مجله‌ی علمی پژوهش در علوم رایانه، شماره ۲۵، بهار ۱۴۰۱، ص ۳۳- ۱۸

ISSN: 2745-3673
http://www.Rcsj.ir

مجله علمی پژوهش در علوم رایانه (سال هفتم)
شماره ۲۵ / بهار ۱۴۰۱ / ص ۳۳- ۱۸

بررسی روش‌های کاربردی تامین امنیت در تبادل اطلاعات سیستم‌های بانکداری همراه

یاسره یوسف‌تبار[۱]، مهدی رضاتبار[۲]

[۱] کارشناس ارشد مهندسی کامپیوتر، نرم افزار، دانشگاه مازندران.

[۲] کارشناس ارشد مهندسی فناوری اطلاعات، شبکه‌های کامپیوتری، دانشگاه مازندران.

نام نویسنده مسئول:
یاسره یوسف‌تبار

تاریخ دریافت: ۱۴۰۰/۱۱/۰۵
تاریخ پذیرش: ۱۴۰۱/۰۱/۲۳

چکیده

در حال حاضر سامانه های اطلاعات مالی، اعتباری و شخصی بطور فزاینده ای از شبکه های مبتنی بر اینترنت در سرتاسر جهان استفاده می‌نمایند. از سوی دیگر، از آنجایی که مسیر گردش اطلاعات و منابع روی شبکه بسیارند، لذا امکان بهره برداری از اطلاعات مذکور توسط اشخاص فراوانی مهیا می‌باشد. بدین ترتیب حفظ امنیت اطلاعات از مباحث مهم تجارت الکترونیک به شمار می آید.

تکنولوژی مالی معاصر که امکان پرداخت توسط تلفن همراه را فراهم می‌کند به طور گسترده توسط موسسات مالی مانند بانک‌ها، به خاطر راحتی و کارایی آن، اتخاذ شده‌است. با این حال، سیستم‌های جدید مستلزم تراکنش‌های عظیم و دینامیک بوده که خطرات امنیتی بالایی را خواهند داشت. با توجه به خسارت‌های مالی بزرگ ناشی از این آسیب‌پذیری‌ها، توسعه امنیت در فن‌آوری مالی حائز اهمیت است.

در پژوهش حاضر پس از بیان اهمیت و ضرورت موضوع امنیت در بانکداری الکترونیکی، مزایای بانکداری مبتنی بر تلفن همراه و خطرات تهدید کننده بانکداری همراه تشریح شده و سپس به تفضیل شیوه های تامین امنیت در تبادل اطلاعات بانکداری همراه شرح داده می شود.

واژگان کلیدی: بانکداری الکترونیکی، بانکداری تلفن همراه، امنیت اطلاعات، آسیب پذیری امنیتی.

مجله‌ی علمی پژوهش در علوم رایانه، شماره ۲۵، بهار ۱۴۰۱، ص ۳۳- ۱۸

ISSN: 2745-3673
http://www.Rcsj.ir

مقدمه

بانکداری الکترونیکی نوع جدیدی از صنعت بانکداری است که خدمات بانکی در آن با استفاده از محیط‌های الکترونیکی صورت می‌گیرد و با به کارگیری فناوری‌های پیشرفته نرم‌افزاری و سخت‌افزاری مبتنی بر شبکه و مخابرات برای تبادل منابع و اطلاعات مالی به صورت الکترونیکی می‌تواند باعث حذف نیاز به حضور فیزیکی مشتری در شعب بانک‌ها شود.

در فن‌آوری مالی معاصر[1] که پرداخت پول الکترونیکی را تشویق می‌کند، خدمات مالی سنتی را به طور قابل توجهی تجزیه کرده‌است، که با اولین دستگاه خودپرداز ATM[2] و در بخش بانکداری الکترونیک به اوج می‌رسد. با این حال، این فناوری چالش‌های جدیدی را برای تنظیم‌کننده‌های مالی ارایه می‌کند، مانند عیوب و آسیب‌پذیری‌های که موجب تلفات عظیم مالی می‌شوند، و در نتیجه مستلزم توسعه موازی فن‌آوری و امنیت اطلاعات است. [۱].

طی چند سال اخیر تجارت با موبایل بانک بعنوان جدیدترین شاخه از تجارت الکترونیکی مورد توجه قرار گرفته و پیشرفت فناوری و نوآوری‌ها، باعث رشد و گسترش کاربردهای آن گشته است به همین دلیل توجه به امنیت در سیستم های موبایل بانک ها بیش از پیش مورد توجه قرار گرفته است. بانک‌ها که مرکز حملات سایبری گسترده‌ای هستند، این حملات منجر به خسارت‌های مالی زیادی به بانک‌ها و مشتریان آن‌ها می‌شود.

یکی از اصلی‌ترین استراتژی‌های مورد استفاده در مقابل حملات سایبری استراتژی دفاع در عمق است. این استراتژی از یک روش دفاعی قدیمی که به روش قلعه نیز معروف است ایده گرفته شده است. درگذشته با ایجاد لایه‌های دفاعی مستقل و پشت سرهم در اطراف قلعه، از دارایی‌های درون قلعه حفاظت می‌کردند. دفاع در عمق کمک می‌کند که حمله را تشخیص و پیشروی آن را به تعویق انداخت و یا مانع پیشروی آن شد [۲].

استفاده از این روش در حملات سایبری زمان بیشتری برای مقابله با حمله برای گروه فناوری اطلاعات سازمان فراهم می‌کند. در بانک‌ها نیز از استراتژی دفاع در عمق برای حفاظت از شبکه داخلی بانک و هر چه که در پشت فایروال وجود دارد استفاده می‌شود. این استراتژی در صورت برقراری سه شرط زیر می‌تواند در مقابله با حملات موفق باشد:

۱. سرمایه‌هایی که می‌خواهیم از آن‌ها محافظت کنیم درون سازمان باشد.
۲. مهاجم خارج از سازمان باشد.
۳. روش‌های دفاعی سازمان برای توقف و مقابله با حملات کافی باشد.

بانکداری الکترونیک

بانکداری الکترونیکی شیوه‌ای از بانکداری است که در آن مشتری بدون حضور فیزیکی در بانک و با استفاده از واسطه‌های ایمن از جمله اینترنت، شبکه‌های ارتباطی بی‌سیم، دستگاه‌های خودپرداز، تلفن و تلفن‌همراه بتواند از خدمات بانکی برخوردار شود و به عبارتی استفاده از تکنولوژی پیشرفته شبکه‌ها و مخابرات جهت انتقال منابع (پول) در سیستم بانکداری می‌باشد. این نوع از بانکداری این امکان را به مشتری می‌دهد که سطح گسترده‌ای از نقل و انتقال وجوه و اطلاعات را به شیوه‌ای الکترونیکی و از طریق وب‌سایت بانک عامل انجام دهد [۳].

از طرفی بانکداری الکترونیک دارای سطوح مختلف می‌باشد. هر چه به سمت سطوح بالاتر یعنی بانکداری الکترونیک کامل، حرکت کنیم عملیات دستی کمتر، سیستم‌های رایانه‌ای متمرکز، قابل دسترسی گسترده‌تر، محدودیت زمانی و مکانی کمتر و در نهایت امنیت اطلاعات بانکی بیشتر خواهد بود.

در یک تعریف جامع تر، بانکداری الکترونیک عبارت است از یکپارچه‌سازی بهینه همه فعالیت‌های بانک از طریق به کارگیری تکنولوژی نوین اطلاعات که امکان ارائه کلیه خدمات مورد نیاز را فراهم می‌سازد.

[1] FinTech: Financial Technology
[2] Automated Teller Machine

ISSN: 2745-3673
http://www.Rcsj.ir

بانکداری مبتنی بر تلفن همراه

روند سریع و رو به رشد اینترنت دارای رقیبی به نام موبایل می باشد. بانکداری مبتنی بر موبایل یکی از جدیدترین دستاوردهای عرصه فناوری در صنعت بانکداری به شمار می‌رود و مزایای زیادی برای بانک‌ها و مشتریان آن‌ها فراهم نموده است [۴].

بانکداری همراه [۳] سامانه‌ای است که از طریق تلفن همراه می‌توان عملیات بانکی خود را انجام داد. در این سامانه با نصب یک نرم‌افزار بر روی گوشی تلفن همراه بدون مراجعه به بانک و در هر ساعتی از شبانه روز می‌توان عملیاتی از قبیل دریافت موجودی حساب، انتقال وجه، پرداخت قبوض و بالانس کردن حساب، تراکنشهای حساب و غیره را انجام داد [۲].

مزایای بانکداری مبتنی بر تلفن همراه

در حال حاضر بهترین راه‌حل برای بهبود امر بانکداری استفاده از فناوری تلفن همراه می‌باشد. از طرفی بانکداری همراه بدلیل عدم استفاده از کابل می‌تواند در مکان هایی که فاقد زیرساخت ارتباطی منظم و منسجم می‌باشند جایگزین تجارت الکترونیکی که به ساختار منظم ارتباطی نیاز دارد، شود [۵]. ارائه خدمات بانکی از طریق تلفن همراه مزایای بسیاری نیز برای بانک ها به همراه خواهد داشت که در ذیل به برخی از آنها اشاره می‌شود:

- کاهش هزینه‌های پرسنلی
- کاهش هزینه‌های مدیریتی
- عدم نیاز به اتصال به اینترنت به روشهای معمول
- ارائه خدمات بانکی در هر نقطه‌ای که مشتری به آن نیازمند باشد
- جلب رضایت مشتریان
- عدم نیاز به تعداد زیادی شعبه در مناطق مختلف
- ارائه خدمات بانکی در ساعات مختلف شبانه‌روز
- سهولت کار کردن و یادگیری نسبت به کامپیوتر

خطرات تهدید کننده بانکداری تلفن همراه

با وجود حملات ویروسی و برنامه های مخرب بر روی تلفن همراه، علاوه بر تخریب عملکرد تلفن همراه، مصرف باطری و حذف اطلاعات تلفن همراه، تهدید بالقوه بانکداری همراه بسیار بیشتر از بانکداری شبکه است. چون ویروس جای گرفته بر ترمینال‌های موبایل ضمن آلوده سازی سیستم عامل، می تواند ترمینال‌های شبکه بی‌سیم را نیز آلوده کند [۶].

عمده ترین خطرات مورد تهدید بانکداری همراه به شرح ذیل می باشد:

- Cloning: هویت تلفن همراه را بر روی دیگری کپی می کند و به هکر اجازه می‌دهد که خود را با هویت قربانی جا بزند (مثلاً از طرف قربانی تماس های صوتی برقرار کند) تا بتواند به حساب مالی قربانی دسترسی داشته باشد.
- Phishing: فرد قربانی را تحریک می کند که اطلاعات مهم شخصی را افشا و یا یک بدافزار[۴] را دانلود کند. اگر از طریق تماس صوتی فرد قربانی را ترغیب به افشا کردن اطلاعات مهم کند به آن Vishing[5] و اگر از طریق پیام کوتاه درخواست جعلی برای افشای اطلاعات مهم فرستاده شود به آن SMishing[۶] گفته می شود.
- Spoofing: فرستادن یک بسته شبکه ای که به نظر می آید از سوی منبع درست آمده است در حالی که واقعیت اینطور نیست.

[3] Mobile Banking
[4] Malware
[5] Voice Phishing
[6] SMS Phishing

ISSN: 2745-3673
http://www.Rcsj.ir

- MITM[7]: در این نوع حمله، حمله کننده با ایجاد اتصال های مستقل با قربانیان، بین دو قربانی (مشتری و بانک) قرار می گیرد و اطلاعات رد و بدل شده بین اینها را استراق سمع کرده و تغییر می دهد و مجدداً ارسال می کند.
- Hijacking: حمله کننده ارتباط بین دو موجودیت[8] را بدست می گیرد و خود را بعنوان یکی از آنها جا می زند و از این طریق می تواند به حساب مالی قربانی دست یابد.
- کدهای مخرب[9]: نرم افزارهایی به شکل ویروس، کرم و بدافزار که معمولاً بصورت مخفی بر روی گوشی، دروازه سرویس پیام کوتاه یا سرور بانک بارگذاری می شوند تا فرایندهای تعیین هویت نشده را که اثر منفی بر روی قابلیت اعتماد، یکپارچگی و دسترسی اطلاعات کاربر دارد، انجام شود.

شیوه های تامین امنیت در تبادل اطلاعات بانکداری همراه

از تجزیه و تحلیل مسائل امنیت اطلاعات بانکداری همراه، می‌توان تفاوت بین مسائل امنیت اطلاعات بانکداری تلفن همراه و بانکداری شبکه را مشاهده کرد. بانکداری همراه با مشکلات امنیتی پیچیده‌تری مواجه است.

نیازهای امنیتی هر سیستم مبادلات الکترونیکی اعم از بانکداری الکترونیک را می‌توان به صورت زیر دسته‌بندی نمود:

۱. محرمانگی[10]: اطلاعات باید برای تمامی اشخاص به غیر از دریافت کننده و فرستنده، غیرقابل دسترس باشد و تضمین گردد که تنها نهادهای مجاز، به محتوای اطلاعات مبادله شده دسترسی دارند. برای مثال، نباید با استراق سمع بتوان تراکنش هایی که یک کاربر خاص انجام داده را یافت.

۲. تصدیق اصالت[11]: هر دو طرف مبادله از اصیل بودن یکدیگر آسوده خاطر باشند. این نیاز امنیتی بیشتر در شیوه اینترنتی بانکداری ملاحظه می‌شود. کاربران باید قبل از ارسال اطلاعات حساس، مطمئن باشند که آنها با بانک واقعی ارتباط برقرار می‌کنند. بانکها باید هویت یک کاربر را قبل از پردازش تراکنش‌های آن بدانند.

۳. صحت داده‌ها[12]: داده های ارسال شده بعنوان بخشی از مبادله نباید طی مراحل ارسال تغییر داده شوند. برای اطلاعات ناقص ایجاد شده در فرایند ارتباطات، سیستم ارتباطات تلفن همراه باید مکانیزم‌های مناسبی برای جلوگیری از وقوع عدم صحت ارائه دهد. کانال‌های حامل، پایانه‌های تلفن همراه، دروازه‌ها[13]، سرورها و سایر تجهیزات بطور مداوم با تهدید حمله ویروس‌ها و سایر حملات مخرب مواجه هستند. سیستم بانکداری تلفن همراه باید مجهز به اقدامات ایمنی مناسبی همچون فایروال‌ها، سیستم تشخیص نفوذ و مکانیزم بازیابی سریع برای تضمین امنیت داده‌ها شود. در روند انتقال داده‌ها، انتقال داده‌های ناقص و شکست رکوردها باید بطور مداوم برای یافتن نقاط ضعف در سیستم مورد بررسی قرار گیرند.

۴. عدم انکار[14]: از انکار تعهدات یا اقدامات قبلی یک نهاد جلوگیری می‌کند طوری که هیچ یک از طرفین مبادله نتوانند مشارکت رد مبادله خود را انکار کنند. بعنوان مثال، یک بانک باید قادر باشد به یک شخص ثالث اثبات کند که یک کاربر تراکنش خاصی را انجام داده، درصورتی که کاربر انجام آن را انکار می‌کند [۲].

برای رسیدن به پیش نیازهای امنیت مبادلات، از روشهای مختلفی همچون رمزنگاری استفاده می‌شود. علاوه بر این، در برخی بخشها همچون تصدیق اصالت، صحت داده‌ها و عدم انکار آن از ابزارهای دیگری مانند امضاء دیجیتال و گواهی‌های کلید عمومی استفاده می‌شود. در ادامه به شیوه‌های تامین امنیت در مبادلات الکترونیکی اشاره می‌شود [۷].

---

[7] Man-in-the-middle attack
[8] Entity
[9] Malicious Code
[10] Confidentiality
[11] Authentication
[12] Data Integrity
[13] Gateway
[14] Non Repudiation

ISSN: 2745-3673
http://www.Rcsj.ir

معتبر سازی Username و Password

اولین قدم برای ورود به یک نرم‌افزار مطمئن و قابل‌اعتماد، ایجاد یک ورودی امن است. یک برنامه‌نویس برای جلوگیری از داده‌ها و ورودی‌های نادرست و مخرب، باید اقدامات امنیتی را در نظر بگیرد. همیشه، هکرها سعی می‌کنند با ورودی‌های نادرست به برنامه شما وارد شوند و یا حتی این امکان وجود دارد که یک کاربر معمولی بدون نیت بد و به صورت خیلی اتفاقی در ورودی داده‌هایی وارد کند که برنامه را مختل کند.

پس اولین کاری که یک برنامه‌نویس حرفه‌ای انجام می‌دهد، قوی کردن ورودی برنامه است. موارد مهمی که در اینجا لازم است بیان شود این است که در کدنویسی های مختلف، روش‌های متعددی نیز برای این کار وجود دارد و کاربر به دلیل اجرای کد برنامه‌نویس نمی‌تواند پرسش از یک ورود امن توسط نرم‌افزار را غیرفعال کند اما در جاوا اسکریپت امکان غیرفعال کردن توسط کاربر وجود دارد و کاربر به راحتی می‌تواند در تنظیمات مرورگرش این کد را غیرفعال کند و برنامه شما عملاً با یک ورودی ناامن باز می‌شود. بهترین راه‌حل در این مواقع این است که علاوه بر نوشتن کد امنیتی ورودی با جاوا اسکریپت، یک‌بار دیگر در محیط برنامه‌نویسی اصلی هم کدهای مربوط به آن با رعایت امنیت کامل قرار داده شود تا ضریب اطمینان امنیتی بالاتری حاصل شود.

حال که صحبت از امنیت ورود اطلاعات شد و به جاوا ختم شد باید بگوییم که جاوا اسکریپت، یک خصوصیت منحصر به فرد برای کد کردن اطلاعات ورودی دارد و آن‌هم هش کردن است که در مبحث بعدی به‌تفصیل آن خواهیم پرداخت.

هش کردن در جاوا اسکریپت

در زبان برنامه‌نویسی اگر یک‌رشته را بتوانیم با استفاده از توابع ریاضی خاص از پیام یا عبارتی به دست بیاوریم، آن عبارت را هش[15] کرده‌ایم و به این تابع ریاضی، تابع هش رمزنگاری[16] می‌گوییم. این توابع باید دارای خصوصیات منحصربه‌فردی باشند. مثلاً این‌که نباید وارون‌پذیر باشند یعنی نتوان دوباره آن پیام اولیه را تولید کرد و دیگر این‌که اگر یک پیام، تغییری هرچند کوچک داشته باشد باید هش به دست آمده از آن تفاوت خیلی زیادی با پیام اولی داشته باشد.

مورد دیگر این است که هیچ‌گاه نباید دو پیام متفاوت، خروجی یا هش یکسانی داشته باشند. این را هم بهتر است بدانیم که برخی از توابع ریاضی هستند که وقتی دو بار یک پیام یکسان را می‌گیرند، در خروجی هش های متفاوتی را تولید می‌کنند که این مورد نیز نباید وجود داشته باشد و با هر بار هش گرفتن از یک عبارت، باید خروجی یکسانی تولید شود.

یکی از خصوصیاتی که باید در تابع‌های هش وجود داشته باشد، کند بودن آن‌هاست زیرا اگر دارای الگوریتم سریعی باشند، امکان حملات را بیشتر می‌کنند و در این نوع حملات، مهاجم‌ها یک رمز عبور را هش می‌کنند و با مقایسه آن با تعداد بسیار زیادی رمز عبور که از قبل و به‌صورت تصادفی آماده دارند، رمز عبور ما را حدس می‌زنند؛ پس ویژگی کند بودن الگوریتم لازم و ضروری است.

موارد ذکرشده در بالا به این علت مهم و حیاتی است که هکرها نتوانند با استفاده از مهندسی معکوس و به دست آوردن تابع هش، پیام اصلی را حدس بزنند.

یکی از توابع معروف هش کردن، الگوریتم MD5[17] است. این الگوریتم، ازنظر محاسبه، بسیار راحت و نیز یک الگوریتم سریع است اما ایرادی که به آن وارد است این است که در برخی مواقع ممکن است که این تابع، با گرفتن دو پیام یا عبارت مختلف، یک کد خروجی تولید کند و این ایراد سبب شده که این الگوریتم قدیمی دیگر محبوبیت نداشته باشد و با توابع و الگوریتم‌های بهینه‌تر جایگزین شده است.

[15] Hash
[16] Hash Function Cryptographic
[17] Message-Digest Algorithm

ISSN: 2745-3673
http://www.Rcsj.ir

تابع SHA-512[18] هم یک کلید طولانی را برای هش کردن تولید می‌کند و به جرأت می‌توان گفت که طولانی‌ترین کد هش شده را در بین توابع موردنظر در جاوا تولید می‌کند. البته در حال حاضر نسخه‌های جدیدتر آن هم عرضه شده است اما هیچ‌کدام ازنظر امنیتی به‌پای آن نمی‌رسند و در ادامه برخی از کدهای آن آورده شده است.

```
salt <- generate-salt;
hash <- salt + ':' + sha512(salt + password)
SecureRandom random = new SecureRandom();
byte[] salt = new byte[16];
random.nextBytes(salt);
MessageDigest md = MessageDigest.getInstance("SHA-512");
md.update(salt);
byte[] hashedPassword = md.digest(passwordToHash.getBytes(StandardCharsets.UTF_8));
```

کدهای فوق که یک کد امنیتی در جاوا می باشد، یک دنباله تصادفی که با Salt عنوان می‌شود را برای هش‌های جدید ایجاد می‌کند و این دنباله که به‌کاربرده می‌شود، میزان تصادفی بودن هش را بالا می‌برد و موجب ایمن‌تر شدن دیتابیس یا بانک اطلاعاتی ما می‌شود. زمانی که از Salt هم استفاده می‌کنیم باید بدانیم که این الگوریتم، یک الگوریتم قوی نیست و در حد متوسط است.

در واقع یکی از ویژگی‌های مثبت یک تابع هش این است که الگوریتم کندی داشته باشد که این کندی می‌تواند بسیار در بهینه بودن یک الگوریتم مؤثر باشد و یک امتیاز مثبت برای آن به‌حساب بیاید.

الگوریتم‌های قوی

دو الگوریتم شرح داده شده در بخش قبل با پیشرفت‌های کدنویسی و امنیتی برنامه‌های کنونی ناسازگار است و دیگر استفاده نمی‌شود زیرا دیگر نمی‌تواند امنیت برنامه را تأمین کند. یکی از الگوریتم‌های قوی، PBKDF2[19] است که در ادامه به توضیح آن و آوردن چند خط کد در باب آن می‌پردازیم:

مفهوم Salt که در قسمت قبل ذکرشده است، در همه الگوریتم‌های هش کردن وجود دارد. در ادامه، کدنویسی این الگوریتم نشان داده شده است:

```
SecureRandom random = new SecureRandom();
byte[] salt = new byte[16];
random.nextBytes(salt);
KeySpec spec = new PBEKeySpec(password.toCharArray(), salt, 65536, 128);
SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
byte[] hash = factory.generateSecret(spec).getEncoded();
```

کدهای فوق، الگوریتم ذکرشده را پیاده‌سازی می‌کند و هش را با این روش تولید می‌کند.

کتابخانه Spring Security موجود در جاوا، الگوریتم‌های هش را پشتیبانی می‌کند و انکودرهای رمز عبور را برای این الگوریتم‌ها فراهم می‌نماید.

زبان برنامه‌نویسی برای نوشتن نرم‌افزار بانکداری

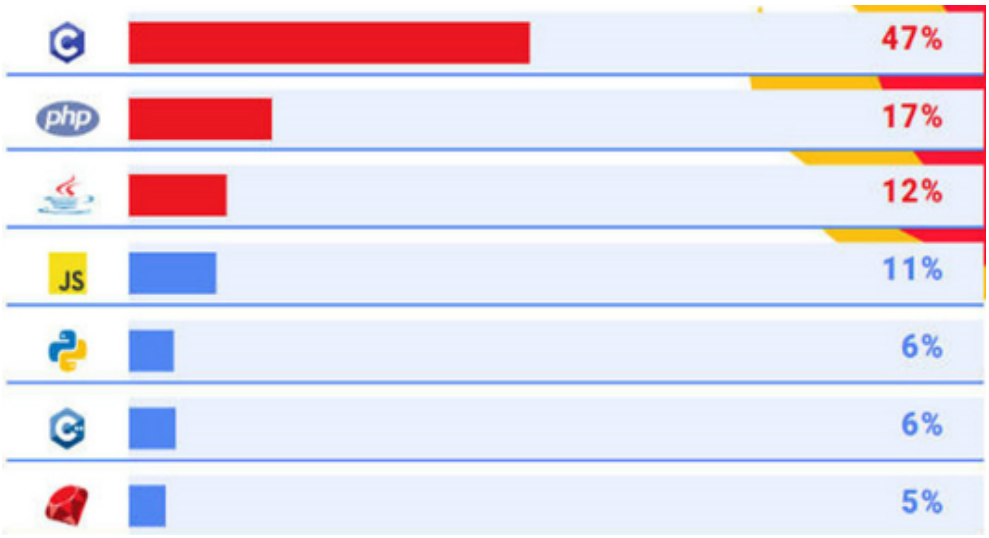
باید به این نکته توجه کنیم که نقص امنیتی در همه جای دنیا و نه فقط در کشور ما و نیز در همه سازمان‌ها و نهادها و نه‌تنها در بانک‌ها وجود دارد، حالا بنا بر عللی از جمله سروکار داشتن با محاسبات فراوان و نیز نقدینگی و سرمایه و پول مردم در بانک‌ها، این امنیت جلوه و اهمیت بیشتری به خود می‌گیرد و از آنجا که هکرها و نیز برنامه‌های مخرب زیادی برای نفوذ به این

[18] Secure Hash Algorithm
[19] Password-Based Key Derivation Function

ISSN: 2745-3673
http://www.Rcsj.ir

برنامه‌ها ساخته می‌شود و حمله‌های بیشتری به این نرم‌افزارها انجام می‌شود، اهمیت بسیار بیشتری نسبت به مثلاً یک نرم‌افزار دبیرخانه دارد چراکه شمار حمله‌ها و احتمال آن به اپلیکیشن هایی که با پول سروکار دارند بیشتر است.

طبق آخرین آمارها از وایت سورس، زبانهای ناامنی که بیشترین آمار هک و حمله را به خود اختصاص داده‌اند، زبانهای سی[۲۰]، جاوا[۲۱]، جاوا اسکریپت[۲۲]، پایتون[۲۳]، روبی[۲۴]، پی اچ پی[۲۵] و سی پلاس پلاس[۲۶] هستند که تا آنجایی که می‌دانیم زبان پایتون یکی از زبان‌های مهمی هست که امروزه در دانشگاه‌های کشورمان متداول است و استادان از دانشجویان می‌خواهند که پروژه‌های درسی‌شان را با آن انجام دهند و کسی به این نکته توجه نمی‌کند که امنیت پایینی دارد و همین‌طور PHP و جاوا که خیلی از برنامه نویسان با آن کار می‌کنند. شکل ۱ میزان حمله و گزارش‌های امنیتی در مورد زبان‌های مختلف که ذکر شد را رتبه‌بندی کرده است.



شکل ۱. رتبه بندی میزان حمله و گزارش‌های امنیتی زبان‌های مختلف

رتبه اول به زبان سی تعلق دارد که بیشترین آموزش را در دانشگاه‌های کشور ما دارد و در دروس دانشگاهی هیچ صحبتی از امنیت این زبان و یا حداقل راهکارهایی برای کد نویسی امن‌تر نشده است و فقط به آموزش دستورالعمل‌ها و ارتباط با پایگاه داده بسنده شده است و حتی در ارتباط با بانک اطلاعاتی هم که باید مسائل امنیتی مطرح شود خیلی سربسته و جزئی گذری زده‌شده است. همان‌گونه که مشاهده می‌شود، ۴۷٪ از گزارش حملات مربوط به این زبان است یعنی چیزی نزدیک به نیمی از حملات صورت گرفته به برنامه‌هایی بوده که با این زبان نوشته شده است.

زبان پی اچ پی در رتبه دوم حمله‌ها قرار دارد که آن هم بازارگرمی در بین برنامه نویسان دارد و ۱۷٪ گزارش‌ها مربوط به آن است. در رتبه سوم این رده‌بندی، زبان جاوا است که با ۱۲٪ حمله‌ها و گزارش‌های امنیتی می‌توان گفت که نفوذپذیری بالایی دارد. زبان‌های جاوا اسکریپت و پایتون و سی پلاس پلاس و روبی هم رتبه‌های بعدی را به خود اختصاص داده‌اند.

حالا اگر واقع‌بینانه به مسئله نگاه کنیم می‌بینیم که نمی‌توان همه انگشت‌ها را به سمت زبان سی نشانه گرفت زیرا اولاً به خاطر این‌که زبان سی قدیمی‌تر از سایر زبان‌هاست و تقریباً جزء اولین زبان‌های سطح بالایی است که وارد بازار شده و سالیان متمادی مورد استفاده و علاقه برنامه نویسان بوده و زبان گرامری آن به زبان محاوره نزدیک‌تر است؛ پس می‌توانیم این نتیجه را

[20] C
[21] Java
[22] Java Script
[23] Python
[24] Ruby
[25] PHP: Personal Home Page
[26] C++

ISSN: 2745-3673
http://www.Rcsj.ir

بگیریم که برنامه‌های بسیاری با این زبان نوشته‌شده است و در مقایسه با زبان‌های دیگر پرکاربردتر است و طبیعی است که بیشترین تعداد حملات نسبت به آن ثبت می‌شود؛ چرا که بیشترین تعداد برنامه‌ها هم با آن نوشته شده است.

اما این نکته را هم باید توجه داشته باشیم که در بسیاری از دستورات این زبان برنامه‌نویسی، پاسخ‌های مناسبی در مقابل درخواست‌های مختلف در شرایط گوناگون تعریف نشده است و همین امر موجب می‌شود که هکرها با علم به این باگ‌ها، دستورالعمل‌های خاصی را به کار ببرند و برنامه را مختل کنند و به هدف خود یعنی آسیب رساندن به نرم‌افزار برسند. نکته دیگری که باید در اینجا ذکر شود این است که بیشترین نقص امنیتی مربوط به اعتبارسنجی داده‌هاست که تقریباً چالشی برای همه زبان‌های برنامه‌نویسی است.

این‌که ما با تحلیل و تجزیه و نمودار به این نتیجه برسیم که کدام زبان برنامه‌نویسی، امنیت بالاتر یا پایین‌تری دارد نمی‌تواند دغدغه‌مان برای کدنویسی نرم‌افزارهای کاربردی بانکی باشد زیرا آن چیزی که مهم است، این است که برنامه‌نویس باید بامطالعه و تجربه فراوان، به آبدیدگی برسد و نکات امنیتی را در برنامه‌اش به کار ببرد و برنامه را طوری طراحی کند و توسعه دهد که هیچ رخنه نرم‌افزاری ممکن نباشد، حالا با هر زبانی باشد.

زبان‌های سطح پایین

زبان‌های سطح پایین یا زبان ماشین[۲۷]، زبان‌هایی هستند که مستقیماً دستورات را به پردازنده صادر می‌کنند و یا از آن می‌گیرند و این زبان‌ها برای دسترسی به پردازنده نیاز به مترجم یا کامپایلر[۲۸] ندارند. اگرچه فهمیدن آن‌ها برای پردازنده آسان است اما برای ما و حتی برنامه نویسان قوی هم این کار دشوار و زمان‌بر است و با وجود اینکه زبان‌های سطح بالا دستوراتی شبیه زبان انگلیسی دارند اما در این زبان‌ها، اغلب از صفر و یک برای کد نویسی استفاده می‌شود و پیچیدگی خاص خود را دارد. به‌عنوان‌مثال، زبان اسمبلی یکی از زبان‌های متداول و محبوب سطح پایین است. بیشتر زبان‌های سطح بالا هم زمانی که ترجمه یا کامپایل می‌شوند به‌صورت یک زبان سطح پایین یعنی صفر و یک درمی‌آیند و بعد که فهمشان توسط پردازنده به این وسیله راحت‌تر شد پردازش می‌شوند.

نکته‌ای که از اینجا برداشت می‌شود این است که اگرچه برنامه‌های سطح پایین، بسیار دشوارتر از نوع سطح بالا هستند اما این قابلیت را دارند که امنیت را برای سیستم‌ها به ارمغان بیاورند به دو دلیل:

۱- چون دستورالعمل‌های طولانی و سختی دارد پس نفوذ به آن هم نیازمند یادگیری این دستورالعمل‌هاست و این از عهده هرکسی برنمی‌آید.

۲- چون این زبان‌ها ارتباطی بی‌واسطه با پردازنده دارند و از کامپایلر استفاده نمی‌کنند یکی از راه‌های نفوذ به آن‌ها بسته می‌شود و این بی‌واسطه بودن در برقراری اتصال با پردازشگر یکی از امتیازات امنیتی این نرم‌افزارها محسوب می‌شود.

پس می‌توان نتیجه گرفت که اگر یک نرم‌افزار بانکداری با زبان سطح پایین نوشته شود، از امنیت بیشتری برخوردار است و دو دلیل فوق نیز این مورد را تائید می‌کند.

رمزنگاری

یک سیستم رمزنگاری[۲۹]، شامل روشی برای تغییر شکل پیام‌ها است که افراد مشخصی بتوانند محتوای آنها را درک کنند. به تبدیل یک متن یا پیام به پیام تغییر شل یافته رمزنگاری یا رمزگذاری[۳۰] گفته می‌شود. به برگرداندن پیام تغییر شکل داده شده به پیام اصلی، رمزگشایی[۳۱] گفته می‌شود. رمزنگاری طوری انجام می‌شود که متن اصلی درهم می‌شود. بطورکلی رمزنگاری به دو دسته رمزنگاری بدون کلید و رمزنگاری با کلید تقسیم می‌گردد [۷].

---

[27] Machine Language
[28] Compiler
[29] Cryptophytic
[30] Encryption
[31] Decryption

ISSN: 2745-3673
http://www.Rcsj.ir

۱- رمزنگاری کلید متقارن: رمزنگاری کلید متقارن[۳۲] یا تک کلیدی، آن دسته از الگوریتم‌ها، پروتکل‌ها و سیستم‌های رمزنگاری است که در آن هر دو طرف فرستنده و گیرنده اطلاعات از یک کلید رمز یکسان برای عملیات رمزنگاری و رمزگشایی استفاده می‌کنند. در این قبیل سیستم‌ها، کلیدهای رمزگذاری و رمزگشایی یکسان هستند و یا با رابطه‌ای بسیارساده از یکدیگر قابل استخراج هستند و رمزگذاری و رمزگشایی اطلاعات نیز دو فرایند معکوس یکدیگر هستند. در این رمزنگاری باید یک کلید رمز مشترک بین دو طرف تعریف شود. چون کلید رمز باید کاملا محرمانه باقی بماند، برای ایجاد و مبادله رمز مشترک باید از کانال امن استفاده نمود یا از روش‌های رمزنگاری نامتقارن استفاده کرد. نیاز به وجود یک کلید رمز به ازای هر دو نفر شامل در رمزنگاری متقارن، موجب بروز مشکلاتی در مدیریت کلیدهای رمز می‌گردد. اکثر کارتهای بانکی ازطریق این روش تبادل اطلاعات می‌کنند. یعنی پایانه‌های خدمات بانکداری الکترونیک با استفاده از رمز عبور کارت و اطلاعات کارت، پیغام رمز شده‌ای را برای بانک یا موسسه خدمات دهنده ارسال می کند و آن موسسه با استفاده از رمز کارت که در بانک اطلاعاتی خود ذخیره کرده، پیغام دریافتی مشتری را رمزگشایی کرده و درصورت صحت اطلاعات ارسالی با مشخصات ذخیره شده در بانک اطلاعاتی، خدمات درخواستی مشتری به او تحویل داده می‌شود. تبادل اطلاعات بین پایانه‌ها و بانک اطلاعاتی بصورت رمزگذاری شده صورت می‌گیرد اما از روش نامتقارن استفاده می‌شود [۷].

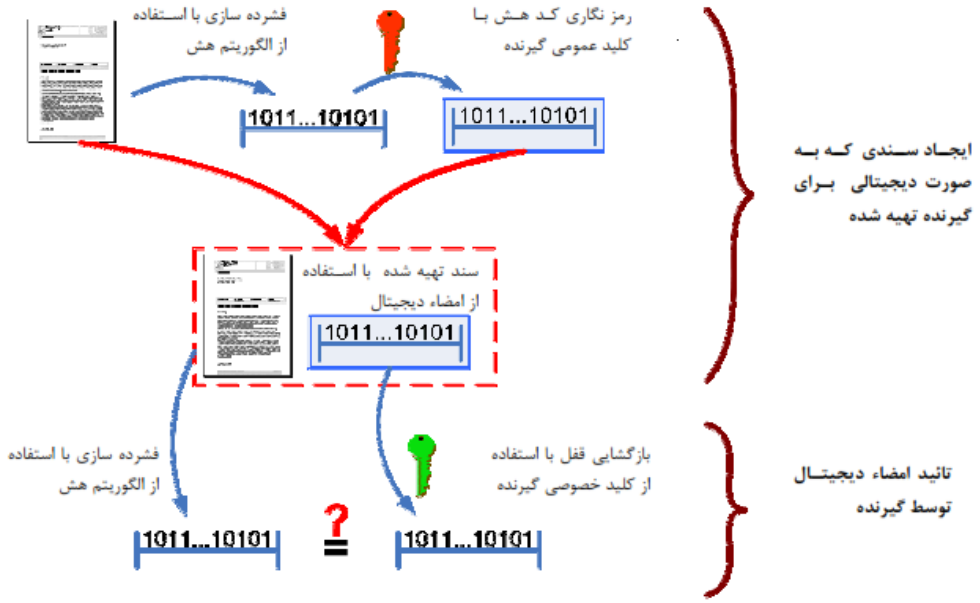
۲- رمزنگاری کلید نامتقارن: رمزنگاری کلید نامتقارن[۳۳]، در ابتدا با هدف حل مشکل انتقال کلید در روش متقارن و در قالب پروتکل تبادل کلید دیفی-هلمن[۳۴] پیشنهاد شد. در این نوع از رمزنگاری، به جای یک کلید مشترک، از یک زوج کلید به نام‌های کلید عمومی[۳۵] و کلید خصوصی[۳۶] استفاده می‌شود. کلید خصوصی تنها در اختیار دارنده آن قرار دارد و امنیت رمزنگاری به محرمانگی کلید خصوصی بستگی دارد. دو کلید خصوصی و عمومی با یکدیگر متفاوت هستند و با استفاده از روابط خاص ریاضی محاسبه می‌شوند. رابطه ریاضی بین این دو کلید به گونه‌ای است که کشف کلید خصوصی با در اختیار داشتن کلید عمومی، عملاً ناممکن است. امروزه در امضاء دیجیتال از این شیوه استفاده می‌شود [۷].

۳- امضاء دیجیتال: امضاء دیجیتال، داده الکترونیکی است که یا به سایر داده‌ها متصل است یا به لحاظ منطقی با آنها مرتبط است و روشی برای احراز هویت به شمار می‌رود. امضاء دیجیتال با دو روش رمزنگاری متقارن و رمزگذاری نامتقارن پیاده سازی شده است. اما به مرور زمان، استفاده از رمزنگاری متقارن در امضاء دیجیتال منسوخ شد زیرا قبل از ارسال پیام باید هر دو طرف مبادله به طریقی کلید مشترک خصوصی را بدست آورند. این مسئله کاربرد امضاء دیجیتالی که به این روش ایجاد شده را در شبکه ارتباطات باز همانند اینترنت با مشکل جدی مواجه می‌کند، چرا که طرفهای مبادله هیچ ارتباط قبلی با یکدیگر ندارند و ممکن است ارتباطات جعلی صورت گیرد [۷].

برای حل این مشکل، شیوه رمزنگاری نامتقارن یا همان کلید عمومی استفاده می‌شود. در این شیوه، هر شخص یک جفت کلید به نامهای کلید خصوصی و کلید عمومی در اختیار دارد. کلید عمومی بصورت عمومی منتشر شده و در اختیار افراد مختلف مبادله کننده قرار می‌گیرد، درحالی که کلید خصوصی هرگز ارسال نمی‌شود و به اشتراک گذاشته نمی‌شود. بعنوان مثال، هنگامی که مشتری بانک تمایل دارد پیامی را به بانک ارسال کند، ابتدا کلید عمومی بانک عامل را از بین کلیدهای عمومی استخراج نموده و با استفاده از آن، پیام را به بانک عامل ارسال می‌کند. بانک عامل با استفاده از کلید خصوصی خود پیام را از حالت رمزگذاری شده خارج می‌کند و به محتوای اصلی پیام دست می‌یابد. بنابراین، در چنین روشی، هر فردی می‌تواند با استفاده از کلید عمومی بانک، پیام خود را ارسال نماید و تنها بانک عامل می‌تواند با استفاده از کلید خصوصی خود، به پیام ارسال شده دسترسی داشته باشد [۷]. روند داخلی امضای دیجیتال در شکل ۲ نشان داده شده است.

[32] Syemmetric-Key Cryptography
[33] Asymmetric-Key Cryptography
[34] Diffie-Hellman key exchange
[35] Public Key
[36] Private Key

ISSN: 2745-3673
http://www.Rcsj.ir



شکل ۲. روند داخلی امضای دیجیتال

لایه سوکتهای امن و HTTP امن

لایه سوکت امن[۳۷] توسط کارگروه مهندسی اینترنت[۳۸] برای پروتکل امنیت لایه انتقال[۳۹] اتخاذ شد. بسیاری از تولیدکنندگان بزرگ محصولات اینترنتی توافق کرده‌اند که از پروتکل رمزنگاری لایه سوکتهای امن استفاده کنند و TLS برای ایجاد یک معادل بی‌سیم، امنیت لایه انتقال بی‌سیم[۴۰] تطبیق داده شد. گرچه تفاوتهایی بین نسخه‌های مختلف این پروتکل‌ها وجود دارد، اما بطور مفهومی آنها سرویس امنیتی مشابهی را بصورت یک کانال امن بین کلاینت و بانک ارائه می‌دهند [۲].

برای رمزنگاری داده‌هایی که از طریق اتصال SSL ارسال می‌شوند از یک کلید خصوصی استفاده می‌شود. بسیاری از وب سایتها از این پروتکل برای گرفتن اطلاعات محرمانه از کاربر مانند کارتهای اعتباری استفاده می‌کنند. این پروتکل که بین لایه‌های پروتکل سطح کاربرد[۴۱]، مانند HTTP[42] و پروتکل لایه انتقال[۴۳] یعنی TCP/IP[۴۴] قرار می‌گیرد، برای جلوگیری از استراق سمع، تحریف و جعل پیغام طراحی شده است. چون SSL در زیر پروتکل لایه کاربرد قرار می‌گیرد، می‌تواند برای سایر پروتکل‌های لایه کاربرد مانند FTP[45] نیز بکار رود [۸].

پروتکل مورد استفاده دیگر برای ارسال ایمن داده‌ها روی وب، پروتکل S-HTTP[۴۶] است که نسخه تغییر یافته و امن شده‌ای از پروتکل HTTP استاندارد است. همان گونه که SSL یک اتصال مطمئن بین یک مشتری[۴۷] و یک سرور ایجاد می‌کند و هر مقدار داده می‌تواند بطور امن منتقل شود، S-HTTP هم طوری طراحی شده که پیغام‌های جداگانه را بصورت ایمن ارسال می‌کند. بنابراین از لحاظ تکنولوژی، SSL و S-HTTP مکمل یکدیگرند [۹].

---

[37] SSL: Secure Sockets Layer
[38] IETF: Internet Engineering Task Force
[39] TLS: Transport Layer Security
[40] WTLS: Wireless Transport Layer Security
[41] Application Layer
[42] Hyper-Text Transfer Protocol
[43] Transport Layer
[44] Transmission Control Protocol / Internet Protocol
[45] File Transfer Protocol
[46] Secure-HTTP
[47] Client

ISSN: 2745-3673
http://www.Rcsj.ir

حمله‌های XSS[48]

در بعضی مواقع، مهاجمان برای نفوذ به برنامه، کدهای جاوا اسکریپت نوشته شده خودشان را که اغلب تخریب کننده است، به صفحات آسیب پذیر سایت ها تزریق می‌کنند یعنی به اصطلاح، آنها را لابلای کد صفحات، جاگذاری می‌کنند. از جمله دستورالعمل‌هایی که می‌توانند بنویسند و در صفحات قرار دهند این است که داده های وارد شده توسط کاربر را بدون اعتبار سنجی بپذیرند.

مثلاً ممکن است وقتی کاربر، یک صفحه را باز می کند و روی یک لینک کلیک می کند و یا کار دیگری مثل باز کردن ایمیل را انجام می دهد، این کد نوشته شده توسط هکرها به صورت کاملاً مخفیانه بر روی سیستم او اجرا شود و کوکی ها یا اطلاعات مهم سیستم و صفحه باز شده را به سرقت ببرند که این در یک نرم افزار بانکی می تواند اطلاعات مربوط به حساب های بانکی و رمز های کارت و رمز های اینترنتی باشد که بسیار پر خطر است و هکر می‌تواند با این کار به حساب های کاربر دسترسی داشته باشد و آنها را سرقت کند.

این حمله ها ممکن است بصورت ایمیلی توسط کاربر دریافت شده باشد که این کد مخرب در آن ایمیل جاسازی شده باشد و با باز کردن آن، در سیستم او ذخیره و اجرا[49] شود، و یا این که سایت طراحی شده توسط بانک یا هر جای دیگر در خود یک حفره امنیتی داشته باشد. این امکان هم وجود دارد که لینکی که در برگیرنده یک کد XSS است را کاربر باز کرده باشد. در مواقعی هم ممکن است کسی که سایت را طراحی کرده، کد مخرب را در صفحات مهم قرار داده باشد که از این نظر هم سازمان‌های بزرگ و علی الخصوص بانک ها باید دقت کافی را داشته باشند که چه کسی یا گروهی سایت سازمانی شان را طراحی می کند؟

علاوه بر موارد ذکر شده فوق، می توانیم این فرضیه را هم مورد توجه قرار بدهیم که این کدها ممکن است در سیستم عامل و یا حتی فراتر از آن در سطح شبکه وجود داشته باشد که با درگیر شدن یک کامپیوتر، بقیه رایانه‌ها و سیستم ها نیزآلوده می شوند و این کد تخریب‌گر در همه سیستم‌ها اجرا می شود.

یکی از راهکارهای مقابله با این حمله‌ها استفاده مرورگرهایی با امنیت بالا است. IE[50] که به صورت پیش فرض با سیستم عامل ویندوز نصب می شود بسیار امنیت پایینی دارد و وجود حفره امنیتی در آن هم ثابت شده است و از آن جایی که با نصب ویندوز، به صورت خودکار بر روی دسک تاپ اعلام حضور می کند، بسیاری از کاربران آن را مورد استفاده قرار می دهند و به مرورگر دیگری فکر نمی‌کنند که این کار اشتباهی است علی الخصوص در باز کردن نرم افزارهای تحت وب بانک ها باید این مورد را مد نظر قرار داد که با امنیت بالا به این کار مبادرت ورزید.

از ابرازهایی که برای جلوگیری از عمل کردن اسکریپت های بیگانه و هر نوع اسکریپتی در نظر گرفته شده و کدهای فلش و اسکریپت را نادیده می گیرد، ابزار Noscript است که لازم است به کار برده شود.

برخی از کاربران به علت این که فراموش کار هستند و یا این که نمی خواهند از حافظه شان برای حفظ کردن رمز عبور و نام کاربری استفاده کنند، قابلیت یادآوری آن توسط مرورگر را در آن فعال می کنند که این کار، بسیار نکوهش شده است و ریسک بالایی دارد و بهتر است که علاوه بر انجام ندادن این کار، به صورت دوره ای این اطلاعات را تغییر دهند.

راهکارهای زیاد دیگری هم هست که در این پژوهش نمی گنجد، اما باید این نکته مهم را در نظر بگیریم که حمله کننده ها همیشه به دنبال راهی هستند که به وب سایت مورد استفاده کاربران نفوذ کنند و هر طوری که هست کدهای مخرب را در آن قرار دهند، یک نمونه می‌تواند این باشد که کاربران را به بازدید از یک صفحه خاص تشویق می‌کنند که همان طور که می توان حدس زد، در آن صفحه، کدهای مخرب وجود دارد؛ مثلاً این کد را در نظر بگیرید:

[48] Cross Site Scripting
[49] Run
[50] Internet Explorer

ISSN: 2745-3673
http://www.Rcsj.ir

```
print "<html>"
print "<h1>Most recent comment</h1>"
print database.latestComment
Print "</html >
```

کار این دستورالعمل این است که آخرین نظر ثبت شده را از بانک اطلاعاتی می خواند و آن را بر روی صفحه مرورگر نشان می دهد. باید بدانیم که این، یک کد ناامن است ؛ زیرا هکر می تواند کد نوشته شده خود را به صورت یک نظر به سایت بفرستد.

```
<script>doSomethingEvil();</script>
```

و این کد در مروگر اجرا شود:

```
<html>
<h1> Most recent comment</h1>
<script>DoSomethingEvil ();</script>
</html>
```

کنترل حمله های CSRF

در حمله CSRF[51]، یک ترفند زیرکانه به کار گرفته می شود که باعث شود کاربر، همه اطلاعات مورد نیاز هکرها را در نرم افزار بارگذاری کند و به طور ناخواسته یک تراکنش انجام دهد. در این جا نقش برنامه نویس خیلی پر رنگ است و باید کد را طوری طراحی کرده و نوشته باشد که این امکان وجود نداشته باشد، مثلا دستور get که در برخی از زبانهای برنامه نویسی مثل سی شارپ هم وجود دارد، برای این منظور استفاده می شود که به اطلاعات دسترسی داشته باشد و نباید برای انجام تراکنش، فراخوانی شود.

در مثالی دیگر که با تشریح یک کد، به آن می پردازیم، کد زیر را در نظر می گیریم:

```
<span style="font-size: 12pt; color: #333333;"><?php
if (isset($_REQUEST["name"], $_REQUEST["amount"])) {
    // process the request and transfer the amount from
    // from the logged in user to the passed name.
}</span>
```

این قطعه کد در زبان PHP نوشته شده است و نام و پسورد را از کاربر دریافت می کند.

اگر طبق کد زیر، کاربری یک URL[52] بسازد و آن را به ایمیل کاربر دیگر بفرستد و کاربر دریافت کننده ایمیل، با وجود این که در نرم افزار بانک، لاگین کرده باشد، بر روی این لینک کلیک کند، به سادگی کاربر اول می تواند هر مقدار پولی را از حساب کاربر دوم برداشت کند.

```
<span style="font-size: 12pt; color: #333333;"><a href="http://example.com/proces
</span>
```

همچنین کاربر حمله کننده حتی می تواند عکسی را در لینک قرار دهد و مرورگر بدون نشان دادن آن عکس و با ترفندهای خاص، درخواست کاربر اول را بر روی سیستم کاربر دوم اجرا کند که راهکار جلوگیری از این روش در کدنویسی این است که کدنویس حرفه ای برای اعمال هر نوع تغییری در دیتابیس از دستور POST استفاده کند و از نوشتن عبارت Request_$ پرهیز کند و پارامترهای GET و POST را به ترتیب به صورت GET_$ و POST_$ استفاده کند که این باگ در برنامه وجود نداشته باشد.

[51] Cross Site Request Forgery
[52] Uniform Resource Locator

ISSN: 2745-3673
http://www.Rcsj.ir

برقراری امنیت از سمت پایگاه داده

یکی از روش‌های مهم جلوگیری از سرقت و یا تغییر اطلاعات و داده های مهم در میانه راه انتقال بین برنامه و پایگاه داده توسط مهاجمان، استفاده از رمز کردن داده ها است که به کد کردن داده معروف است. مثلاً می توان از پروتکل بسیار معروف و کاربردی SSL برای این کار استفاده نمود. این پروتکل، امنیت بالایی را فراهم می کند و انتقال اطلاعات به صورت امن بر روی گستره وب سایت ها نیز کار همین پروتکل است که امنیت بالای آن ثابت شده است. البته کاری که این پروتکل انجام می‌دهد، حفاظت از اطلاعات رد و بدل شده بین کلاینت[۵۳] و سرور[۵۴] است و استفاده از این پروتکل برای انتقال اطلاعات با امنیت بالا بین سرور و پایگاه داده توصیه می‌شود و اگر این مورد را نرم افزار واسط ما پشتیبانی نکند باید از یک برنامه شبیه به آن برای رمزگذاری اطلاعات استفاده شود [۱۰].

یکی از روش های جالب توجه که ذکر آن در اینجا خالی از لطف نیست، این است که می توانیم با به وجود آوردن یک دیواره یا تونل امنیتی به هدف خود برسیم؛ به این صورت که توسط برنامه‌ی SSH[55] که یک سازوکار برای برقراری ارتباط امن بین کلاینت و سرور می‌باشد را طوری بارگذاری کنیم که بر روی یکی از پورت ها که مورد نظر ماست شنود داشته باشد و سپس با گرفتن کلیه اطلاعات از این پورت، آن را رمز کند و به کامپیوتر مقصد بفرستد و در کامپیوتر مقصد، این رمز را به حالت عادی خود برگرداند و به پورت مورد نظر بدهد. این روش که کاربرد زیادی هم دارد موجب می شود که یک تونل مجازی بین دو سرور به وجود بیاید و اطلاعات به سرقت نرود.

یک مورد دیگر در مورد بانک اطلاعاتی، وجود دیواره آتش[۵۶] بر روی لایه بالایی آن است که این دیواره، مابین پایگاه داده و نرم افزار کاربردی قرار می گیرد و برای انتقال اطلاعات، کارهای امنیتی را انجام می دهد. کاری که در این جا انجام می شود این است که دیواره آتش، پرس و جوها یا به اصطلاح کوئری‌ها[۵۷] را قبل از فرستادن به بانک اطلاعاتی مورد بررسی قرار می دهد و آنهایی که از حیطه قوانین خاص آن خارج هستند را فیلتر می کند و نیز علاوه بر این کار، بر اجرای درست آنها کنترل می کند و آسیب پذیری های پایگاه داده را شناسایی و از هرگونه سوء استفاده جلوگیری می کند.

دیتابیس دارای لایه های متعددی است و باید هرکدام به درستی شناسایی شود تا دسترسی های امنیتی بر روی هر لایه به درستی شناخته و اعمال شود؛ همچنین در سمت پایگاه داده، Role های مشترک وجود دارد که هر کدام از نقش‌های User، Programmer، Administrator را پشتیبانی می‌کند و در زمانی که برنامه نویس، این نقش ها را تعریف می کند باید دسترسی مناسب با محدوده عملکرد هریک را به انها اختصاص دهد تا بتواند بر اختیارات، کنترل داشته باشد.

کار دیگری که می توان انجام داد این است که وقتی یک پرس و جو یا کوئری می خواهد اجرا شود از PDO[۵۸] استفاده شود. PDO یک افزونه برای PHP است که به برنامه نویس این امکان را می دهد تا یک کد را برای همه دیتابیس ها ایجاد کند. بنابراین می‌توان یک پایگاه داده را طوری طراحی نمود که نوع آن را در هر زمان بتوان تغییر داد. مثلاً ممکن است تا حالا با بانک اطلاعاتی اوراکل[۵۹] کار کرده باشیم ولی بخواهیم آن را به SQL Server تغییر دهیم که در صورت استفاده از این افزونه و با PDO، کدها را نوشته و به آسانی با استفاده از چند دستورالعمل ساده، نوع پایگاه داده را عوض نماییم.

به جز مواردی که در بالا برای به وجود آوردن امنیت بالا در دیتابیس ذکر شد، می‌توان با ترفندی جلوی حملات به SQL را گرفت آن هم با به کار بردن Prepared Query و Parameterized Query است. در ادامه به بررسی این موارد پرداخته می‌شود.

اگر بخواهیم داده ها را به صورت مستقیم و با استفاده از یک متغیر به دیتابیس وارد کنیم وکاربر داده صحیحی را وارد کند که اشکالی به وجود نمی آید ولی در زمانی که یک خرابکار، اطلاعات خاص و نادرستی را در فیلدها قرار بدهد و به پایگاه داده بفرستد، می تواند به پایگاه داده دسترسی پیدا کند و به آن نفوذ داشته باشدکه با وجود Prepared Query ها این امکان از او

[53] Client
[54] Server
[55] Secure Shell
[56] Firewall
[57] Query
[58] PHP Data Object
[59] Oracel

ISSN: 2745-3673
http://www.Rcsj.ir

گرفته می شود وکوئری های صحیح و اصولی در پایگاه داده قرار می گیرند که اطلاعات نادرست را تشخیص می دهند و آن را حذف می کنند.

در مورد Parameterized Query هم که همان کوئری ها یا پرس وجوهای پارامتربندی هستند می‌توان این طور بیان کرد که با پارامتربندی متغیرها از تغییر دادن انها توسط کدهای مخرب می توان جلوگیری کرد.

```php
<?php
$sql = "SELECT * FROM users WHERE name=:name and age=:age";
$stmt = $db->prepare($sql);
$stmt->execute(array(":name" => $name, ":age" => $age));
```

در قطعه کد بالا، برای پارامترهای name و age از تکنیک Prepare استفاده شده است که موجب Pre-Compile شدن کوئری توسط موتور دیتابیس می شود و پس از آن، مقادیر را به پارامترهای اسم دار منتسب می‌کند و ربط می‌دهد. در این کد، زمانی که تابع ()execute فراخوانده می شود، کوئری مربوط به آن با مقدارهای واقعی اجرا می شود و در نوشتن کد به روش بالا دیگر هکرها و مهاجمان نمی‌توانند SQL تخریب‌گر را تحت عنوان یک پرس و جو به پایگاه داده شما بفرستند.

امن نمودن فایل‌های سیستم

باید در هنگام برنامه نویسی از طراحی و کدهایی استفاده کنیم که نتوان از طریق آنها به فایل های سیستم دسترسی داشت. مثلاً کد زیر را در نظر بگیرید:

```php
<?php
if (isset($_GET['filename']) {
    $filename = $_GET['filename'];
    header('Content-Type: application/x-octet-stream');
    header('Content-Transfer-Encoding: binary');
    header('Content-Disposition: attachment; filename="' . $filename . '";');
    echo file_get_contents($filename);
}
```

این کد یک کد امن به حساب نمی آید چراکه همه فایل ها و پوشه هایی که کاربر به آن دسترسی دارد را مورد استفاده قرار می دهد و این زنگ خطر بزرگی است. علی الخصوص برای پوشه های سیستمی و پوشه Session که اطلاعات مهم و حیاتی سیستم و کاربر را در خود دارد. برنامه نویسان حرفه ای حتماً این موضوع را مدنظر قرار می دهند که کدی که می نویسند به کاربر اجازه دسترسی به فایل های دیگر و علی الخصوص فایل های حیاتی سیستم را ندهد. پس قطعه کد بالا یک اسکریپت نادرست و پرخطر است.

مراقبت از اطلاعات Sessions

Sessionها حافظه های کوتاه مدتی هستند که اطلاعات مربوط به ورود و خروج کاربران و برخی اطلاعات دیگر را در خود ذخیره می‌کنند و این اطلاعات را در اختیار آنالیزور گوگل می گذارند تا در تحلیل ها مورد استفاده گوگل قرار بگیرد.

اطلاعاتی که Session ذخیره می کند در یک پوشه موقت در حافظه ذخیره می شود. حال اگر یک هکر، اسکریپتی را بنویسد که به این پوشه دسترسی پیدا کند به راحتی می تواند همه اطلاعات ذخیره شده در این فولدر را در دسترس داشته باشد. اطلاعاتی مثل رمز عبور و شماره حساب بانکی و رمز اینترنتی که مهم هستند. پس یک راه خوب این است که این اطلاعات ذخیره شده در Session را رمزگذاری کنیم. همچنین علاوه بر ذخیره در پوشه مخصوص، می توانیم آن ها را در دیتابیس دیگری هم ذخیره کنیم تا در صورت هک شدن آنها، ما این اطلاعات را در جای دیگری هم داشته باشیم.

ISSN: 2745-3673
http://www.Rcsj.ir

نتیجه گیری

بانکداری همراه، مزایای بسیاری را برای بانکها و مشتریان ارائه می دهد. با این حال، امنیت یک مانع مهم برای پذیرش گسترده برنامه های بانکداری همراه است. از آنجایی که ریسک‌های امنیتی بسیاری در استفاده از برنامه‌های بانکداری همراه وجود دارد، برای بانکها و مصرف کنندکان مهم است که از این خطرات آگاه باشند و اقدامات لازم را برای کاهش خطرات انجام دهند.

تمرکز این تحقیق بر تکنولوژی‌های بانکداری همراه، اقدامات و سیاست های امنیتی بوده و با هدف بررسی چالش‌های امنیتی برنامه‌های بانکداری همراه، تلاش شد درک بهتری از وضعیت امنیت و عواملی اصلی که سطح امنیت مبادلات بانکداری همراه را تحت تاثیر قرار می‌دهند شناسایی شود.

امید است که بررسی روش های تامین امنیت انجام شده در بانکداری الکترونیکی، به مدیریت ریسکهای خدمات بانکداری همراه و بهبود امنیتی برای استفاده از بانکداری همراه موثر بوده باشد.

ISSN: 2745-3673
http://www.Rcsj.ir

منابع و مراجع

[1] Douglas W Arner, J`anos Barberis, and Ross P Buckley, (2017). “Fintech and Regtech in a Nutshell, and the future in a Sandbox”, Research Foundation Briefs, 3(4):1–20.

[2] Prateek Panda, (2015). “Security Report of Top 100 Mobile BankingApps-APAC”, Technical report, Appknox.

[۳] سرمدسعیدی، سهیل، میرابی، وحیدرضا، ۱۳۸۳، ”تجارت الکترونیک“، چاپ اول، تهران، انتشارات پرسمان.

[4] Lester Hio, (2015). “Over $7,000 lost in malware attack at fake banking portal”, http://www.straitstimes.com/singapore/over-7000-lost-in-malware-attack-at-fake-banking portal.

[۵] شجاعی، محسن، ملکی زاده، احمد، ۱۳۸۳، ”تجارت الکترونیک“، چاپ اول، مشهد، انتشارات پرتونگار.

[6] “Banks targeted by SMS phishing scam”, http://www.acma.gov.au/theACMA/engage-blogs/engage-blogs/Cybersecurity/Banks-targetted-by-SMSphishing-scam ,(2016)

[7] Henry-bodkin, (2017). “Flaw discovered in banking apps leaving millions vulnerable to hack”, http://www.telegraph.co.uk/science/2017/12/06/flaw-discovered-banking-apps-leavingmillions-vulnerable-hack. /

[8] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumg¨artner, Bernd Freisleben, and Matthew Smith, (2012). “Why Eve and Mallory love Android: An analysis of Android SSL (in) security”, Proceedings of the 2012 ACM conference on Computer and communications security, pp. 50–61.

[9] John Leyden, (2017). “Hackers Delight: Mobile bank app security flaw couldhave smacked millions”, https://www.theregister.co.uk/2017/12/11/mobile banking security research. /

[10] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov, (2012). “The most dangerous code in the world: validating SSL certificates in non-browser software”, Proceedings of the 2012 ACM conference on Computer and communications security, pp. 38–49.