیف اجزای امنیتی آن پرداخته شده است. این اجزا در راستای فراهم‌سازی کارکردهای امنیتی موردنیاز برای سادا در سطح معماری در نظر گرفته می‌شوند. با توجه به کارکردی که این اجزا فراهم می‌کنند، بر طبق چارچوب امنیت سایبری NIST[1]، می‌توانند در یک یا چند دسته به شرح ذیل قرار گیرند:

- **Identify**: فعالیت‌های این دسته برای شناسایی عملکردها، سرویس‌ها و زیرساخت‌های حیاتی سازمان و ریسک‌های سایبری است که می‌توانند در آنها اختلال ایجاد کنند. شناسایی اهداف، فعالیت‌ها و ذینفعان سازمان، مدیریت دارایی‌های سازمان شامل سیستم‌ها، داده‌ها، دستگاه‌ها، تاسیسات و کارمندان برای انجام عملکردهای حیاتی آن، در این دسته از فعالیت‌ها قرار می‌گیرند.
- **Protect**: فعالیت‌های این دسته دربرگیرنده کنترل‌های امنیتی هستند که برای محافظت از عملکردها، سرویس‌ها و زیرساخت‌های حیاتی باید پیاده‌سازی شوند. شناسایی کاربران و مدیریت دسترسی‌های آنها، آموزش و آگاهی‌رسانی امنیتی و حفاظت از داده‌ها و زیرساخت‌ها جزو این فعالیت‌ها محسوب می‌شوند.

[1] NIST Cyber Security Framework (NIST CSF v1.1)

- **Detect**: در این دسته فعالیت‌هایی قرار می‌گیرند که برای تشخیص و کشف به موقع حوادث امنیتی باید انجام گیرند. ثبت رویدادهای امنیتی و پایش مداوم آنها، از فعالیت‌های این دسته محسوب می‌شوند.
- **Respond**: این دسته دربرگیرنده فعالیت‌هایی است که برای انجام اقدام سریع در شرایط وقوع حملات سایبری، جهت کاهش صدمات ناشی از آنها باید انجام شود. برنامه‌ریزی برای پاسخ‌دهی به حملات برمبنای سطح شدت و نوع آنها، نمونه این فعالیتها است.
- **Recover**: کارکردها و فعالیت‌هایی که در این دسته قرار می‌گیرند برای پشتیبانی از برنامه‌هایی است که جهت تاب‌آوری و بازیابی سرویس‌ها و قابلیت‌های سیستم‌ها به وضعیت نرمال آنها ضروری هستند.

در ادامه توصیف هریک از اجزای امنیتی معماری و دسته‌بندی آنها بر طبق چارچوب CSF آورده شده است.

SE-01: سرویس Identity Provider

توصیف:

مولفه Identity Provider (به اختصار IdP) سیستمی است که برای ایجاد، مدیریت و ذخیره‌سازی متمرکز هویت‌های دیجیتال کاربران سازمان و احرازهویت آنها استفاده می‌شود. با استفاده از این سیستم، دیگر نیازی نیست که کاربر برای استفاده از سرویس‌ها و پلتفرم‌های مختلف موجود در سازمان، credentialهای مجزایی داشته باشد و فقط کافی است در سیستم IdP یک حساب کاربری با credentialهای مشخص داشته باشد. بنابراین امکان اعمال پالیسی‌های قوی برای احرازهویت کاربران همانند احرازهویت چند فاکتوره (MFA[1]) وجود خواهد داشت که در نهایت منجر به بهبود وضعیت امنیت در تمام سرویس‌ها و پلتفرم‌های ادغام شده با این سیستم خواهد شد.

در معماری سادا از سرویس‌های دایرکتوری موجود در سازمان که می‌توانند پیاده‌سازی‌هایی از پروتکل LDAP همانند Microsoft Active Directory (به اختصار AD) یا OpenLDAP باشند، به عنوان سیستم‌های IdP استفاده می‌شود. برای این منظور، سرویس Keystone از ماژول کلون سادا با این سیستم ادغام می‌شود. در این سیستم‌ها، اطلاعات هویتی کاربران سادا اعم از کاربران بهره‌بردار، مدیران ارشد و مدیران سادا ذخیره خواهد شد.

دسته‌بندی: Protect

SE-02: سرویس IAM سیستم سادا

[1] Multi-Factor Authentication

توصیف:

سرویس IAM[1] سیستم سادا که وظیفه مدیریت احرازهویت و کنترل دسترسی کاربران سادا به منابع این سیستم را برعهده دارد، همان سرویسِ Identity سیستمِ OpenStack است که به آن Keystone هم گفته می‌شود. این سرویس احرازهویت کاربران سادا را به کمک سیستم LDAP سازمان انجام می‌دهد و برای مجازشماری آنها از یک چارچوب کنترل دسترسی RBAC[۲] استفاده می‌کند. علاوه بر این، احرازهویت و مجازشماری سرویس‌های سادا همانند سرویس‌های OpenStack، وب‌اپلیکیشن حافظ، rClone و درایور بدافزار هم از طریق Keystone انجام می‌شود. در نتیجه‌ی فرایند احرازهویت و مجازشماری کاربران توسط این سرویس، یک توکن Fernet تولید می‌شود که می‌تواند در سرآیند درخواست‌های ارسالی به سیستم مورداستفاده قرار گیرد.

دسته‌بندی: Protect

SE-03: درایور تحلیل بدافزار (FAM)

توصیف:

درایور تحلیل بدافزار که در سیستم سادا با نام FAM[3] شناخته می‌شود، به عنوان کلاینت سیستم Multi-AV، وظیفه ارسال فایل‌ها به این سیستم برای اسکن آنها از نظر وجود بدافزار برعهده دارد. نتایج دریافتی از سیستم Multi-AV در اختیار سیستم حافظ قرار داده می‌شود. درایور FAM، برای انجام این فعالیت‌ها، APIهای ارائه شده توسط سیستم Multi-AV را که از نوع REST هستند، فراخوانی می‌کند.

دسته‌بندی: Detect

SE-04: سیستم Multi-AV

توصیف:

این سیستم، مجموعه‌ای از چند موتور آنتی‌ویروس است که اقدام به تحلیل و بررسی مخرب بودن فایل‌ها می‌کنند. این سیستم همواره با ارتباط با ریپازیتوری خود، از آخرین الگوهای به‌روزشده برای شناسایی فایل‌های مخرب استفاده می‌کند.

دسته‌بندی: Detect

[1] Identity and Access Management
[2] Role-based Access Control
[3] File Analysis Management

SE-05: سیستم متمرکز مدیریت لاگ‌ها

توصیف:

سیستم متمرکز مدیریت لاگ‌ها، امکان جمع‌آوری، ذخیره‌سازی، پردازش، تحلیل و کنترل دسترسی به لاگ‌های مربوط به مولفه‌های مختلف مستقر شده در محیط عملیاتی یک سامانه را به صورت متمرکز فراهم می‌کند. در محیط عملیاتی سادا از سیستم ELK برای این منظور استفاده می‌شود که یک پشته متشکل از سه مولفه با عناوین Elasticsearch، Logstash و Kibana است. به کمک این سیستم می‌توان وضعیت مولفه‌های مختلف سادا و زیرساخت آنها را به صورت مداوم پایش کرد و براساس معیارهای تعریف شده برای پایش، انواع گزارش‌های بصری تولید کرد. همچنین امکان انجام انواع تحلیل‌های امنیتی نیز به کمک این سیستم فراهم خواهد شد. به عبارت دیگر با استفاده از ELK امکان استقرار یک سیستم SIEM[1] برای سادا فراهم می‌شود.

دسته‌بندی: Detect

SE-06: سرویس Replication

توصیف:

فرایند کپی کردن داده‌ها و ذخیره‌سازی آنها در محل‌های مختلف، Replication نام دارد که با هدف افزایش دسترس‌پذیری داده‌ها، قابلیت اتکاپذیری به داده‌ها و اطمینان از دقت و صحت آنها و همچنین بازیابی سریع از حادثه در صورت وقوع آن انجام می‌شود. سرویس Object Storage از سیستم OpenStack که Swift نیز نامیده می‌شود، خود حاوی چند سرویس دیگر با عنوان سرویس‌های سازگاری[2] است که یکی از آنها سرویس Replicator است. این سرویس با مقایسه داده‌های نود محلی با نسخه‌های کپی آنها در نودهای دیگر (replicaها)، این اطمینان را فراهم می‌کند که نودها شامل آخرین نسخه داده‌ها هستند. برای این منظور از یک لیست hash استفاده می‌کند. بنابراین، در صورتی که یک خطا در شبکه یا دیسک رخ بدهد، امکان جایگزینی نسخه‌های کپی وجود خواهد داشت. replication در سطح اشیاء و همچنین پایگاه داده‌های کیسه و حساب انجام می‌شود.

دسته‌بندی: Recover

SE-07: سرویس Auditor

توصیف:

[1] Security Information and Event Management
[2] Consistency

یکی دیگر از سرویس‌های سازگاری Swift، Auditor نام دارد که با بررسی صحت[1] اشیاء، کیسه‌ها و حساب‌ها در هیولا، وجود هر نوع تخریب و دستکاری در آنها را تشخیص می‌دهد. درصورت مخدوش شدن صحت این موجودیت‌ها، Replicator فایل‌های سالم را از دیگر replicaها جایگزین فایل‌های مخدوش خواهد کرد.

دسته‌بندی: Detect

SE-08: سرویس Updater

توصیف:

در برخی از مواقع ممکن است به‌روزرسانی داده‌های موجود در کیسه و حساب، بلافاصله قابل انجام نباشد؛ مثلا زمانی که بار زیادی روی سرورها باشد. در این حالت، ممکن است مثلا با افزوده شدن یک شیء جدید، سرویس container نتواند بلافاصله پایگاه داده خود را به‌روزرسانی کند. در این حالت، این به‌روزرسانی‌ها در سیستم فایل محلی ذخیره شده و بعدا توسط سرویس Updater مورد پردازش قرار می‌گیرند. سرویس Updater یکی از سرویس‌های سازگاری Swift است.

دسته‌بندی: Recover

SE-09: سیستم مدیریت صحت فایل‌ها (FIM)

توصیف:

سیستم پایش صحت فایل‌ها یا به اختصار FIM[2]، با پایش فایل‌ها و دایرکتوری‌های حساسی همانند فایل‌های سیستم‌عامل، رجیستری‌های ویندوز، نرم‌افزارها، فایل‌های پیکربندی سیستمی و غیره، هر نوع تغییر و دستکاری غیرمجاز در آنها را تشخیص و هشدار می‌دهد. در این سیستم، به کمک مکانیزم hash، یک fingerprint از این فایل‌ها و دایرکتوری‌های حساس، تهیه و به عنوان یک baseline برای مقایسه مورداستفاده قرار می‌گیرد.

فایل‌های پیکربندی سرویس‌های OpenStack، فایل‌های مربوط به certificateها و کلیدهای TLS، دایرکتوری حاوی کلیدهای Fernet و دایرکتوری حاوی اطلاعات instanceها نمونه‌ای موجودیت‌هایی هستند که صحت آنها باید به کمک سیستم FIM مورد پایش قرار گیرد. ابزارهایی همانند OSSEC، Wazuh و Samhain نمونه ابزارهای FIM هستند که در معماری سادا قابل استقرار خواهند بود.

دسته‌بندی: Detect

[1] Integrity
[2] File Integrity Monitoring

SE-10: سیستم متمرکز مدیریت secretها

توصیف:

سیستم متمرکز مدیریت secretها امکان ذخیره‌سازی امن داده‌های حساسی همانند کلمات عبور، کلیدها، certificateها و توکن‌ها را در یک محیط امن و با اعمال کنترل دسترسی فراهم می‌کند. با استفاده از این سیستم‌ها مانع از درج secretها به صورت plaintext در فایل‌های پیکربندی خواهد شد و امکان نگهداری آنها را مستقل از کد فراهم می‌کند. همچنین امکان تعریف پالیسی‌های مختلف برای مدیریت آنها را نیز فراهم می‌کند. به عنوان مثال، اینکه این credentialها چه طولی داشته باشند، از هر چند وقت یکبار تولید مجدد شوند و چه زمان منقضی شوند، در این سیستم قابل تعریف خواهد بود. ضمنا امکان ممیزی دسترسی‌ها به این داده‌ها با تعریف انواع مجوزهای دسترسی به آنها فراهم خواهد شد.

در حال حاضر، سرویس Key Manger از سیستم OpenStack که Barbican هم نام دارد، امکان مدیریت متمرکز secretها را برای سایر سرویس‌های آن فراهم می‌کند. استفاده از سیستم Barbican مزایایی همانند کنترل دسترسی به secretها برمبنای پالیسی‌های تعریف شده، اعمال quota، امکان تنظیم لیست ACL به ازای هر secret، گروه‌بندی secretها و ردیابی استفاده‌کنندگان از secretها را فراهم می‌کند. البته ناگفته نماند که این سرویس می‌تواند با سیستم‌های خارجی همانند HashiCorp Vault، KMIP و دیگر سیستم‌هایی که پلاگین آنها در OpenStack موجود است، نیز ادغام شود. برای این منظور کافی است که در backend، APIهای ارائه شده این سیستم‌ها را فراخوانی کند. البته امکان ذخیره‌سازی secretها در پایگاه‌داده خود سرویس Barbican نیز وجود دارد، اما استفاده از سیستم‌های خارجی امکان ذخیره‌سازی secretها را در یک سیستم کاملا منفک فراهم می‌کند.

دسته‌بندی: Protect

SE-11: مولفه File Digital Signature Creator

توصیف:

با توجه به اینکه فایل‌های کاربران قبل از انتقال از سیستم هیولای محیط نامطمئن به محیط مطمئن باید از دستکاری غیرمجاز محافظت شوند، برای اطمینان از حفظ صحت و اصالت آنها، سمت مبدا آن یعنی محیط نامطمئن، توسط مولفه File Digital Signature Creator، یک امضای دیجیتال برای فایل‌ها تولید و همراه فایل‌ها به مقصد یعنی محیط مطمئن ارسال می‌شود تا در آنجا بررسی شود. برای این منظور، با استفاده از یک کلید خصوصی در محیط نامطمئن، امضای دیجیتال، تولید می‌شود.

دسته‌بندی: Detect

SE-12: مولفه File Digital Signature Verifier

توصیف:

سمت محیط مطمئن قبل از انتقال فایل‌ها به هیولا، امضای دیجیتال فایل‌ها باید مورد بررسی قرار گیرد و فایل‌هایی که صحت آنها مخدوش شده است، به هیولا منتقل نشوند و یک لاگ هم برای آنها ثبت شود. این فرایندها در مولفه File Digital Signature Verifier انجام می‌شود.

دسته‌بندی: Detect

SE-13: Rules Engine

توصیف:

این مولفه تعیین‌کننده تمام قوانینی خواهد بود که جهت اطمینان از آلوده نبودن و سلامت فایل‌ها قبل از انتقال به محیط ذخیره‌سازی مطمئن (هیولای محیط مطمئن)، باید بررسی شوند. این قوانین شامل نتیجه اسکن فایل‌ها توسط Multi-AV، نتیجه بررسی صحت فایل‌ها و هر قانون دیگری است که سازمان تعیین کند. مثلا تعیین نوع و فرمت فایل‌های قابل ذخیره‌سازی در هیولای مطمئن، نمونه‌ای از این قوانین می‌تواند باشد.

دسته‌بندی: Detect

SE-14: Decision Engine

توصیف:

برمبنای قوانینی که در Rules Engine مشخص می‌شود، در مولفه Decision Engine، کار بررسی قوانین و تصمیم‌گیری نهایی برای امکان انتقال یا عدم انتقال فایل‌ها به محیط ذخیره‌سازی مطمئن انجام می‌شود.

دسته‌بندی: Detect

SE-15: مولفه Monster Load Balancer

توصیف:

مولفه Load Balancer امکان توزیع ترافیک بین سرورهای پراکسی هیولا و در نتیجه بهبود کارایی سیستم و افزایش دسترس‌پذیری آن خواهد شد. برای این منظور می‌توان از ابزارهای مختلفی همانند Nginx یا HAProxy به عنوان یک پراکسی معکوس قبل از سرورهای پراکسی هیولا استفاده کرد.

دسته‌بندی: Recover

SE-16: مولفه API Gateway

توصیف:

با توجه به استفاده از معماری Microservices برای استقرار سرویسهای OpenStack در سادا، می‌توان از یک API Gateway همانند پراکسی معکوس، برای انتقال و هدایت درخواست‌ها به میکروسرویس‌های مربوطه استفاده کرد. مزیت استفاده از این الگو آن است که برخی از مکانیزم‌های امنیتی همانند احرازهویت، مجازشماری، محدودسازی نرخ درخواست‌ها[1]، توزیع بار، رویدادنگاری، cache نمودن پاسخ‌های بازگشتی و فیلتر کردن آدرس‌های IP برمبنای یک لیست سفید، به صورت متمرکز قابل انجام است.

دسته‌بندی: Protect

SE-17: سیستم WAF

توصیف:

سیستم WAF[2]، با رصد ترافیک HTTP وارد شده و خارج شده از یک وب‌اپلیکیشن یا وب‌سرویس، ترافیک مخرب را شناسایی و فیلتر می‌کند. با استقرار WAF در سیستم سادا می‌توان ترافیک تمام APIهای وبی آن را مورد بررسی قرار دارد.

دسته‌بندی: Protect

SE-18: قابلیت Image Signature Verification

توصیف:

سرویس Image سیستم OpenStack که Glance نام دارد، قابلیت image signature verification را برای بررسی صحت و اصالت imageهای مورداستفاده برای instanceها برای کاربر و دیگر سرویس‌ها همانند Compute (Nova) فراهم نموده است. در نتیجه اگر imageها هنگام انتقال آنها به سرویس Glance (آپلود آن) و یا دانلود آنها توسط سرویس Compute، دستکاری شوند، توسط مقصد قابل تشخیص خواهند بود. برای این منظور یک امضای دیجیتال برای imageها در مبدا تولید می‌شود و در مقصد این امضا مورد بررسی قرار می‌گیرد.

دسته‌بندی: Detect

[1] Rate Limiting
[2] Web Application Firewall

SE-19: قابلیت سهمیه‌بندی منابع

توصیف:

برای جلوگیری از مصرف تمام ظرفیت منابع محاسباتی، ذخیره‌سازی و شبکه‌ای سیستم توسط یک tenant (پروژه)، امکان سهمیه‌بندی[1] این منابع برای tenantها وجود دارد. این قابلیت در سرویس Image برای تعیین محدودیت روی اندازه imageها و تعداد imageهای قابل آپلود، در سرویس Compute برای تعیین محدودیت روی بسیاری از آیتم‌ها همانند حداکثر تعداد آدرس‌های IP شناور، تعداد هسته‌های cpu، اندازه RAM، تعداد قوانین security group و غیره و در سرویس Object Storage برای تعیین محدودیت برای تعداد یا اندازه اشیاء قابل ذخیره‌سازی در یک کیسه و حساب، قابل پیکربندی است.

دسته‌بندی: Protect

SE-20: قابلیت Security Groups

توصیف:

Security Groupها نقش یک فایروال مجازی را برای instanceها بازی می‌کنند و امکان تعیین قوانین فیلتر آدرس‌های IP را برای instanceهای فراهم می‌کنند. برمبنای این قوانین، دسترسی instanceها در شبکه مشخص و محدود می‌شود. این قابلیت در سرویس‌های Compute و Network (Neutron) قابل پیکربندی است.

دسته‌بندی: Protect

SE-21: لیست Container ACL

توصیف:

صاحب یک حساب در هیولا می‌تواند به کمک لیست‌های ACL[2]، امکان دسترسی به اشیاء و کیسه‌های حساب خود را برای سایر کاربران فراهم کند. این لیست‌ها در متادیتای کیسه ذخیره می‌شوند.

دسته‌بندی: Protect

SE-22: فهرست Roleها

[1] Quota
[2] Access Control List

توصیف:

سرویس Keystone با استفاده از چارچوب کنترل دسترسی RBAC و تخصیص Roleها به کاربران در پروژه‌ها، دسترسی آنها را مدیریت و کنترل می‌کند.

دسته‌بندی: Protect

SE-23: فهرست Policyها

توصیف:

هریک از سرویس‌های OpenStack منابعی دارند که پالیسی‌ها یا قوانین دسترسی به آنها در فایل‌های Policy قابل تعیین است.

دسته‌بندی: Protect

SE-24: توکن‌های Fernet

توصیف:

در نتیجه احرازهویت کاربران توسط سرویس Keystone، یک توکن Fernet تولید می‌شود که از نوع bearer است و شامل حداقل داده‌های لازم برای شناسایی و مجازشماری کاربر است. این داده‌ها به صورت رمزشده و امضاشده در payload این توکن قرار می‌گیرند. توکن‌های Fernet از نوع موقت هستند به این معنا که برای اعتبارسنجی آنها نیازی به ذخیره‌سازی آنها وجود ندارد. همین مزیت، موجب می‌شود که استفاده از این توکن‌ها برای سیستم OpenStack که در آن سرویس‌های مختلف، نیاز به احرازهویت و مجازشماری دارند و ذخیره‌سازی توکن‌ها می‌تواند نیاز به فضای زیادی داشته باشد، مناسب باشد.

دسته‌بندی: Protect

SE-25: مکانیزم TLS

توصیف:

استفاده از پروتکل TLS در ارتباطات بین مولفه‌های مختلف سیستم سادا، از افشا و دستکاری داده‌ها هنگام انتقال در این ارتباطات جلوگیری می‌کند. همچنین کلاینت می‌تواند هویت سرور را با بررسی گواهی TLS مربوطه در CAهای معتبر، احراز کند.

دسته‌بندی: Protect

SE-26: مکانیزم احرازهویت mTLS

توصیف:

در مکانیزم احرازهویت mTLS[1] که به آن، احرازهویت TLS client authentication هم گفته می‌شود، کلاینت هم می‌بایست همانند سرور، گواهی TLS خود را ارائه کند تا سرور بتواند هویت آن را برمبنای این گواهی، بررسی و احراز کند. بکارگیری این مکانیزم برای احرازهویت سرویس‌ها در Keystone، پایگاه داده‌هایشان و RabbtiMQ، یک فاکتور دیگر علاوه بر نام کاربری و کلمه عبور، در فرایند احرازهویت فراهم می‌کند و ریسک دسترسی غیرمجاز را در صورت لو رفتن نام کاربری و کلمه عبور آنها کاهش می‌دهد.

دسته‌بندی: Protect

SE-27: مکانیزم احرازهویت MFA

توصیف:

مکانیزم احرازهویت چند فاکتوره (MFA) به عنوان یک لایه امنیتی دیگر، تهدید دسترسی غیرمجاز به حساب‌های کاربری را کاهش می‌دهد. در این مکانیزم، یک فاکتور دیگر جهت بررسی هویت کاربر قبل از اعطای دسترسی به او در نظر گرفته می‌شود. این فاکتور می‌تواند آن چیزی باشد که کاربر می‌داند (همانند پسورد یا کد PIN)، یا آن چیزی باشد که کاربر دارد (مثلا یک USB Key) و یا یک ویژگی از کاربر باشد (مثلا اثر انگشت کاربر یا تصویر چهره او). در سیستم سادا، احرازهویت کاربران مدیر و مدیر ارشد باید برمبنای مکانیزم MFA انجام شود.

دسته‌بندی: Protect

SE-28: مکانیزم احرازهویت بایومتریک

توصیف:

در مکانیزم احرازهویت بایومتریک برمبنای یک ویژگی فیزیکی از کاربر همانند اثر انگشت، تصویر چهره، تصویر عنبیه، صدا و غیره انجام می‌شود. با توجه به اینکه سرور تغذیه فایل یا همان حافظ، نقطه اصلی ورود فایل‌ها به سیستم هیولا است، بهتر است قبل از اتصال USD Disk به این سرور، هویت کاربر، از طریق مکانیزم بایومتریک، محرز شود.

[1] Mutual Authentication

دسته‌بندی: Protect

SE-29: چارچوب کنترل دسترسی MAC

توصیف:

برای حفاظت از فایل‌ها و دایرکتوری‌های حساس سیستم سادا، همانند فایل‌های پیکربندی، کلیدها و certificateها و همچنین افزایش سطح کنترل دسترسی instanceها به منابع موردنیازشان، باید از چارچوب کنترل دسترسی MAC[1] استفاده شود. در این چارچوب، کنترل دسترسی به objectها (یعنی همان منابع)، برمبنای برچسب‌هایی انجام می‌شود که به این objectها و subjectها (همانند پراسس‌ها یا کاربران) اختصاص داده می‌شود. درواقع این برچسب‌ها، میزان حساسیت اطلاعاتی را که objectها دربرمی‌گیرند، مشخص می‌کنند. هر subject تنها می‌تواند به objectهایی دست یابد که برچسب حساسیت آنها کمتر از برچسب اعطا شده به کاربر باشد.

دسته‌بندی: Protect

SE-30: لاگ‌های Syslog

توصیف:

پروتکل Syslog برای جمع‌آوری و ذخیره‌سازی داده‌های لاگ مربوط به رویدادهای سیستم‌های مبتنی بر یونیکس استفاده می‌شود. از این پروتکل برای ارسال داده‌های لاگ به سرورهای مرکزی نیز می‌توان استفاده کرد. لاگ‌های تولید شده در سیستم هیولا و سیستم‌عامل‌های سادا از نوع Syslog هستند.

دسته‌بندی: Detect

SE-31: لاگ‌های Sela

توصیف:

برای برخی از رویدادهایی که در سیستم سادا رخ می‌دهند، یعنی فعالیت‌هایی که کاربر بهره‌بردار و مدیر سادا در سرور انتقال فایل (حافظ) انجام می‌دهند، همچنین برای رویدادهای مربوط با ارتباط خروجی این سرور با محیط مطمئن و در نهایت رویدادهای مربوط به ارتباط هیولا و ماشین‌های مجازی، می‌بایست لاگ‌هایی از نوع SELA تولید شود.

دسته‌بندی: Detect

[1] Mandatory Access Control

SE-32: لاگ‌های Audit

توصیف:

با استفاده از لاگ‌های Audit، امکان ردیابی فعالیت‌های مختلف کاربران سیستم فراهم خواهد شد. این لاگ‌ها حداقل برای پایگاه داده‌ها و خود سرور مرکزی لاگ (ELK) و همچنین سیستم عامل‌های مورداستفاده در زیرساخت مولفه‌های سادا هم بهتر است تولید شوند.

دسته‌بندی: Detect

SE-33: فایروال شبکه

توصیف:

این فایروال شامل تمام قوانینی خواهد بود که دسترسی به شبکه‌های مدیریتی و داده سادا را محدود و کنترل می‌کنند. شبکه مدیریتی سادا، حوزه‌ای است که در آن سرویس‌ها با یکدیگر و با پایگاه داده‌هایشان و سیستم MQ[1] (در اینجا RabbitMQ) تعامل می‌کنند. درواقع ترافیک این شبکه، مربوط به فرماندهی و کنترل سیستم خواهد بود.

دسته‌بندی: Protect

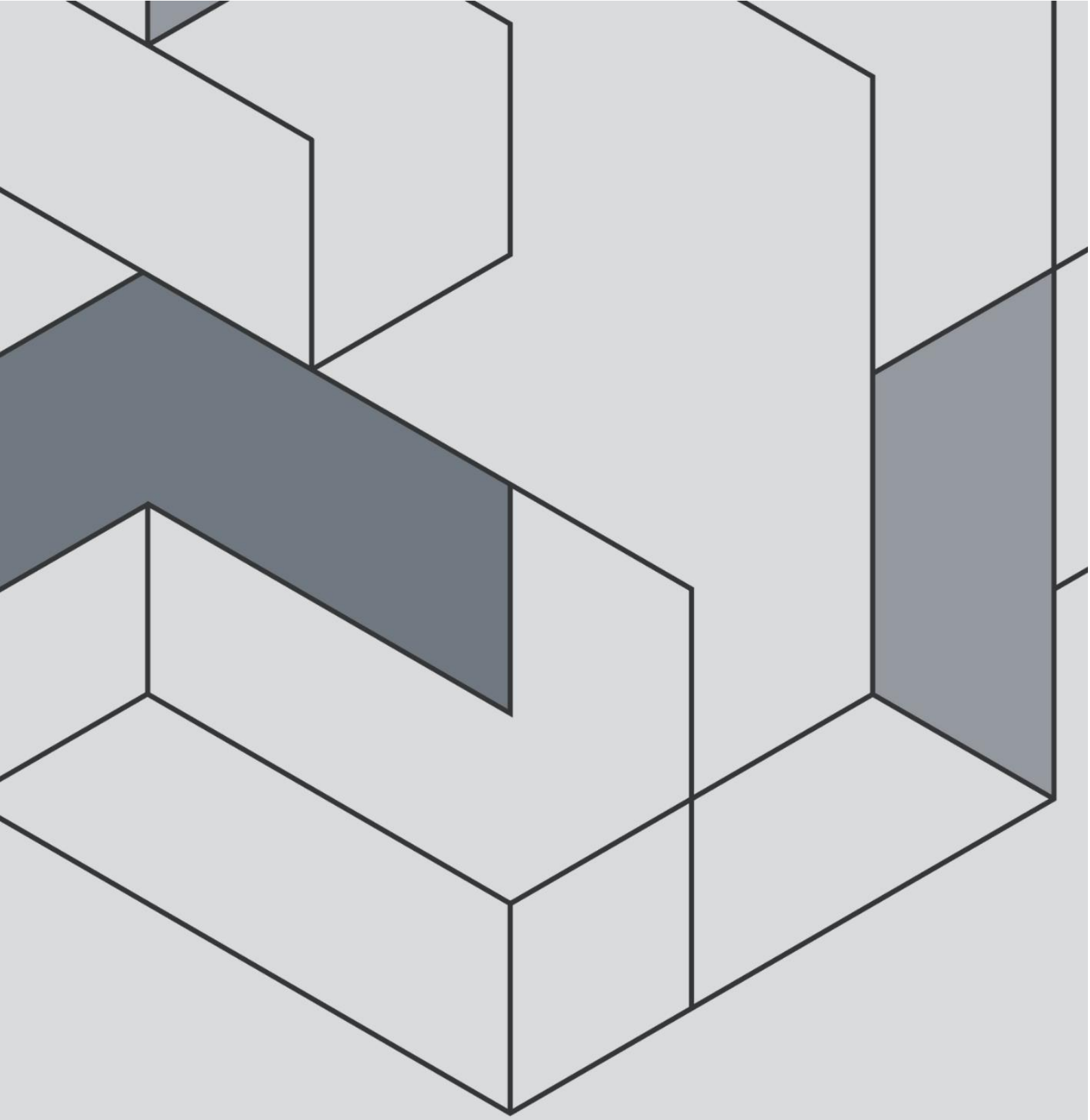
SE-34: مکانیزم Rate Limiting برای APIها

توصیف:

محدودسازی نرخ فراخوانی APIها در یک بازه زمانی مشخص، برای حفاظت از سیستم‌ها در برابر حملات DoS و DDoS مفید است. در سرویس‌های مختلف OpenStack از جمله Compute و Object Storage امکان پیکربندی این قابلیت برای APIهای آنها وجود دارد.

دسته‌بندی: Protect

[1] Message Queuing



www.aminraay.com
info@aminraay.com
+98 21 44899372
+98 24 33411694