The WEB, REST, and SOAP

این فصل به استفاده از PowerShell هنگام انجام تست نفوذ روی REST و APIهای SOAP می‌پردازد. ما با توضیح اینکه چگونه می‌توانیم از PowerShell برای رمزگذاری و رمزگشایی JSON و XML استفاده کنیم، شروع خواهیم کرد. JSON و XML فناوری های اصلی در رابطه با REST و SOAP API هستند. در بخش‌های بعدی، نحوه استفاده از PowerShell را به عنوان بخشی از تست OWASP در رابطه REST و APIهای SOAP توضیح خواهیم داد.

در زیر موضوعات اصلی مورد بررسی در این فصل آمده است:

- PowerShell and the web
- Encoding JSON and XML in PowerShell
- PowerShell and REST
- PowerShell and SOAP

PowerShell and the web

PowerShell که ابتدا توسط مایکروسافت به عنوان یک چارچوب مدیریت پیکربندی و اتوماسیون کار توسعه داده شده بود، به یک زبان برنامه نویسی همه کاره تبدیل شده است که نقش مهمی در ارزیابی امنیت برنامه‌های وب، REST و SOAP ایفا می‌کند. قابلیت‌های آن فراتر از اتوماسیون و مدیریت است و آن را به ابزاری ضروری برای متخصصان امنیت سایبری در هنگام انجام ارزیابی‌های امنیتی کامل تبدیل می‌کند. در این بخش، به این خواهیم پرداخت که چگونه PowerShell می‌تواند به طور موثر به عنوان بخشی از تست امنیت برنامه وب، REST و SOAP استفاده شود.

Web application security testing with PowerShell

با ترکیب قدرتمند PowerShell و تست امنیت برنامه‌های کاربردی وب در این فصل، به قلمرو امنیت سایبری پیشگیرانه کاوش کنید. کشف کنید که چگونه قدرت برنامه نویسی PowerShell می‌تواند برنامه‌های وب شما را در برابر تهدیدات احتمالی تقویت کند. از شناسایی آسیب‌پذیری‌ها تا اجرای پروتکل‌های امنیتی قوی، این کاوش مدیران را با ابزارهایی برای انجام ارزیابی‌های امنیتی کامل مجهز می‌کند. مثال‌های دنیای واقعی و بینش‌های عملی نشان می‌دهند که چگونه PowerShell کارایی وظایفی مانند تست نفوذ و اسکن آسیب‌پذیری را افزایش می‌دهد. در ادامه به برخی از قابلیت‌های PowerShell در این زمینه می پردازیم:

Automated Scanning: PowerShell به عنوان یک پلتفرم عالی برای خودکار کردن اسکن‌های امنیتی برنامه‌های کاربردی وب عمل می‌کند. آزمایش‌کنندگان می‌توانند ابزارهای مختلف اسکن

آسیب‌پذیری مانند OWASP ZAP، Burp Suite یا Nessus را با اسکریپت‌های PowerShell برای اسکن وب‌سایت‌ها برای آسیب‌پذیری‌هایی مانند SQL injection، Cross-Site Scripting (XSS) و غیره ادغام کنند. قابلیت‌های اسکریپت نویسی PowerShell امکان سفارشی سازی و زمان بندی اسکن‌ها، بهینه سازی زمان و تضمین پوشش جامع را فراهم می‌کند.

Data Extraction and Analysis: توانایی PowerShell در ایجاد درخواست‌های HTTP و تجزیه محتوای HTML برای استخراج و تجزیه و تحلیل داده‌ها در طول تست امنیتی بسیار ارزشمند است. آزمایش کنندگان می‌توانند از آن برای واکشی صفحات وب، استخراج اطلاعات و کشف مسائل امنیتی پنهان استفاده کنند. این موضوع شامل بررسی قرار گرفتن در معرض داده‌های حساس، Crawling سایت‌ها برای شناسایی دایرکتوری‌های مخفی، یا تجزیه و تحلیل جاوا اسکریپت برای آسیب‌پذیری‌های امنیتی بالقوه است.

Password Bruteforcing and Enumeration: از اسکریپت‌های PowerShell می‌توان برای حملات brute force پسورد استفاده کرد و به آزمایش کنندگان اجازه می‌دهد تا رمزهای عبور ضعیف یا قابل حدس را در یک برنامه وب شناسایی کنند. این به ارزیابی قدرت سیستم‌های ورود و کنترل‌های دسترسی کمک می‌کند.

Custom Exploitation: انعطاف‌پذیری اسکریپت‌نویسی PowerShell ایجاد پیلودهای سفارشی را برای بهره‌برداری از آسیب‌پذیری‌های کشف شده در طول آزمایش امکان‌پذیر می‌سازد. برای مثال، یک تست نفوذگر می‌تواند یک اسکریپت PowerShell برای نشان دادن تأثیر یک آسیب‌پذیری XSS با اجرای کد دلخواه در یک برنامه وب ایجاد کند.

Web Application Firewall (WAF) Bypass: متخصصان امنیتی می‌توانند از PowerShell برای آزمایش اثربخشی WAF ها با ایجاد پیلودهای مخرب و تکنیک‌های Bypass برای دور زدن آن‌ها استفاده کنند. این به سازمان‌ها در تقویت مکانیسم‌های دفاعی خود کمک می‌کند.

Authentication and Session Management Testing: PowerShell می‌تواند سناریوهای مختلف احراز هویت مانند brute forcing یا آزمایش مکانیسم‌های مدیریت جلسه را شبیه سازی کند. این به شناسایی نقاط ضعف در کنترل‌های دسترسی و مدیریت کاربر کمک می‌کند.

Reporting and Documentation: اسکریپت‌های PowerShell می‌توانند تولید گزارش‌های آزمایشی دقیق، از جمله آسیب‌پذیری‌های کشف‌شده، شدت آن‌ها و مراحل اصلاح توصیه‌شده را خودکار کنند. این تضمین می‌کند که نتایج آزمایش برای ارتباط با ذینفعان و برای اهداف انطباق به خوبی مستند شده است.

REST application security testing with PowerShell

ماموریتی را برای تقویت امنیت برنامه‌های REST خود با استفاده از قابلیت‌های پویا PowerShell آغاز کنید. در این فصل، ما رابطه همزیستی بین اسکریپت نویسی PowerShell و تست امنیتی قوی برای API های RESTful را بررسی می‌کنیم. با نمایش‌های عملی و سناریوهای دنیای واقعی، مدیران بینش‌هایی را در مورد استفاده از PowerShell برای انجام ارزیابی‌های امنیتی موثر به دست می‌آورند. مواردی که در این مورد می‌توان استفاده نمود عبارتند از:

API Endpoint Testing: توانایی PowerShell برای ارسال درخواست‌های HTTP برای آزمایش API های REST بسیار ارزشمند است. آزمایش‌کنندگان می‌توانند اسکریپت‌هایی را برای تعامل با نقاط پایانی API، ارسال انواع مختلف درخواست‌ها (GET، POST، PUT، DELETE) برای ارزیابی امنیت داده‌های ورودی، خروجی و مکانیزم‌های احراز هویت بسازند.

Authentication and Authorization Testing: PowerShell می‌تواند سناریوهای مرتبط با احراز هویت و تعین سطح دسترسی را در برابر API های REST شبیه سازی کند. این به آزمایش‌کنندگان اجازه می‌دهد تا بررسی کنند که چگونه API با توکن‌ها، نقش‌ها و مجوزهای احراز هویت مدیریت می‌شوند و هر گونه آسیب‌پذیری یا مسائل کنترل دسترسی را شناسایی می‌کند.

Input Validation Testing: از PowerShell می‌توان برای خودکارسازی تست‌ها برای اعتبارسنجی ورودی با ارسال پیلودهای مختلف و انواع داده‌ها به نقاط پایانی API استفاده کرد. این به شناسایی آسیب‌پذیری‌هایی مانند تزریق SQL یا دستکاری داده‌ها کمک می‌کند.

Load and Performance Testing: اسکریپت‌های PowerShell می‌توانند چندین درخواست API همزمان را شبیه‌سازی کنند و آزمایش‌کنندگان را قادر می‌سازند تا نحوه عملکرد REST API تحت بارگذاری و اینکه آیا در معرض حملات Denial-of-Service (DoS) است را ارزیابی کنند.

SOAP application security testing with PowerShell

در این فصل می‌توانید امنیت برنامه SOAP خود را با قابلیت‌های پویا PowerShell مورد ارزیابی قرار داده و افزایش دهید. از هم افزایی بین اسکریپت نویسی PowerShell و تست امنیتی قوی پرده برداری نموده و مدیران را برای انجام ارزیابی‌های کامل بر روی API های مبتنی بر SOAP مجهز کنید. مواردی که در این مورد می‌توان استفاده نمود عبارتند از:

SOAP Endpoint Testing: PowerShell همچنین می‌تواند با سرویس‌های وب مبتنی بر SOAP تعامل داشته باشد. آزمایش‌کنندگان می‌توانند اسکریپت‌هایی را برای ارسال درخواست‌های SOAP، آزمایش متدهای مختلف و تجزیه و تحلیل پاسخ‌ها برای آسیب‌پذیری‌های امنیتی ایجاد کنند.

XML Data Parsing: SOAP برای تبادل داده به شدت به XML متکی است. قابلیت تجزیه XML PowerShell برای تجزیه و تحلیل پاسخ‌ها و پیلودهای SOAP برای شناسایی مسائل امنیتی، مانند تزریق XML یا حملات XXE مفید است.

Authentication and Encryption Testing: از PowerShell می‌توان برای آزمایش اینکه چگونه سرویس‌های SOAP احراز هویت و رمزگذاری داده‌ها را مدیریت می‌کنند، استفاده می‌شود و اطمینان حاصل می‌کند که اطلاعات حساس به اندازه کافی در طول انتقال محافظت می‌شوند.

Boundary Testing: اسکریپت‌های PowerShell می‌توانند تست مرزی را خودکار کنند، درخواست‌های SOAP را با مقادیر داده‌های شدید یا غیرمنتظره ارسال کنند تا نحوه رسیدگی سرویس به موارد لبه و ضعف‌های امنیتی احتمالی را ارزیابی نمایند.

به طور خلاصه، PowerShell یک ابزار ضروری برای متخصصان امنیتی هنگام انجام تست‌های امنیتی برنامه‌های وب، REST و SOAP است. تطبیق پذیری، قابلیت‌های اتوماسیون و امکانات یکپارچه سازی آن با سایر ابزارهای امنیتی، آزمایش کنندگان را قادر می‌سازد تا ارزیابی‌های کاملی انجام دهند، آسیب‌پذیری‌ها را کشف کنند و وضعیت امنیتی این برنامه‌ها را افزایش دهند. با این حال، استفاده مسئولانه از PowerShell ضروری است، و می‌بایست اطمینان حاصل شود که تمام فعالیت‌های آزمایشی در محدوده‌های قانونی و اخلاقی بوده و همچنین با مجوز مناسب از صاحب برنامه یا سازمان انجام می‌شوند.

Encoding JSON and XML in PowerShell

در تست نفوذ، رمزگذاری و رمزگشایی داده‌های JSON و XML در PowerShell برای تجزیه و تحلیل و دستکاری پاسخ‌های برنامه‌های کاربردی وب و API ها ضروری است. در ادامه به نحوه انجام این وظایف پرداخته شده است.

Encoding JSON in PowerShell

JSON فرمت رایجی است که برای تبادل داده بین کلاینت و سرور استفاده می‌شود. برای رمزگذاری داده‌ها به صورت JSON در PowerShell، مراحل زیر را دنبال کنید:

Create a PowerShell Object: با ایجاد یک شی PowerShell شروع کنید که داده‌هایی را که می‌خواهید به عنوان JSON رمزگذاری کنید، نگه می‌دارد. این شی می‌تواند یک hashtable، شی سفارشی یا یک آرایه باشد. در اینجا یک مثال است:

```
$data = @{
    Username = "ajcblyth"
    PassCode = 9816 }
```

Encode to JSON: از cmdlet ConvertTo-Json برای تبدیل شی PowerShell به رشته‌ای با فرمت JSON استفاده کنید:

```
$jsonString = $data | ConvertTo-Json
```

Optional Formatting: می‌توانید پارامترهای قالب‌بندی را به ConvertTo-Json اضافه کنید تا عمق و قالب‌بندی خروجی JSON را کنترل کنید و آن را خواناتر نمایید:

```
$jsonString = $data | ConvertTo-Json -Depth 2 -Pretty
```

Decoding JSON in PowerShell

برای Decoding داده‌های JSON در PowerShell برای تست نفوذ، اغلب با پاسخ‌های JSON از برنامه‌های کاربردی وب یا APIهایی مواجه می‌شوید که باید آن‌ها را تجزیه و تحلیل کنید:

Retrieve JSON Data: داده‌های JSON را از یک درخواست HTTP، فایل یا منبع دیگری دریافت کنید:

```
$jsonString = Invoke-RestMethod -Uri "https://snowcapcyber.com/
api/data" -Method GET
```

Decode JSON: از ConvertFrom-Json برای Decoding رشته JSON به یک شی PowerShell استفاده کنید:

```
$decodedData = $jsonString | ConvertFrom-Json
```

Access Data: اکنون می‌توانید مانند هر شیء دیگر PowerShell به داده‌های موجود در شی Decode شده دسترسی داشته باشید:

```
$name = $decodedData.Username
$age = $decodedData.PassCode
```

Encoding XML in PowerShell

XML فرمت دیگری است که برای تبادل داده استفاده می‌شود. برای Encoding داده‌ها به عنوان XML در PowerShell، مراحل زیر را دنبال کنید:

Create an XML Object: از New-Object برای ایجاد یک سند XML استفاده کنید:

```
$xml = New-Object System.Xml.XmlDocument
```

Add Data to XML: سند XML را با داده‌ها پر کنید. شما می‌توانید عناصر و ویژگی‌ها را ایجاد کنید:

```
$root = $xml.CreateElement("Person")
$xml.AppendChild($root)
$name = $xml.CreateElement("Username")
$name.InnerText = "ajcblyth"
$root.AppendChild($name)
$code = $xml.CreateElement("PassCode")
$code.InnerText = "9816"
$root.AppendChild($age)
```

Convert XML to String: از ویژگی OuterXml برای تبدیل سند XML به رشته استفاده کنید:

```
$xmlString = $xml.OuterXml
```

Decoding XML in PowerShell

برای Decoding داده‌های XML در PowerShell در طول تست نفوذ، مراحل زیر را دنبال کنید:

Retrieve XML Data: داده‌های XML را از یک درخواست HTTP، فایل یا منبع دیگری دریافت کنید:

```
$xmlString = Get-Content -Path "userdetails.xml"
```

Load XML: از متد LoadXml برای بارگذاری داده‌های XML در یک سند PowerShell XML استفاده کنید:

```
$xml = New-Object System.Xml.XmlDocument
$xml.LoadXml($xmlString)
```

Access Data: در صورت نیاز می‌توانید داده‌ها را از سند XML پیمایش و استخراج کنید:

```
$name = $xml.SelectSingleNode("//Username").InnerText
$code = $xml.SelectSingleNode("//PassCode").InnerText
```

با تسلط بر Encoding و Decoding مربوط به JSON و XML در PowerShell، تست نفوذگران می‌توانند به طور موثر پاسخ‌ها را تجزیه و تحلیل کنند، آسیب‌پذیری‌ها را شناسایی کنند و داده‌ها را در طول ارزیابی‌های امنیتی دستکاری کنند. چه ارزیابی REST APIها یا برنامه‌های کاربردی وب باشد، این تکنیک‌ها برای ارزیابی کنترل‌های امنیتی و شناسایی مسائل بالقوه‌ای که نیاز به Mitigation دارند ضروری هستند. هنگام انجام تست نفوذ، همیشه از داشتن مجوز مناسب اطمینان حاصل کنید و به دستورالعمل‌های اخلاقی پایبند باشید.

PowerShell and REST

استفاده از Representational State Transfer (REST) در PowerShell برای تست نفوذ یک رویکرد ارزشمند برای ارزیابی امنیت برنامه‌ها و سرویس‌های وب است. با تعامل با RESTful API، تست نفوذگران می‌توانند آسیب‌پذیری‌ها و نقاط ضعفی را که می‌توانند توسط عوامل مخرب مورد سوء استفاده قرار گیرند، شناسایی کنند. بیایید نحوه استفاده از REST در PowerShell را برای تست نفوذ در حالی که تجزیه و تحلیل خود را با چارچوب Open Web Application Security Project (OWASP)، یک منبع شناخته شده برای امنیت برنامه‌های وب، همسو می‌کنیم، بررسی کنیم.

OWASP analysis – injection

Objective: آسیب‌پذیری‌های تزریق را در API های REST تست کنید.

Methodology: می‌توانید از PowerShell برای ایجاد ورودی مخرب استفاده کنید و آن را به عنوان بخشی از یک درخواست برای آزمایش آسیب‌پذیری‌های تزریق مانند تزریق SQL، تزریق NoSQL یا تزریق دستور OS ارسال کنید. به عنوان مثال ما تست تزریق SQL زیر را داریم:

```
$uri = "https://api.snowcapcyber.com/resource"
$queryParam = "inputValue' OR '1'='1"
$response = Invoke-RestMethod -Uri "$uri?param=$queryParam" -Method
GET
```

OWASP analysis – broken authentication

Objective: احراز هویت و مدیریت جلسه را در REST API ارزیابی کنید.

Methodology: می‌توانید از PowerShell برای ارسال درخواست‌های احراز هویت و تجزیه و تحلیل پاسخ‌ها استفاده کنید. به عنوان مثال، تأیید اعتبار ضعیف آزمایش زیر را داریم:

```
$uri = "https:// api.snowcapcyber.com/authenticate"
$headers = @{
 "Authorization" = "Basic <base64EncodedCredentials>" }
$response = Invoke-RestMethod -Uri $uri -Method GET -Headers $headers
```

OWASP analysis – sensitive data exposure

Objective: ارزیابی کنید که آیا داده‌های حساس در پاسخ‌های API در معرض دید قرار می‌گیرند یا خیر.

Methodology: از PowerShell برای ارسال درخواست‌ها و تجزیه و تحلیل پاسخ‌ها برای قرار گرفتن در معرض داده‌های ناخواسته استفاده کنید. لازم به ذکر است که می‌توانیم از عبارات منظم برای فیلتر کردن کوئری‌ها استفاده کنیم. برای مثال، بررسی کنید که آیا اطلاعات حساسی مانند رمز عبور یا شماره کارت اعتباری در پاسخ‌ها وجود دارد یا خیر:

```
$uri = "https:// api.snowcapcyber.com/resource"
$response = Invoke-RestMethod -Uri $uri -Method GET
```

OWASP analysis – XML External Entities (XXE)

Objective: آسیب‌پذیری‌های مرتبط با XML مانند XXE را در API های RESTful تست کنید.

Methodology: از PowerShell می‌توان برای ارسال پیلودهای XML مخرب به API و تجزیه و تحلیل پاسخ‌ها استفاده کرد. به عنوان مثال ما آزمایش زیر را برای XXE داریم:

```
$uri = "https://api.snowcapcyber.com/resource"
$xmlPayload = '<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE foo [
<!ENTITY xxe SYSTEM "file:///etc/passwd"> ]><foo>&xxe;</foo>'
$headers = @{
    "Content-Type" = "application/xml" }
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
-Body $xmlPayload
```

OWASP analysis – broken access control

Objective: تست کنید آیا API کنترل‌های دسترسی مناسب را اعمال می‌کند یا خیر.

Methodology: از PowerShell برای ارسال درخواست‌هایی با سطوح مختلف مجوز استفاده کنید و تجزیه و تحلیل کنید که آیا کاربران غیرمجاز می‌توانند به منابع محدود دسترسی داشته باشند یا خیر. به عنوان مثال، می‌توانید کنترل‌های دسترسی ناکافی را آزمایش کنید:

```
$uri = "https://api.snowcapcyber.com/restricted-resource"
$headers = @{
    "Authorization" = "Bearer <accessToken>" }
$response = Invoke-RestMethod -Uri $uri -Method GET -Headers $headers
```

OWASP analysis – security misconfiguration

Objective: پیکربندی نادرست امنیتی در API را شناسایی کنید.

Methodology: از PowerShell می‌توان برای ارسال درخواست‌ها و تجزیه و تحلیل پاسخ‌ها برای نشانه‌های پیکربندی نادرست مانند اطلاعات Debug آشکار یا اعتبارنامه‌های پیش‌فرض استفاده کرد:

```
$uri = "https://api.snowcapcyber.com/debug-info"
```

```
$response = Invoke-RestMethod -Uri $uri -Method GET
```

OWASP analysis – Cross-Site Scripting (XSS)

Objective: آسیب‌پذیری‌های XSS را در پاسخ‌های REST API تست کنید.

Methodology: از PowerShell برای ایجاد پیلودهای مخرب و ارسال آن‌ها در درخواست استفاده کنید. برای شناسایی هر گونه آسیب‌پذیری XSS از نوع Reflected یا Stored، پاسخ‌ها را تجزیه و تحلیل کنید. به عنوان مثال، می‌توانید XSS از نوع Reflected را آزمایش کنید:

```
$uri = "https://api.example.com/search"
$searchQuery = '<script>alert("XSS");</script>'
$response = Invoke-RestMethod -Uri "$uri?q=$searchQuery" -Method GET
```

OWASP analysis – Cross-Site Request Forgery (CSRF)

Objective: API را برای آسیب‌پذیری‌های CSRF ارزیابی کنید.

Methodology: صفحات HTML مخرب را با پیلودهای CSRF در PowerShell ایجاد کنید و کاربران را فریب دهید تا با آن‌ها تعامل داشته باشند. برای تعیین موفقیت آمیز بودن حملات CSRF، پاسخ‌های API را زیر نظر بگیرید. به مثال زیر توجه کنید:

```
$html = @"
<html>
    <body>
        <form id="maliciousForm" action="https://api.snowcapcyber.com/
action" method="POST">
            <input type="hidden" name="csrfToken" value="attackerToken">
        </form>
        <script>
            document.getElementById("maliciousForm").submit();
        </script>
```

```
    </body>
</html>
"@
```

OWASP analysis – unvalidated redirects and forwards

Objective: تغییر مسیرها و فورواردهای نامعتبر در API را آزمایش کنید.

Methodology: از PowerShell برای ارسال درخواست‌هایی با URLهای تغییر مسیر یا فوروارد دستکاری شده استفاده کنید و تجزیه و تحلیل کنید که آیا API اجازه تغییر مسیر نامعتبر را می‌دهد یا خیر. به عنوان مثال، می‌توانید تغییر مسیرهای نامعتبر را به شکل زیر آزمایش کنید:

```
$uri = "https://api.snowcapcyber.com/redirect?url=http://malicious.
com"
$response = Invoke-RestMethod -Uri $uri -Method GET
```

OWASP analysis – insecure deserialization

Objective: آسیب‌پذیری‌های Insecure Deserialization را در API ارزیابی کنید.

Methodology: از PowerShell برای ارسال درخواست‌ها با اشیاء Serialized مخرب استفاده کنید و تجزیه و تحلیل کنید که آیا API تلاش می‌کند آن‌ها را Deserialization کند یا خیر. به مثال زیر توجه کنید:

```
$uri = "https://api.snowcapcyber.com/process-data"
$serializedPayload = "maliciousSerializedObject"
$headers = @{
 "Content-Type" = "application/xml"}
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
-Body $serializedPayload
```

گنجاندن چارچوب OWASP در فعالیت‌های تست نفوذ هنگام استفاده از REST در PowerShell برای ارزیابی جامع امنیت برنامه‌های وب ضروری است. انعطاف‌پذیری PowerShell به آزمایش‌کنندگان اجازه می‌دهد تا درخواست‌ها و پیلودهای سفارشی بسازند و پاسخ‌ها را برای شناسایی آسیب‌پذیری‌های همتراز با ده برتر OWASP Top 10 تجزیه و تحلیل کنند، که در نهایت به توسعه و استقرار برنامه‌ای امن‌تر کمک می‌کند. همیشه اطمینان حاصل کنید که مجوزهای لازم را دارید و دستورالعمل‌های اخلاقی را هنگام انجام تست‌های نفوذ دنبال می‌کنید.

PowerShell and SOAP

استفاده از Simple Object Access Protocol یا SOAP در PowerShell برای تست نفوذ می‌تواند به ارزیابی امنیت سرویس‌های وب و API هایی که بر این پروتکل متکی هستند کمک کند. SOAP معمولا برای ارتباط بین برنامه‌ها استفاده می‌شود و برای شناسایی آسیب‌پذیری‌ها بسیار مهم است. در اینجا راهنمای نحوه استفاده از SOAP در PowerShell برای تست نفوذ در حین پیوند دادن تجزیه و تحلیل به چارچوب OWASP آورده شده است.

OWASP analysis – injection

Objective: آسیب‌پذیری‌های تزریق در درخواست‌ها و پاسخ‌های SOAP را آزمایش کنید.

Methodology: مانند آزمایش برای تزریق در پروتکل‌های دیگر، می‌توانید پیلودهای SOAP را برای آزمایش تزریق SQL، تزریق XML یا سایر آسیب‌پذیری‌های تزریق دستکاری کنید. به عنوان مثال، می‌توانید برای تزریق SQL در یک درخواست SOAP کد زیر را آزمایش کنید:

```
$uri = "https://api.snowcapcyber.com/soap-endpoint"
$soapPayload = @"
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:web="http://www.example.com/webservice">
    <soapenv:Header/>
    <soapenv:Body>
        <web:Login>
            <web:username>' OR '1'='1</web:username>
            <web:password>password</web:password>
        </web:Login>
    </soapenv:Body>
</soapenv:Envelope>
"@
$headers = @{
    "Content-Type" = "text/xml; charset=utf-8"
}
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
```

```
-Body $soapPayload
```

OWASP analysis – XXE

Objective: آسیب‌پذیری‌های XXE را در پیام‌های SOAP تست کنید.

Methodology: مشابه آزمایش XXE در REST، می‌توانید پیلودهای مخرب XML را برای آزمایش آسیب‌پذیری‌های XXE ایجاد کنید. به عنوان مثال، می‌توانید XXE را در یک درخواست SOAP آزمایش کنید:

```
$uri = "http s://api.snowcap cyber.com/soap-endpoint"
$soapPayload = @"
<soapenv:Envelope xmlns:soapenv="htt p://schemas.xml soap.org/soap/
envelope/" xmlns:web="http://www.example.com/webservice">
   <soapenv:Header/>
   <soapenv:Body>
      <web:ProcessXML>
         <web:xmlInput><![CDATA[<!DOCTYPE foo [ <!ENTITY xxe SYSTEM
"fil e:///etc/passwd"> ]><foo>&xxe;</foo>]]></web:xmlInput>
         </web:ProcessXML>
   </soapenv:Body>
</soapenv:Envelope>
"@
$headers = @{
   "Content-Type" = "text/xml; charset=utf-8"
}
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
-Body $soapPayload
```

OWASP analysis – authentication bypass

Objective: ارزیابی مکانیسم‌های احراز هویت در سرویس‌های مبتنی بر SOAP

Methodology: با ایجاد درخواست‌های SOAP با سناریوهای مختلف احراز هویت، آسیب‌پذیری‌های دور زدن احراز هویت را آزمایش کنید. به عنوان مثال، می‌توانید احراز هویت ضعیف را آزمایش کنید:

```
$uri = "https://example.com/soap-endpoint"
$soapPayload = @"
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:web="http://www.example.com/webservice">
   <soapenv:Header/>
   <soapenv:Body>
      <web:Login>
         <web:username>admin</web:username>
         <web:password>password123</web:password>
      </web:Login>
   </soapenv:Body>
</soapenv:Envelope>
"@
$headers = @{
   "Content-Type" = "text/xml; charset=utf-8"
}
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
-Body $soapPayload
```

OWASP analysis – insecure deserialization

Objective: آسیب‌پذیری‌های Insecure Deserialization را در پیام‌های SOAP آزمایش کنید.

Methodology: مشابه آزمایش برای Insecure Deserialization در زمینه‌های دیگر، در این بخش نیز باید پیلودهای مخرب SOAP را برای آزمایش آسیب‌پذیری‌ها ارسال کرد. به عنوان مثال، شما می‌توانید برای Insecure Deserialization در یک درخواست SOAP کد زیر را آزمایش کنید:

```
$uri = "https://example.com/soap-endpoint"
$soapPayload = @"
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.example.com/webservice">
```

```
    <soapenv:Header/>
    <soapenv:Body>
        <web:ProcessData>
<web:data><![CDATA[O:8:"Example":1:{s:4:"data";s:10:"malicious";}]]></web:d
ata>
        </web:ProcessData>
    </soapenv:Body>
</soapenv:Envelope>
"@
$headers = @{
    "Content-Type" = "text/xml; charset=utf-8"
}
$response = Invoke-RestMethod -Uri $uri -Method POST -Headers $headers
-Body $soapPayload
```

OWASP analysis – unvalidated redirects and forwards

Objective: برای تغییر مسیرها و فورواردهای نامعتبر در سرویس‌های مبتنی بر SOAP آزمایش کنید.

Methodology: درخواست‌های SOAP را با URL های تغییر مسیر دستکاری شده ایجاد کنید و تجزیه و تحلیل کنید که آیا این سرویس اجازه هدایت مجدد نامعتبر را می‌دهد یا خیر. به عنوان مثال، می‌توانید برای تغییر مسیرهای نامعتبر در یک درخواست SOAP کد زیر را آزمایش کنید:

```
$uri = "https://example.com/soap-endpoint"
$soapPayload = @"
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:web="http://www.example.com/webservice">
    <soapenv:Header/>
    <soapenv:Body>
        <web:Redirect>
            <web:url>https://malicious.com</web:url>
        </web:Redirect>
```

```
    </soapenv:Body>
</soapenv:Envelope>
"@
$headers = @{
    "Content-Type" = "text/xml; charset=utf-8"
}
$response = Invoke-R
```

با استفاده از این متدولوژی‌ها برای استفاده از SOAP در PowerShell در طول تست نفوذ، می‌توانید به طور موثر امنیت وب سرویس‌ها و API ها را ارزیابی کنید و آسیب‌پذیری‌های همسو با چارچوب OWASP را شناسایی کنید. همیشه از داشتن مجوز مناسب، پیروی از دستورالعمل‌های اخلاقی و دریافت مجوزهای لازم هنگام انجام تست‌های نفوذ اطمینان حاصل کنید. علاوه بر این، گزارش آسیب‌پذیری‌های شناسایی‌شده را به طرف‌های مسئول برای اصلاح در نظر بگیرید.

Summary

به طور خلاصه، این فصل نحوه کدگذاری داده‌ها در PowerShell در ساختارهای JSON و XML را معرفی می‌کند. سپس به شما نشان دادیم که چگونه می‌توان از PowerShell در چارچوب OWASP برای آزمایش API های REST و SOAP استفاده کرد.

در فصل بعدی، نحوه استفاده از PowerShell را به عنوان بخشی از یک تست نفوذ جامع بر روی بلاک پیام سرور (SMB)، اکتیو دایرکتوری (AD) و پروتکل دسترسی دایرکتوری سبک وزن (LDAP) بررسی خواهیم کرد.