آموزش کاربردی

تست نفوذ وب

تالیف: مهندس علیرضا عظیم‌زاده میلانی

انتشارات پندار پارس

۳۴۴ صفحه وزیری

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

لفن: ۶۶۵۷۳۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸همراه: ۰۹۲۱۴۳۷۱۹۶۴ **info@pendarepars.com**

نام کتاب : **آموزش کاربردی تست نفوذ وب**

ناشر : **انتشارات پندار پارس**

تالیف : **علیرضا عظیم‌زاده میلانی**

چاپ نخست : **مهر ماه ۹۵**

شمارگان : **۱۰۰۰ نسخه**

طرح جلد : **رامین شکرالهی**

چاپ، صحافی : **روز**

قیمت : **۳۰۰۰۰ تومان به همراه DVD** **شابک : ۲-۲۱-۸۲۰۱-۶۰۰-۹۷۸**

فهرست

درباره مؤلف

مهندس علیرضا عظیم‌زاده میلانی دانشجوی مقطع کارشناسی ارشد دانشگاه خواجه نصیرالدین طوسی تهران – گرایش شبکه‌های کامپیوتری می‌باشند. ایشان در سال‌های ۹۴ و ۹۵ موفق به کشف چندین آسیب‌پذیری ناشناخته با سطح "ریسک امنیتی بالا" از دو مدل WIMAX و ثبت در سایت‌های معتبر 0day.today و exploit-db شدند و از چند شرکت مطرح داخلی و خارجی پاداش (BugBounty) دریافت نمودند. ایشان در پروژه VLAN مدیریت شعب مرکزی بانک تجارت استان زنجان همکاری داشته‌اند و همچنین سخنران کنفرانس و سمینارهای علمی درباره "راه‌کارهای نفوذ و مقابله با تهدیدها در شبکه‌های کامپیوتری"، "نوع‌آوری با پایتون" و... در سازمان‌ها و دانشگاه‌های معتبر تهران، قم و نیشابور بوده‌اند. ایشان مؤلف کتاب‌های "آموزش کاربردی تست‌نفوذ با کالی لینوکس"، "آموزش کاربردی برنامه‌نویسی به زبان پایتون" و مؤلف و مترجم کتاب‌های الکترونیکی "هک و امنیت با BackTrack" و "مدیریت سرورهای لینوکسی با CentOS" نیز می‌باشند.

تخصص‌ها:

CCNA, CEH, Security+, LPIC-1, LPIC-2, RHCSA, RHCE, HIDS, WiFu, PWK, ...

پیش‌گفتار

مخاطبان اصلی این کتاب، افراد علاقه‌مند به موضوع "حمله به برنامه‌های کاربردی وب و مقابله با حملات تحت وب" هستند همچنین افرادی که نقش توسعه‌دهنده یا یک مدیر را در حوزه مدیریت برنامه‌های کاربردی وب بر عهده دارند نیز می‌توانند از مطالب کاربردی این کتاب استفاده کنند.

فرض مؤلف کتاب بر این است که شما به عنوان یک علاقه‌مند در یادگیری مفاهیم عملیاتی دوره "تست نفوذ وب"، پیش‌تر با مباحث Network+، Linux(LPIC1/Essentials)، Programming و Security+ آشنا بوده‌اید و اکنون می‌خواهید با موضوعات جدیدی در دنیای وب آشنا شوید.

مطالب این کتاب به‌صورت کاملا گویا و همراه با تصویر و آخرین تغییرات در ابزارها تشریح شده است که نیاز علاقه‌مندان را برای یادگیری دوره "تست نفوذ وب" تا سطح بسیار خوبی بر طرف می‌کند. همچنین در نگارش این کتاب، اطلاعات فنی و تجربی بسیاری که برای یادگیری آن نیاز به چندین ماه تجربه می‌باشد نیز گنجانده شده است.

اگر تاکنون کار با توزیع لینوکسی (در حوزه امنیت و هکینگ) مانند کالی، بک‌ترک، بک‌باکس و... تجربه نکرده‌اید و با زبان‌های برنامه‌نویسی مانند Python، PHP، C/C++، JavaScript، Ruby، Perl و... آشنایی ندارید، پیشنهاد می‌شود کتاب‌هایی که توسط همین مؤلف با عناوین "آموزش کاربردی تست‌نفوذ با کالی لینوکس" و "آموزش کاربردی برنامه‌نویسی به زبان پایتون" به وسیله نشر **پندارپارس** به چاپ رسیده‌اند را تهیه و در کنار این کتاب، آنها را نیز مطالعه فرمایید.

همچنین پیشنهاد می‌شود که خوانندگان، هر فصل را یک‌بار تا پایان بخوانند و به نکات، اخطارها و پیشنهادهایی که در فصل‌ها به آن اشاره شده است دقت کنند. هرگونه ایراد یا خطای فنی در کتاب را به ایمیل نویسنده (ali.azimzadeh70@Gmail.com) با موضوع "کتاب تست‌نفوذ وب" ارسال فرمایید.

از حمایت‌های مادی و معنوی شما خوانندگان عزیز که از کپی برداری، اسکن و الکترونیکی کردن کتاب پرهیز می‌نمایید و مؤلف و ناشر را برای تألیف و نشر کتاب‌های بعدی یاری می‌رسانید، بسیار سپاسگزاری می‌نمایم.

به همراه کتاب دو حلقه DVD ارائه شده است. هر دو حلقه شامل فیلم‌های آموزشی، مقالات و ابزارهای مورد نیاز برای کار با دستورات کتاب می‌باشند.

چرا تست نفوذ؟

تست نفوذپذیری، فرآیند ارزیابی امنیتی شبکه یا سیستم‌های رایانه‌ای است که به صورت شبیه‌سازی یک حمله توسط یک هکر اخلاقی صورت می‌گیرد. هدف از تست نفوذپذیری، افزایش ضریب امنیتی داده‌ها است. اطلاعات و ضعف‌های امنیتی که در تست نفوذپذیری مشخص می‌شود محرمانه تلقی می‌شود و نباید تا برطرف شدن کامل، افشاء شود. ولی در مورد هکرهای بد اندیش به این صورت نخواهد بود؛ چون هکرها از هر موقعیت زمانی و حفره امنیتی، برای نفوذ استفاده می‌کنند و برخی از آنها به اکسپلویت‌های 0day[1] دسترسی دارند و اغلب از حملات مشخص و غیر قابل تشخیص/جلوگیری استفاده می‌کنند.

اما، چرا تست‌نفوذ؟ زیرا یکی از روش‌های مقابله با حملات، شبیه‌سازی واقعی یک حمله و پیاده‌سازی تفکرات هکرهای بد اندیش، پیش از آسیب رسیدن و سوء استفاده از منابع نرم‌افزاری و سخت‌افزاری است.

نفونگر آزمون یا هکر شدن!؟ کدام یک؟ ...

مهم‌ترین تفاوت بین هکر و شخصی که تست نفوذپذیری[2] انجام می‌دهد این است که تست‌نفوذپذیری، با مجوز و قراردادی که با سازمان/شرکت امضا می‌شود، انجام می‌شود و در آخر، خروجی این تست به صورت یک گزارش تهیه می‌شود. بنابراین نفوذ به شبکه‌های کامپیوتری و دسترسی به منابع یک شبکه، در حالت معمول، کاری غیرقانونی است. اما یک متخصص امنیت، با بستن قرارداد با صاحبان، مدیران شبکه، سایت‌ها و... افزون بر قانونی کردن آن، نتایج را به نفع شما باز می‌گرداند. بنابراین نفوذگر آزمون، از دید یک هکر به برنامه یا شبکه شما می‌نگرد و تلاش می‌کند تا تمامی مشکلات و شکاف‌های امنیتی آن‌را شناسایی و به شما ارائه دهد. بدین ترتیب، شما با رفع این معایب افزون بر بالا بردن امنیت سرویس‌های خود و جلب رضایت بیشتر مشتریان، راه را بر هکرهای بد اندیش نیز خواهید بست.

---

[1] کدهایی برای استفاده از ضعف‌های امنیتی که هنوز آموزشی برای وصله‌ی(patch) آنها در دسترس عموم قرار نگرفته است.

[2] نفونگر آزمون یا Penetration-Tester

حتما بخوانید . . .

مطالب بیان شده در این کتاب صرفا برای مقاصد آموزشی است و نویسنده کتاب هیچ مسؤلیتی در برابر سوء استفاده از مطالب ارائه شده ندارد.

برابر قوانین جرائم رایانه‌ای، "ماده۱: هرکس به‌طور غیرمجاز به داده‌ها یا سامانه‌های رایانه‌ای که به وسیله تدابیر امنیتی حفاظت می‌شوند دسترسی یابد، به حبس از ۹۱روز تا یک سال یا جزای نقدی از ۵۰۰ هزار تا ۲میلیون تومان یا هر دو مجازات محکوم خواهد شد".

"ماده۸: هرکس به‌طور غیرمجاز داده‌های دیگری را از سامانه‌های رایانه‌ای یا حامل‌های داده حذف یا تخریب یا مختل یا غیرقابل پردازش کند به حبس از ۶ماه تا ۲سال یا جزای نقدی از یک میلیون تا ۴میلیون تومان یا هر دو مجازات محکوم خواهد شد".

"ماده۱۰: هرکس به‌طور غیرمجاز با اعمالی همچون مخفی کردن داده‌ها، تغییر گذر واژه یا رمزنگاری داده‌ها، مانع دسترسی اشخاص مجاز به داده‌ها یا سامانه‌های رایانه‌ای شود، به حبس از ۹۱ روز تا یک سال یا جزای نقدی از ۵۰۰ هزار تا ۲میلیون تومان یا هر دو مجازات محکوم خواهد شد".

برای مطالعه تمامی جرائم و مجازات‌های رایانه‌ای می‌توانید به لینک زیر رجوع کنید:

http://internet.ir/law.html

پرداخت پاداش به ازای کشف آسیب‌پذیری . . .

شرکت‌ها، مدیران وب‌سایت‌ها و توسعه‌دهندگان نرم‌افزار، در تدابیر سیاستی خویش با نام BugBounty-Program، به هکرهای قانونمندی که از محصولات آنها آسیب‌پذیری‌های امنیتی کشف می‌کنند، با توجه به قدرت تخریب و نوع آسیب‌پذیری، مبلغی به عنوان پاداش پرداخت می‌کنند. در ادامه، تعدادی عکس از پاداش‌هایی که مؤلف کتاب صرفا از یک شرکت مشهور داخلی دریافت نموده است، ارائه شده است:

فصل نخست

پیش‌درآمد

در این فصل، موضوعات پایه و مهم حوزه‌ی وب را که چگونه برنامه‌های کاربردی تحت وب، وب‌سرورها، تکنولوژهای وب و... به‌طور مستقیم یا غیرمستقیم با انسان یا سیستم‌ها تعامل برقرار می‌کنند را مورد بررسی قرار خواهیم داد

Web-Application چیست؟

عبارت W-B[1] برای هر فرد با توجه به موقعیتی که در آن قرار دارد و اینکه درباره‌ی چه موضوعی از وب (Web Site، Web-based System، Web Application، Web-based software یا Web) صحبت می‌کند معنای متفاوتی خواهد داشت. در این کتاب از عبارت W-B برای اشاره به هرگونه نرم‌افزار تحت وبی که عملیاتی مبنی بر دریافت ورودی از سمت کاربر انجام می‌دهد و با سیستم تعامل برقرار می‌کند تا به نیازها پاسخ دهد، استفاده می‌شود. برای نمونه زمانی که یک کاربر با یک وب‌سایت به‌منظور انجام کاری تعامل برقرار می‌کند (ورود به حساب تجاری یا حساب بانکی و انجام یکسری تراکنش) را W-B گویند.

یادآوری: به کاربران داخلی (مدیران و برنامه‌نویسان) که W-Bها را کنترل می‌کنند و کاربران خارجی (کلاینت‌ها و مشتریان) که با سایت‌ها در ارتباط هستند، Web-Users گویند.

Web-Server چیست؟

وب سرور تنها بخشی از یک نرم‌افزار در حال اجرا روی یک سیستم‌عامل از یک سرور است که به افراد اجازه برقراری ارتباط و دسترسی به یک W-B را می‌دهد. رایج‌ترین وب سرورها IIS[2] در سرورهای ویندوزی و آپاچی[3] در سرورهای لینوکسی نام دارند. به‌طور معمول مهم‌ترین دایرکتوری در ویندوز که شامل فایل‌های پیکربندی IIS و دیگر برنامه‌های‌کاربردی است **wwwroot**

---

[1] مخفف Web Application

[2] Internet Information Service

[3] Apache

نام دارد. در لینوکس مهمترین دایرکتوری‌ها **/etc/shadow** (شامل رمزهای‌عبور هش[1] شده تمامی کاربران)، **/var/*** (شامل فایل‌های دیتابیس‌ها، Logهای سیستم و...) و **/bin** (شامل برنامه‌هایی است که سیستم برای انجام کارهایش به آنها نیاز دارد؛ مانند shells، ls، grep و...) نام دارند.

HTTP (Request، Header، Body، Method) چیست؟

HTTP[2] پروتکل مرکزی برای متصل شدن و برقراری ارتباط با WWW[3] است. HTTP از مدل مبتنی بر پیام استفاده می‌کند، برای نمونه؛ کلاینت یک درخواست برای سرور ارسال می‌کند و سرور یک پاسخ به کلاینت باز می‌گرداند. پروتکل HTTP ذاتا به‌صورت Connectionless کار می‌کند. HTTP اگرچه از پروتکل Stateless-TCP برای انتقال‌هایش استفاده می‌کند، اما تبادل هر درخواست و پاسخ، یک تراکنش مستقل محسوب می‌شود و ممکن است از اتصال‌های TCP متمایزی استفاده کند.

HTTP-Request

تمام پیام‌های HTTP (درخواست‌ها و پاسخ‌ها) شامل یک یا چندین هدر، درخواست و.. می‌باشند که هریک از آنها در خط‌های جداگانه‌ای قرار گرفته‌اند. شکل زیر نشان دهنده یک HTTP-Request است:

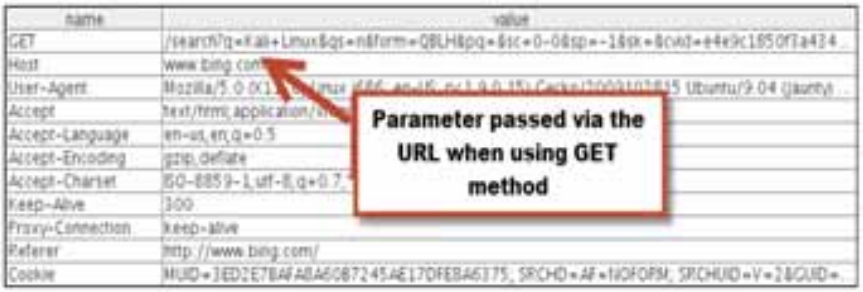
```
GET /auth/488/YourDetails.ashx?uid=129 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, application/x-shockwave-
flash, */*
Referer: https://mdsec.net/auth/488/Home.ashx
Accept-Language: en-GB
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET CLR 1.1.4322)
Accept-Encoding: gzip, deflate
Host: mdsec.net
Connection: Keep-Alive
Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476
```

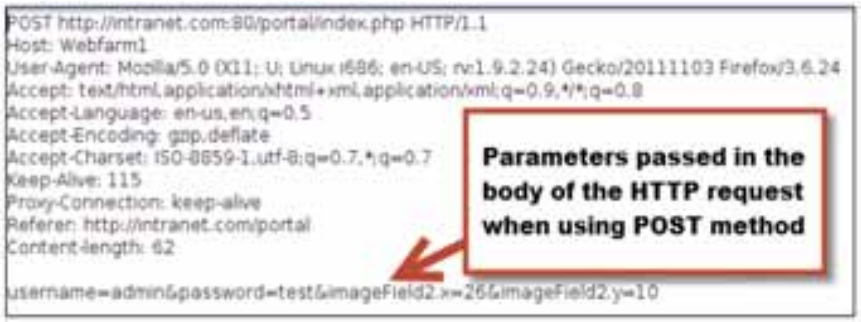
نخستین خط در تمام درخواست‌های HTTP شامل سه بخش است که با فضای خالی از یکدیگر جدا شده‌اند:

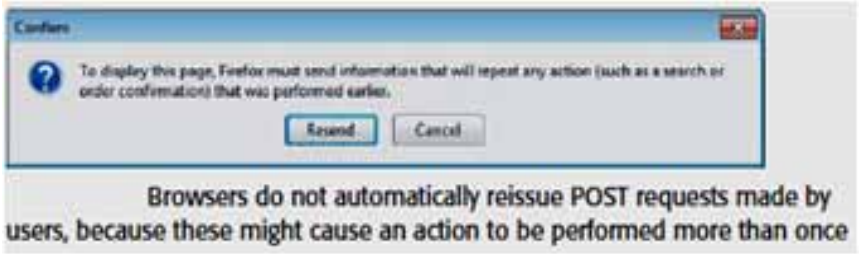
---

[1] hash
[2] Hypertext Transfer Protocol
[3] Word Wide Web

- واژه‌ای است که تعیین کننده HTTP-Method می‌باشد (معمولا GET یا POST). GET و POST، هر دو قابلیت ارسال درخواست به سوی سرور را دارند؛ اما این دو با یکدیگر یک تفاوت کوچک دارند و آن این است که روش GET، تمام داده را در بخش URL قرار می‌دهد؛ در حالیکه روش POST تمام داده را در بدنه‌ی درخواست قرار می‌دهد که اینکار باعث می‌شود امنیت داده‌های ارسالی نسبت به روش GET بالاتر رود و هر شخصی به آسانی نتواند بخش داده را تغییر دهد. همچنین از متد POST می‌توان به‌منظور بازیابی داده‌ها از سرور نیز استفاده کرد. تا چند سال پیش برخی از بانک‌ها از روش GET برای ارسال تراکنش مشتریان استفاده می‌کرند که این کار باعث می‌شد هکرها به‌راحتی بتوانند به تراکنش مشتریان بانک مربوطه دسترسی یابند و محتویات آن را مشاهده کنند. البته نباید ضعف موجود در ساده بودن (یا قابل حدس زدن) شماره‌ی هر یک از گزارش‌هایی که برای مشتریان در آن زمان تولید می‌شد را نادیده گرفت.







در ادامه به تشریح دیگر متدهای پروتکل HTTP خواهیم پرداخت:

- **متد HEAD**: تقریبا شبیه متد GET است. تنها تفاوت آنها در نوع درخواست ارسالی به سرور است؛ چون در متد HEAD نمی‌توانیم محتویات منبع درخواست شده را دریافت کنیم، بلکه صرفا می‌توانیم اطلاعات فرا-داده‌ی (Metadata) آن منبع را دریافت کنیم.
- **متد PUT**: از این متد برای آپلود و درج منبع روی سرور استفاده می‌شود. دقت داشته باشید که اگر آن منبع از پیش روی سرور باشد، معنای به‌روز رسانی منبع درخواست شده را دارد. فعال بودن این متد به هکرها اجازه می‌دهد تا کدهای مخرب روی سرور قربانی بارگذاری (آپلود) کرده و آنها را اجرا کنند.
- **متد TRACE**: از این متد برای شناسایی عیب‌ها یا خطایابی (Diagnosis) استفاده می‌شود. در حملات XSS پیرامون این متد بیشتر صحبت خواهیم کرد.
- **متد OPTION**: این متد از سرور درباره اینکه چه نوع متدهایی برای یک منبع خاص در دسترس هستند، می‌پرسد. سرور در پاسخ به این پرسش، هدر Allow ارسال می‌کند که شامل متدهای تحت پوشش و در دسترس برای آن منبع خاص است.

یادآوری: هر چیزی که در سمت راست علامت "**?**" در URL باشد را "رشته پرس‌وجو" گویند.

```
Protocol: //hostname[:port]/[path/]file[?param=value]
```

مانند:

http://IrSecTeam.org/auth/488/YourDetails.ashx?**uid=129**

- ✓ نشان دهنده URL درخواستی است. URL به‌طور معمول یک نام برای توابع درخواست شده می‌باشد که به‌صورت اختیاری ممکن است با یک رشته پرس‌وجو (query) که شامل پارامترهایی که از سوی کلاینت برای دستیابی به منبع مربوطه ارسال می‌شود نیز همراه باشد.
- ✓ نشان دهنده ورژن HTTP است.

در ادامه به تشریح دیگر خط‌ها که در یک درخواست وجود دارند خواهیم پرداخت:

- ✓ **هدر User-Agent**: این هدر برای نمایش اطلاعاتی درباره مرورگر کلاینت یا نرم‌افزار کلاینت که درخواستی را تولید کرده به‌کار می‌رود.
- ✓ **هدر Host**: نشان دهنده نام میزبان (host) در URL است.
- ✓ **هدر Cookie**: این هدر یک یا چندین کوکی به‌سوی سرور ارسال می‌کند که شامل نشست (session) کاربر می‌باشد. فرض کنید برای نخستین بار قصد ورود به حساب کاربری خود در

سایت یاهو را دارید؛ اگر روی Sing-in کلیک کنید برایتان پیامی مبنی بر اینکه تمایل دارید گذرواژه‌ی شما یادآوری شود و در دفعات بعد نیاز به وارد کردن آن و نام‌کاربری نداشته باشید نمایان می‌شود؛ اگر روی Remember یا yes یا ok کلیک کنید، یک نشست (session) بین خودتان با سرور برای سریع وارد شدن به حساب کاربری خود ساخته‌اید. هر سایتی، تا مدت زمان معینی این نشست را حفظ می‌کند و پس از چند هفته یا چند ماه، دوباره باید نام‌کاربری و گذرواژه‌ی خود را برای وارد شدن به سایت وارد کنید.

HTTP-Response

شکل زیر نشان دهنده یک HTTP-Response ساده است:

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 09:23:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1067

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml" ><head><title>Your details</title>
...
```

نخستین خط در تمام پاسخ‌های HTTP شامل **سه** بخش است که با فضای خالی از یکدیگر جدا شده‌اند:

- ✓ نشان دهنده ورژن HTTP است.
- ✓ نشان دهنده یک شماره کد-وضعیت (status-code) است که برای مشخص کردن نوع پاسخ سرور به درخواست ارسال شده است. کدهای وضعیت در **پنج** گروه دسته‌بندی می‌شوند:

1xx: جنبه اطلاع رسانی دارد.

2xx: درخواست ارسالی با موفقیت بوده است.

3xx: کلاینت مجددا به یک منبع دیگر هدایت می‌شود.

4xx: درخواست ارسال شده دارای خطا بوده است.

5xx: سرور در زمان انجام درخواست با خطا مواجه شده است. در واقع اینگونه خطاها از سمت سرور رخ می‌دهند.

اما هر یک از گروه‌های بالا دارای کد وضعیت‌های ویژه‌ی خود هستند که باید آنها را به‌خاطر داشته باشید:

- **101 Continue**: این کد بیانگر این است که سرور، نخستین بخش درخواست را دریافت کرده و یک پاسخ به منظور دریافت شدن درخواست برای کلاینت می‌فرستد و کلاینت باید ادامه‌ی درخواست‌های خود را برای سرور ارسال کند.
- **200 OK**: این کد بیانگر این است که درخواست ارسال شده از سوی کلاینت با موفقیت توسط سرور پردازش شده است.
- **201 Created**: این کد در پاسخ به درخواست کلاینت که از متد PUT برای ارسال داده‌ها استفاده کرده، برایش ارسال می‌شود (درخواست با موفقیت انجام شده است).
- **301 moved permanently**: این کد به معنای هدایت درخواست آمده از سوی کلاینت به یک مکان دائمی دیگری (URL جدید) که در هدر Location مشخص شده است، می‌باشد.
- **302 Found**: این کد به معنای هدایت درخواست آمده از سوی کلاینت به یک مکان موقت دیگری (URL جدید) که در هدر Location مشخص شده است، می‌باشد.
- **304 Not found**: سرور در پاسخ به کلاینت دستور می‌دهد از کشی (cache) که پیش‌تر از آن کپی گرفته است استفاده کند؛ چون هیچ تغییر یا به‌روز رسانی در منبع درخواستی صورت نگرفته است. در واقع این عمل باعث صرفه‌جویی در منابع سرور می‌شود و در افزایش سرعت پردازش‌های کلاینت نیز موثر است.
- **400 Bad request**: این کد بیانگر این است که کلاینت یک درخواست HTTP نامعتبر ارسال کرده است. مثلا کاربر در URL مربوطه، یک کارکتر فضای خالی (Space) قرار داده است.
- **401 Unauthorized**: پیش از پردازش درخواست آمده از سوی کلاینت، باید عمل احرازهویت صورت گیرد و هویت فرستنده مشخص شود. نوع مکانیزم هویت‌سنجی در هدر WWW-Authenticate مشخص می‌شود.

مکانیزم‌های احرازهویت در پروتکل HTTP

- **Basic**: یک مکانیزم احرازهویت ساده است که اعتبارنامه‌ی کلاینت‌را از طریق یک رشته Base64-Encoded در هدر درخواستی به همراه هر پیام ارسال می‌کند
- **NTLM**: یک مکانیزم challenge-response است و از نسخه پروتکل NTLM ویندوز استفاده می‌کند.
- **Digest**: یک مکانیزم challenge-response است و از MD5-Checksums به‌منظور ارسال اعتبارنامه‌ی کلاینت‌ها استفاده می‌کند. نسخه به‌روز شده اعتبارنامه‌ی هویت[1] مکانیزم Basic است. در این مکانیزم از یک مقدار Nonce به‌منظور ایجاد Salt در کلاینت‌ها استفاده می‌شود تا امنیت اعتبارنامه‌ها را نسبت به مکانیزم Basic بهبود بخشد.

- **403 Forbidden**: این کد بیانگر این است که هیچ فردی اجازه دسترسی به منبع درخواست شده بر روی سرور را ندارد؛ حتی با وجود مکانیزم احرازهویت.
- **404 Not found**: این کد بیانگر این است که منبع درخواست شده از سوی کلاینت، هم‌اینک وجود ندارد.
- **405 Method not allowed**: به معنای این است که سرور از متد استفاده شده در درخواست ارسالی پشتیبانی نمی‌کند (یعنی غیرمجاز بودن متد استفاده شده).
- **413 Request entity too large**: زمانی که سرور قادر به پردازش درخواست کلاینت نباشد (برای نمونه؛ زمانی که هکر در حال بررسی آسیب‌پذیری سرریز بافر است) این کد را در پاسخ ارسال می‌کند.
- **414 Request URL too long**: تقریبا مشابه کد 413 است. زمانی که URL ارسال شده از کلاینت به سرور بزرگتر از حدی است که بتوان آن‌را مدیریت کند، سرور این کد را در پاسخ ارسال می‌کند.
- **500 Internal server errors**: یعنی سرور در زمان پردازش درخواست ارسال شده با مشکل مواجه شده است.
- **503 Service unavailable**: یعنی وب سرور در حال انجام دادن وظایفش بوده و می‌تواند به درخواست‌ها پاسخ دهد؛ ولی آن برنامه کاربردی که از طریق سرور باید در دسترس باشد

[1] authentication credentials

و به درخواست پاسخ دهد در دسترس نیست. این رخداد به‌صورت موقت بوده و معمولا پس از چند دقیقه یا چند ساعت برطرف می‌شود.

برای مشاهده تمام کدهای وضعیت پروتکل HTTP می‌توانید به سایت زیر رجوع کنید:

www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

در ادامه به تشریح دیگر خط‌ها که در یک پاسخ وجود دارند خواهیم پرداخت:

- ✓ **هدر Server**: این هدر شامل یک بنر می‌باشد که نشان دهنده نرم‌افزاری است که در وب سرور نصب شده و در حال استفاده شدن می‌باشد. گاهی ممکن است این اطلاعات به اشتباه درج شده باشند؛ پس به این هدر اطمینان کامل نداشته باشید (چون از دقت کمی برخوردار است).
- ✓ **هدر Set-Cookie**: سرور یک Cookie برای مرورگر کلاینت تولید کرده و این کوکی را با یک پاسخ به‌سوی کلاینت ارسال می‌کند پس از انجام این کار توسط سرور، مرورگر کلاینت در دفعات بعدی (در صورت تمایل) از این هدر Cookie در ارسال درخواست(ها) استفاده می‌کند.
- ✓ **هدر Pragma**: این هدر به مرورگر کلاینت دستور می‌دهد پاسخ‌ها را در کش خود نگهداری **نکند**. چون ممکن است که محتوای پاسخ پیشین منقضی شده باشد (تعیین منقضی شدن یا نشدن، توسط هدر Expires صورت می‌گیرد).
- ✓ **هدر Content-Type**: این هدر نشان می‌دهد که بدنه یا متن داخلی پیام، حاوی چه نوع فایلی است (در شکل بالا حاوی یک سند HTML است).
- ✓ **هدر Content-Length**: این هدر نشان دهنده طول بدنه یا طول متن پیام برحسب بایت است.
- ✓ **هدر Content-Encoding**: این هدر نشان دهنده نوع رمزگذاری (Encoding) که بر روی محتویات پیام‌ها استفاده می‌شود، است (برای نمونه؛ gzip). این عمل توسط برخی از برنامه‌های کاربردی برای فشرده‌سازی پاسخ‌ها به‌منظور تسریع در انتقال داده‌ها استفاده می‌شود.

برای تشخیص متدهای HTTP می‌توانید از ابزارهای مشهوری مانند NC، Nmap و Metasploit نیز استفاده کنید:

root@kali: nmap --script=http-methods **<target>**

root@kali: nmap -p80 --script=http-methods **nmap.org**

root@kali: nmap -p80 --script=http-methods –script-args http.useragent="Mozilla 42" **<target>**

root@kali: nmap -Pn -p80 -n --script=http-methods **173.255.243.189**

```
Nmap scan report for nmap.org (173.255.243.189)
Host is up (0.014s latency).
PORT   STATE SERVICE
80/tcp open  http
| http-methods: GET HEAD POST OPTIONS TRACE
| Potentially risky methods: TRACE
```

تشخیص متدهای HTTP به کمک ابزار Metasploit (مد CommandLine-Interface):

```
root@kali: msfcli scanner/http/options THREADS=1 RHOSTS=173.55.243.189 E
```

شماهای رمزگذاری (Encoding Schemes)

برنامه‌های کاربردی تحت وب، از چندین رمزگذار گوناگون برای ارسال یا دریافت داده‌ها استفاده می‌کنند. در زمان حمله به برنامه‌های کاربردی وب، گاهی پیش از ارسال داده(ها) ممکن است که نیاز به رمزگذاری داده(ها) داشته باشید؛ در ادامه با رایج‌ترین شماهای رمزگذاری آشنا خواهید شد.

۱) URL-Encoding:

در این روش از رمزگذاری، URLها تنها بوسیله مجموعه کارکترهای ASCII می‌توانند روی بستر اینترنت فرستاده شوند. آن دسته از کارکترهایی که خارج از مجموعه ASCII هستند را با نماد % به همراه دو رقم در مبنای هگزا دسیمال (Hex) جایگزین می‌کنند.

نکته: اگر URLیی حاوی کارکتر space باشد، معمولا از نماد (کارکتر) + برای نمایش این فضای خالی استفاده می‌شود.

چند نمونه از کارکترهای مهمی که در URLها رمزگذاری می‌شوند:

%3d ⇔ =	%20 ⇔ Space	%25 ⇔ %	%3b ⇔ ;
%0a ⇔ New-Line	%00 ⇔ null-byte	%23 ⇔ #	

http://site.com/news.php?id=64**%20**union**%20**all**%20**select**%20**1,2,3,4,5,6,7**%23**

برای مشاهده دیگر کدها و کسب اطلاعات بیشتر می‌توانید به لینک‌های زیر مراجعه نمایید:

http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters

http://en.wikipedia.org/wiki/Percent-encoding

www.tutorialspoint.com/html/html_url_encoding.htm

www.rapidtables.com/code/text/ascii-table.htm

www.branah.com/ascii-converter

www.ascii.cl/htmlcodes.htm

۲) Unicode-Encoding:

یونی‌کد، یک استاندارد رمزگذاری کارکتری است که هدفش پشتیبانی از بیشتر زبان‌های دنیا است. یونی‌کد می‌تواند توسط رمزگذارهای کارکتری گوناگونی پیاده‌سازی شود. پر کاربرد-ترین رمزگذارها عبارت‌اند از: UTF-8 و UTF-16.

۳) Base64-Encoding:

از این روشِ رمزگذاری بیشتر در مواقعی استفاده می‌شود که داده‌ها باینری باشند. معمولا از Base64 برای رمزگذاری پیوست‌های ایمیل، عکس‌ها و اعتبارنامه‌ی کاربران در احراز هویت عمومی (برای نمونه: استفاده از مکانیزم Basic) در پروتکل HTTP استفاده می‌شود. یکی از روش‌های تشخیص اینکه آیا از این روش استفاده شده است یا خیر، نگاه کردن به [انتهای] فرم رمزگذاری شده و کشف یک تا سه کارکتر = است؛ برای نمونه:

Before Encoding:

The Web Application Hacker's Handbook

After Encoding:

VGhlIFdlYiBBcHBsaWNhdGlvbiBIYWNrZXIncyBIYW5kYm9va w==

۴) HTML-Encoding:

این رمزگذار کارکترهای ویژه‌ای مانند <، >، &، ' و " که در رشته‌های درون اسناد HTML استفاده می‌شوند را به اصطلاح escape می‌کند تا از تداخل اِلمان‌های HTML با رشته‌ها جلوگیری کند. همین که بدانید نوع رمزگذاری اعمال شده روی رشته‌ها چیست کفایت می‌کند. برای نمونه:

"<hello>world</hello>"

To:

"<hello>world</hello>"

www.degraeve.com/reference/specialcharacters.php

http://www.ascii.cl/htmlcodes.htm

http://character-code.com

۵) **Hex-Encoding**:

در فصل چهارم و پنجم کتاب از رمزگذار Hex یا Base16 به ویژه در پیاده‌سازی حملات SQLi بسیار استفاده خواهیم کرد. هر دو کارکتر هگز، نشان‌دهنده‌ی یک حرف است. همچنین گاهی بین هر دو کارکتر هگز یک 0x خواهید دید که این 0x بیانگر هگز بودن است. برای نمونه:

ASCII: Alireza-Milani

Hex without 0x: 416c6972657a612d4d696c616e69

Hex with 0x: **0x**416c6972657a612d4d696c616e69

Hex with 0x: **0x**41**0x**6c**0x**69**0x**72**0x**65**0x**7a**0x**61**0x**2d**0x**4d**0x**69**0x**6c**0x**61**0x**6e**0x**69

فایل متنی Robots.txt

از فایل robots.txt عموما در مواردی استفاده می‌شود که نمی‌خواهیم صفحاتی از سایت‌مان توسط موتورهای جست‌وجو(ربات‌های خزنده) بررسی و ایندکس شوند، یا ممکن است پیش‌تر این کار صورت گرفته باشد و اکنون می‌خواهیم آن صفحات را به هر دلیلی، از دسترس دیگران خارج کنیم. این فایل از عبارت‌های Disallow، Allow و User-Agent استفاده می‌کند:

Disallow: بیانگر آدرس صفحه‌ای است که می‌خواهید از دید روبات‌ها پنهان بماند.

Allow: بیانگر آدرس صفحه‌ای است که می‌خواهید در معرض دید روبات‌ها باشد (حالت پیش‌فرض).

User-Agent: نشان دهنده نوع روبات است.

مثال۱: مسدود کردن تمام صفحات برای تمامی ربات‌ها:

User-agent: *

Disallow: /

مثال۲: مسدود کردن یک دایرکتوری برای یک ربات خاص (Googlebot):

User-agent: Googlebot

Disallow: /AzimZadeh/

مثال۳: مسدود کردن تمام فایل‌هایی که پسوندشان png است:

User-agent: *

Disallow: /*.png$

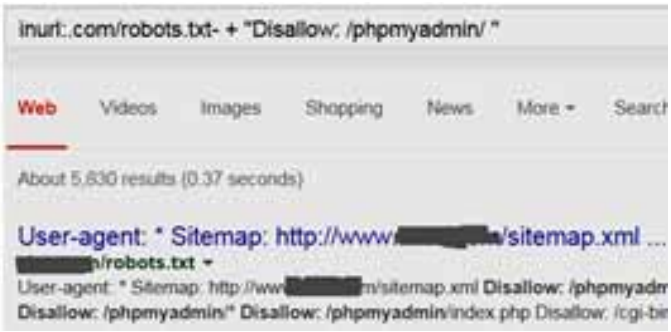
آنالیز فایل‌متنی Robots.txt با ابزار Parsero:

Parsero نام یک اسکریپت است که بطور خودکار محتویات فایل robots.txt را می‌خواند. این اسکریپت به زبان پایتون نوشته شده است. برای اجرای این ابزار باید بسته‌های زیر را نصب کنید:

```
root@kali: apt-get install python3 python3-pip libavl1 python-utidylib parsero
root@kali: apt-get install python3-pip && sudo pip3 install parsero
```

مثال۱: فرض کنید یک نفوذگر قصد دارد آدرس صفحه‌ی phpmyadmin که توسط وب‌مستر[۱] پنهان شده را با خواندن فایل Robots.txt ازطریق گوگل دورک‌ها (رجوع به فصل دوم کتاب) به‌دست آورد:

```
inurl:.com/robots.txt- + "Disallow: /phpmyadmin/"
```



مثال۲: گوگل دورک برای شناسایی صفحات مدیریت WordPress:

```
inurl:".com/robots.txt" + "Disallow: /wp-admin/
```

مثال۳: گوگل دورک برای شناسایی صفحات مدیریت Drupal:

```
inurl:".com/robots.txt" + "Disallow: ?q=admin"
```

مثال۴: گوگل دورک برای شناسایی صفحات مدیریت Joomla:

```
inurl:"/robots.txt" + "Disallow: joomla"
```

مثال۵: گوگل دورک برای شناسایی صفحات Plesk-Statistics:

```
inurl:"/robots.txt" + "Disallow: plesk-stat"
```

---

[۱] به فردی که مسؤل ساخت و نگهداری از یک یا چند سایت است می‌گویند.

مثال۶: گوگل دورک برای شناسایی سایت‌های با پسوند org. که در URLشان Administrator دارند:

site:org inurl:"robots.txt" intext:"Disallow: Administrator"

اکنون می‌خواهیم به تشریح پارامترهای ابزار Parsero بپردازیم:

-u : مشخص کردن URLیی که قصد آنالیز آن‌را دارید.

-o : تنها نمایش کد وضعیت HTTP-200.

-sb : جست‌وجو از طریق سایت Bing.com.

-f : خواندن دامنه‌ها از یک فایل متنی.

root@kali: parsero -u www.example.com

همان‌گونه که در شکل بالا مشاهده می‌کنید، لینک‌های سبز رنگ به معنای در دسترس بودنِ‌شان در وب‌سرور هستند. یکی از این لینک‌ها، **/server-status/** نام دارد. اگر آدرس **www.example.com/server-status/** را در مرورگر وارد کنیم، می‌توانیم Logهای آپاچی که از دسترس Crawlerها پنهان هستند را مشاهده کنیم:

تمرین: به‌طور دستی و بدون استفاده از ابزار parsero، برنامه‌ی vicnum در آزمایشگاه OWASP-BWA را آنالیز کنید.

برای دریافت اطلاعات بیشتر درباره آنالیز Robots.txtها می‌توانید به لینک‌های زیر رجوع کنید:

https://en.wikipedia.org/wiki/Robots_exclusion_standard

http://tools.seobook.com/robots-txt

owasp.org/index.php/Testing:_Review_Webserver_Metafiles_for_Information_Leakage_%28OTG-INFO-003%29

https://github.com/behindthefirewalls/Parsero

آسیب پذیری‌های رایج در وب

۱) **Injection**[1]: حملات تزریق زمانی رخ می‌دهند که برنامه‌های کاربردی تحت‌وب بدون اعتبارسنجی و انجام آنالیز روی داده‌های ورودی (پرس‌وجو[2])، اقدام به پردازش داده‌های وارد شده می‌کنند. برخی از رایج‌ترین حملات تزریق، هدف قرار دادن موارد زیر است:

- ✓ پرس‌وجوهای SQL (Structured Query Language)
- ✓ پرس‌وجوهای LDAP (Lightweight Directory Access Protocol)
- ✓ پرس‌وجوهای XPATH (XML Path Language)
- ✓ دستورات سیستم‌عامل (OS)

[1] تزریق

[2] query

۲) **XSS (Cross-Site Scripting)**: در این حمله، کدهای سمت کلاینت همچون جاوا اسکریپت به سایت تزریق می‌شوند و هدف اصلی هکرها کاربرانی هستند که به سایت مراجعه کرده‌اند. در حقیقت هکرها در این نوع از حمله، اطلاعات کاربران سایت را بدون آگاهی آنها به سرقت می‌برند. به زبانی ساده‌تر این آسیب‌پذیری به هکرها اجازه می‌دهد تا آنها اسکریپت‌هایی مانند Java-Script یا VB-Script روی مرورگر کلاینت(ها) اجرا کنند؛ مانند ربودن نشست کلاینت(ها) و هدایت آنها به‌سوی سایت(های) آلوده و... . این حمله بر دو نوع است:

- ✓ **Reflected (Non-Persistent)**: این روش در هر لحظه از حمله، تنها برای یک درخواست اعتبار دارد به زبان ساده‌تر می‌توان گفت رابطه‌ی ایجاد شده بین هکر و قربانی‌ها به صورت یک به یک (1:1) است. مثلا ارسال یک لینک آلوده از طریق ایمیل به کاربر که حاوی یک اسکریپت مخرب است.
- ✓ **Stored (Persistent)**: این روش برخلاف روش پیشین بسیار خطرناک است. چون اسکریپت مخرب درون برنامه کاربردی جاسازی شده و در یک لحظه می‌تواند به تعداد زیادی از کاربران حمله کند. به زبان ساده‌تر می‌توان گفت رابطه‌ی ایجاد شده بین هکر و قربانی‌ها به صورت یک به چند (1:N) است.

۳) **CSRF (Cross-Site Request Forgery)**[1]: این حمله به XSRF نیز مشهور است. نفوذگر در این حمله، کاربری که وارد (Loggined) برنامه تحت‌وب شده را مجبور به ارسال یک درخواست به نرم‌افزار وب آسیب‌پذیر می‌کند تا عملیات دلخواه نفوذگر را انجام دهد. برای نمونه، اگر کاربر به‌طور همزمان وارد ایمیل خود و حساب بانکی‌اش شده باشد، هکر برای کاربر یک ایمیل ارسال می‌کند و بیان می‌کند که اطلاعات بانکی او نیاز به به‌روزرسانی از طریق یک پیوند دارد؛ و کاربر با کلیک بر روی پیوند یا لینک، بدون آنکه آگاه باشد دستور انتقال وجه از حساب بانکی‌اش به حساب بانکی دیگری را به‌صورت خودکار برای نرم‌افزار بانک فرستاده است و در این لحظه اگر نرم‌افزار بانک فاقد مکانیزم‌های اعتبارسنجی باشد، درخواست آمده حتما پردازش خواهد شد ☠. امروزه برای جلوگیری از اینگونه حملات، از شناسه‌های معتبر (Valid-Tokens) پیچیده، طولانی و تصادفی استفاده می‌شود.

۴) **Broken-Authentication & Session-Management**[2]: گاهی به دلیل عدم پیاده‌سازی درست توابع در بخش‌هایی مانند مدیریت نشست‌ها، توکن‌ها، اعتبارنامه‌ها،

---

[1] جعل درخواست

[2] مدیریت احرازهویت و نشست‌ها به شکل نامعتبر

هویت‌سنجی استفاده‌کنندگان، سطوح دسترسی و...، باعث می‌شود هکر از چنین حفره‌های امنیتی برای دستیابی به اطلاعات مهم افراد سوء استفاده کند.

۵) **IDOR (Insecure-Direct-Object-References)**[۱]: این نوع از آسیب‌پذیری‌ها زمانی رخ می‌دهند که توسعه‌دهنده، بدون پیاده‌سازی مکانیزم‌های کنترل سطح دسترسی و...، دسترسی منبع/منابع به شیء/اشیاء داخلی برنامه/سیستم را ناآگاهانه باز می‌گذارد. هکر با دستکاری مقادیر چنین ارجاعاتی می‌تواند به داده‌های مهم برنامه‌ی وب یا سرور دسترسی غیرمجاز به‌دست آورد. برای درک بیشتر این آسیب‌پذیری توصیه می‌شود فیلمی که در مسیر "DVD>Chapter-5>Additional-Tuts" گنجانده شده است را حتما ببینید.

۶) **Sensitive-Data-Exposure**[۲]: بسیاری از برنامه‌های کاربردی تحت‌وب از اطلاعات حساسی مانند اطلاعات اعتبارسنجی کاربران، اطلاعات کارت بانکی، اعتبارنامه‌ها، و... به‌درستی محافظت نمی‌کنند هکر با ربودن این اطلاعات قادر به سوء استفاده از آنها و ایجاد خرابکاری خواهد بود. اطلاعات محرمانه و مهم، نیازمند تدابیر محافظتی ویژه‌ای است که از آن جمله می‌توان به رمزنگاری اطلاعات در زمان تبادل اطلاعات با مرورگرها اشاره نمود.

۷) **Missing-Function-Level-Access-Control**[۳]: بسیاری از برنامه‌های کاربردی وب پیش از اجرای توابع و نمایش خروجی در بخش واسط کاربری (UI)، حق دسترسی را بررسی می‌کنند. پس، نرم‌افزار باید همان سطح دسترسی را در سمت سرور نیز بررسی کند و چنانچه درخواست(های) آمده اعتبارسنجی نشود، هکر قادر به جعل درخواست‌ها بوده و می‌تواند از این رخنه برای دسترسی به امکانات توابع استفاده کند.

۸) **Componnents-with-Known-Vulnerabilities**[۴]: مؤلفه‌هایی مانند کتابخانه‌ها، فریم‌ورک‌ها و دیگر ماژول‌های نرم‌افزاری، معمولا با سطح دسترسی کامل اجرا می‌شوند. اگر در یکی از این مؤلفه‌ها آسیب‌پذیری افشا شود، باعث به‌دست آوردن دسترسی به سرور و تخریب اطلاعات خواهد شد. نرم‌افزارهایی که از این‌گونه مؤلفه‌های دارای آسیب‌پذیری‌های شناخته شده استفاده می‌کنند، امکان رخ دادن حملات گسترده‌ای را فراهم می‌سازند.

---

[۱] ارجاع نا امن (مستقیم) به أشیاء داخلی برنامه

[۲] افشای اطلاعات حساس

[۳] عدم اعمال سطح دسترسی مناسب به توابع

[۴] استفاده از مؤلفه‌های دارای آسیب‌پذیری آشکار

۹) **Seucurity-Misconfiguration**[۱]: برقراری امنیت نیازمند تعریف و استقرار پیکربندی مناسب در نرم‌افزارها، فریم‌ورک‌ها، وب‌سرورها، بانک‌های اطلاعاتی، سیستم‌عامل‌ها و.. می‌باشد. تنظیمات امنیتی می‌بایست تعریف، پیاده‌سازی و نگه‌داری شوند (تنظیمات پیش‌گزیده نا امن می‌باشند). همچنین نرم افزارها باید به‌روز نگه‌داشته شوند.

۱۰) **Unvalidated-Redirects-and-Forwards**[۲]: برنامه‌های کاربردی وب همواره در حل تغییر دادن مسیر کاربران به صفحات دیگر می‌باشند و متأسفانه از تکنیک‌های ضعیف به‌منظور تشخیص صفحات مقصد استفاده می‌کنند. بدون اعتبارسنجی مناسب، هکر قادر به هدایت قربانیان به سایت(های) آلوده خواهد بود.

برای دریافت اطلاعات بیشتر درباره آسیب‌پذیری‌های بالا می‌توانید به لینک زیر رجوع کنید:

www.owasp.org/index.php/Top_10_2013-Top_10

متدولوژی و استانداردهای تست‌نفوذ

به‌منظور تأمین امنیت مطلوب منابع نرم‌افزاری و سخت‌افزاری شبکه‌ها و سامانه‌های اطلاعاتی (با شناخت متدولوژی‌های موجود در زمینه امنیت)، نخستین گام در راستای امن‌سازی شبکه‌ها و سامانه‌های اطلاعاتی تحت وب ارزیابی مخاطرات امنیتی آنها است. ارزیابی امنیتی، امکان تشخیص وضعیت و سطح موجود امنیت منابع را فراهم می‌آورد تا بتوان پس از ترسیم وضعیت مطلوب امنیت، اقدامات مورد نیاز برای دستیابی به آن وضعیت را برنامه‌ریزی و اجرا شود. هم اکنون، متدولوژی‌های گوناگونی در حوزه‌ی ارزیابی امنیت و آزمون‌نفوذ ارائه شده‌اند که مشهورترین آنها عبارتند از:

1) OWASP (Open Web Application Security Project)

   https://www.owasp.org/index.php/OWASP_Testing_Project

   https://www.owasp.org/index.php/Web_Application_Penetration_Testing

2) PCI DSS (Payment Card Industry Data Security Standard)

   http://www.pcicomplianceguide.org/pci-faqs-2

---

[۱] پیکربندی نادرست امنیت

[۲] اعمال تغییر نامعتبر در مسیر هدایت‌کننده‌ها

3) NIST 800-115 (Technical Guide to Infromation Security Testing and Assesment)

http://csrc.nist.gov/publications/PubsSPs.html

http://www.pivotpointsecurity.com/services/nist

4) Penetration Testing Framework

http://www.pen-tests.com/penetration-testing-framework.html

5) ISSAF (Information Systems Security Assessment Framework)

http://sourceforge.net/projects/isstf

6) OSSTMM (Open Source Security Testing Methodology manual)

http://www.isecom.org/research/osstmm.html

7) PTES (Penetration Testing Execution Standard)

http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

resources.infosecinstitute.com/penetration-testing-methodology-web-applications

a) Pre-engagement Interactions

تعیین محدوده، جدول زمانی و مخاطبان، انتخاب نوع تست‌نفوذ (جعبه سیاه خاکستری یا سفید).

b) Intelligence Gathering (Information Gathering & Footprinting)

جمع‌آوری اطلاعات و آنالیز دقیق آیتم‌های مشخص شده در گام a.

c) Threat Modeling

مدل‌سازی تهدیدات به وسیله‌ی اطلاعات به‌دست آمده در گام b، برای ساخت کامل یک پروفایل از دارایی‌های شرکت.

d) Vulnerability Analysis

آنالیز دارایی‌های کشف و طبقه‌بندی شده در گام‌های b و c، برای کشف نقاط/درگاه‌های آسیب‌پذیر.

e) Exploitation & Post Exploitation

هنگامی‌که تیم نفوذ، به سیستم‌ها دسترسی پیدا کرد، افزون بر اینکه تلاش می‌کنند از ردیابی ابزارها و تکنولوژی‌های تشخیص نفوذ بگریزند، سعی می‌کنند با پیاده‌سازی استراتژی "ارتقای دسترسی"، دسترسی [سطح] بالاتری به سیستم(ها) به‌دست آورند تا با این‌کار بعدها بتوانند به دیگر نقاطی که به‌طور پیش‌گزیده در دسترس نیستند، دسترسی یابند و حملات خطرناک‌تری را شبیه‌سازی کنند.

f) Reporting