हैकिंग

हैक करने के विभिन्न तरीके हैं, और यह विषय काफी दिलचस्प है। एक शौकिया □□□□□ □□□ हैकर के रूप में, मुझे लगता है कि इस क्षेत्र में सीखने के लिए बहुत सारा ज्ञान है। यहां, मैं कुछ ऐसे तरीकों को रिकॉर्ड करूंगा जिन्हें मैंने खोजा है।

डिफ़ॉल्ट पासवर्ड

कुछ वेबसाइटें, जिनमें सरकारी एजेंसियों की वेबसाइटें भी शामिल हैं, अभी भी डिफ़ॉल्ट पासवर्ड का उपयोग करती हैं। हालांकि कई कंपनियाँ या उपयोगकर्ता अपने डिफ़ॉल्ट क्रेडेंशियल्स को बदल देते हैं, लेकिन अन्य ऐसा करने में विफल रहते हैं। उपयोगकर्ता अक्सर आलसी होते हैं, और 12345678 जैसे पासवर्ड अभी भी आमतौर पर देखे जाते हैं। यह विशेष रूप से पुराने या विशिष्ट सिस्टम के लिए सच है।

□□□□ या □□□□□□

□□□□ और □□□□□□ दोनों नेटवर्क टूल हैं जो नेटवर्क स्कैनिंग और डेटा ट्रांसफर के लिए उपयोग किए जाते हैं। यहां इन दोनों टूल्स के बारे में संक्षिप्त जानकारी दी गई है:

□□□□

□□□□ (□□□□□□□ □□□□□□) एक ओपन-सोर्स टूल है जिसका उपयोग नेटवर्क डिस्कवरी और सुरक्षा ऑडिटिंग के लिए किया जाता है। यह नेटवर्क पर होस्ट्स और सर्विसेज की जानकारी एकत्र करने के लिए विभिन्न तकनीकों का उपयोग करता है। □□□□ का उपयोग निम्नलिखित कार्यों के लिए किया जा सकता है:

- □ होस्ट डिस्कवरी
- □ पोर्ट स्कैनिंग
- □ सर्विस और वर्जन डिटेक्शन
- □ ऑपरेटिंग सिस्टम डिटेक्शन

उदाहरण:

```
nmap -sP 192.168.1.0/24
```

यह कमांड 192.168.1.0/24 सबनेट पर सभी होस्ट्स की खोज करेगा।

□□□□□□□

□□□□□□□ (□□) एक सरल यूटिलिटी है जिसका उपयोग नेटवर्क कनेक्शन बनाने, डेटा ट्रांसफर करने और नेटवर्क टेस्टिंग के लिए किया जाता है। इसे अक्सर "स्विस आर्मी नाइफ" के रूप में जाना जाता है क्योंकि यह विभिन्न नेटवर्क कार्यों के लिए उपयोगी है। □□□□□□□ का उपयोग निम्नलिखित कार्यों के लिए किया जा सकता है:

- □ पोर्ट स्कैनिंग
- □ डेटा ट्रांसफर
- □ पोर्ट लिसनिंग
- □ नेटवर्क डीबगिंग

उदाहरण:

```
nc -zv 192.168.1.1 22
```

यह कमांड 192.168.1.1 पर पोर्ट 22 की जांच करेगा और बताएगा कि यह खुला है या बंद।

दोनों टूल्स नेटवर्क प्रशासकों और सुरक्षा पेशेवरों के लिए अत्यंत उपयोगी हैं और विभिन्न नेटवर्क कार्यों को करने में मदद करते हैं।

ये टूल्स सर्वर के पोर्ट्स को स्कैन करने के लिए उपयोग किए जाते हैं। विशेष रूप से आमतौर पर उपयोग किए जाने वाले पोर्ट्स जैसे 80, 22, और 443 पर ध्यान दें। □□□ इंस्टेंस के लिए, डिफ़ॉल्ट यूजरनेम `ec2-user` होता है। □□□□□ इंस्टेंस के लिए, यह `azure-user` होता है। □□□□□□ □□□□□ इंस्टेंस के लिए, डिफ़ॉल्ट यूजरनेम आमतौर पर `ubuntu` या `google-cloud` होता है। अन्य क्लाउड इंस्टेंस के लिए, यह आमतौर पर `root` होता है।

ब्राउज़र कंसोल का उपयोग करना

ब्राउज़र कंसोल छिपी हुई जानकारी का निरीक्षण करने के लिए उपयोगी है। कभी-कभी, महत्वपूर्ण डेटा □□□□ या □□□□□□□□□□ कोड में एम्बेडेड होता है लेकिन पेज पर दिखाई नहीं देता है।

बैकडोर्स

जीवन में, बैकडोर इमारतों में अनधिकृत प्रवेश प्रदान करते हैं, जो अक्सर बिना ध्यान दिए या बिना सुरक्षा के होते हैं, जैसे पार्किंग स्थल या साइड के दरवाजे। इसी तरह, सिस्टम में छिपे हुए बैकडोर हो सकते हैं जो सामान्य सुरक्षा प्रोटोकॉल को बायपास करते हैं।

सोशल इंजीनियरिंग

लोगों के उपनाम, जन्मदिन और सोशल मीडिया पोस्ट बहुत सारी व्यक्तिगत जानकारी प्रकट कर सकते हैं। अक्सर, इन विवरणों का उपयोग कमजोर पासवर्ड बनाने के लिए किया जाता है। वाई-फाई नेटवर्क के लिए, किसी के घर का नंबर या अन्य पहचान संबंधी विवरण जानने से उनके □□□□ या पासवर्ड का अनुमान लगाने में मदद मिल सकती है।

□□□ इंजेक्शन

किसी भी इनपुट फ़ील्ड के लिए, `? 1=1` के साथ परीक्षण करना कमजोरियों और संभावित □□□ इंजेक्शन बिंदुओं की पहचान करने के लिए एक सामान्य तकनीक है।

एक्चुएटर या हेल्थ एपीआई

□□□ सर्वर के लिए, □□□□□□ □□□□ जैसे एप्लिकेशन एक `/actuator` एंडपॉइंट प्रदान करते हैं जो मशीन और एप्लिकेशन स्वास्थ्य डेटा प्रदान करता है। अन्य वेब फ्रेमवर्क में भी इसी तरह की कार्यक्षमता होती है जो संवेदनशील सर्वर विवरण को उजागर कर सकती है।

ट्रैफिक मॉनिटरिंग

फ्रंटएंड और बैकएंड के बीच कैसे इंटरैक्शन होता है, यह समझने के लिए □□□□□ पर □□□□□□□□ □□□□□ जैसे प्रॉक्सी एप्लिकेशन का उपयोग करें ताकि रिक्वेस्ट लॉग्स को रिकॉर्ड और विश्लेषण किया जा सके। इससे आपको कंपोनेंट्स के बीच पाथ और डेटा एक्सचेंज के बारे में जानकारी मिल सकती है।

□□□ की सीमाएँ और एज केस

□□□ (एप्लिकेशन प्रोग्रामिंग इंटरफेस) डेवलपर्स के लिए एक शक्तिशाली टूल है, लेकिन इसकी अपनी सीमाएँ और एज केस भी होते हैं। इन्हें समझना और उन्हें संभालना एक अच्छे डेवलपर की पहचान है। इस ब्लॉग पोस्ट में, हम □□□ की कुछ सामान्य सीमाओं और एज केस पर चर्चा करेंगे।

1. □□□□ □□□□□□□□□

- **समस्या:** कई □□□ प्रदाता अपने □□□ के उपयोग पर □□□□ □□□□□□□□□ लागू करते हैं। यह एक निश्चित समय अवधि में □□□ कॉल की संख्या को सीमित करता है।
- **समाधान:** □□□□ □□□□□□□□□ को संभालने के लिए, आपको अपने कोड में थ्रॉटलिंग और रिट्री लॉजिक को लागू करना चाहिए। इसके अलावा, □□□ प्रदाता द्वारा प्रदान किए गए □□□□ □□□□□□□□□ हेडर को मॉनिटर करना भी महत्वपूर्ण है।

```
import time
import requests

def make_api_call():
    response = requests.get('https://api.example.com/data')
    if response.status_code == 429:
        time.sleep(int(response.headers['Retry-After']))
```

```
        make_api_call()
    return response.json()
```

2. □□□□□□□□□□□□□□□□ □□□□ □□□□□□□□□□□□□□□□

- □ **समस्या:** □□□□ को सुरक्षित रखने के लिए □□□□□□□□□□□□□□□□ और □□□□□□□□□□□□□□□□ की आवश्यकता होती है। यदि □□□□ कुंजी या टोकन गलत हो, तो □□□□ कॉल असफल हो जाएगी।
- □ **समाधान:** सुनिश्चित करें कि आपका कोड सही ढंग से □□□□ कुंजी या टोकन को पास कर रहा है। इसके अलावा, टोकन को समय-समय पर रिफ्रेश करना भी आवश्यक है।

```
import requests

headers = {
    'Authorization': 'Bearer YOUR_ACCESS_TOKEN'
}

response = requests.get('https://api.example.com/data', headers=headers)
print(response.json())
```

3. □□□□□□ □□□□□□□□□□

- □ **समस्या:** □□□□ कॉल के दौरान विभिन्न प्रकार की त्रुटियाँ हो सकती हैं, जैसे नेटवर्क त्रुटि, सर्वर त्रुटि, या अन्य अपेक्षित त्रुटियाँ।
- □ **समाधान:** अपने कोड में उचित □□□□□□ □□□□□□□□□□ लागू करें। यह सुनिश्चित करें कि आपका कोड विभिन्न □□□□□ स्टेटस कोड को संभाल सकता है और उपयोगकर्ता को सही त्रुटि संदेश प्रदर्शित कर सकता है।

```
import requests

try:
    response = requests.get('https://api.example.com/data')
    response.raise_for_status()
except requests.exceptions.HTTPError as err:
    print(f"HTTP error occurred: {err}")
except Exception as err:
    print(f"Other error occurred: {err}")
```

4. □□□□ □□□□□□□□□□

- **समस्या:** □□□ से प्राप्त डेटा हमेशा अपेक्षित प्रारूप में नहीं हो सकता है। यह डेटा प्रोसेसिंग में त्रुटियाँ पैदा कर सकता है।
- **समाधान:** □□□ से प्राप्त डेटा को प्रोसेस करने से पहले उसकी वैधता की जाँच करें। यह सुनिश्चित करें कि डेटा अपेक्षित प्रारूप में है और आवश्यक फ़ील्ड मौजूद हैं।

```
def validate_data(data):
    if not isinstance(data, dict):
        raise ValueError("Expected a dictionary")
    if 'key' not in data:
        raise ValueError("Missing 'key' in data")
    return data
```

5. □□□□□□□□□□

- **समस्या:** □□□ के नए संस्करण जारी होने पर, पुराने संस्करण अप्रचलित हो सकते हैं और उनका समर्थन बंद हो सकता है।
- **समाधान:** □□□ के संस्करण को ध्यान में रखें और अपने कोड को नए संस्करण के साथ अपडेट करें। □□□ प्रदाता द्वारा प्रदान किए गए □□□□□□□□□□ दस्तावेज़ को पढ़ें और उसके अनुसार अपने कोड को संशोधित करें।

```
import requests

response = requests.get('https://api.example.com/v2/data')
print(response.json())
```

6. □□□□ □□□□□

- **समस्या:** □□□ कॉल के दौरान विभिन्न एज केस हो सकते हैं, जैसे खाली डेटा, अमान्य इनपुट, या अप्रत्याशित सर्वर प्रतिक्रियाएँ।
- **समाधान:** अपने कोड में एज केस को संभालने के लिए विशेष लॉजिक लागू करें। यह सुनिश्चित करें कि आपका कोड विभिन्न परिदृश्यों में सही ढंग से काम करता है।

```
def handle_edge_cases(data):
    if not data:
        return "No data available"
    if 'error' in data:
        return f"Error: {data['error']}"
    return data
```

निष्कर्ष

□□□ की सीमाओं और एज केस को समझना और उन्हें संभालना एक महत्वपूर्ण कौशल है। यह सुनिश्चित करता है कि आपका एप्लिकेशन विभिन्न परिस्थितियों में स्थिर और विश्वसनीय रहता है। उचित □□□□□ □□□□□□□□, □□□□ □□□□□□□□□□, और □□□□ □□□□□□□□ जैसी तकनीकों का उपयोग करके, आप अपने □□□ कॉल को अधिक मजबूत और विश्वसनीय बना सकते हैं।

किसी □□□ या सर्वर की सीमाओं और एज केस (□□□□ □□□□□) का परीक्षण करना महत्वपूर्ण है। एक □□□□□□□□□□□ □□□□□□ □□ □□□□□□□ (□□□□) हमला अनुरोध सीमाओं को पार करने का प्रयास करता है। इसके अलावा, एज केस ऐसे परिदृश्य होते हैं जहां □□□□ प्रतिबंधित डेटा तक पहुंच की अनुमति दे सकते हैं। इनका परीक्षण करने से यह सुनिश्चित करने में मदद मिलती है कि उचित पहुंच नियंत्रण (□□□□□□ □□□□□□□□) लागू हैं।

एडमिन पैनल

कभी-कभी, एडमिन या आंतरिक पैनल पर्याप्त रूप से सुरक्षित नहीं होते हैं। `/admin` जैसे पथ तक पहुंचने या `admin.xx.com` जैसे सबडोमेन पर जाकर यह जांचना उचित है कि क्या ये क्षेत्र ठीक से सुरक्षित हैं।