SecuritySnack: Repo The Repo - NPM Phishing

⋮ 10/16/2025

Recently, a series of high profile supply chain compromises were caused by malicious code written to NPM repositories managed by stolen developer credentials. While developers of prominent NPM repositories have been targeted for many years,these events prompted CISA to release an alert due to their widespread nature. Attackers stole developer accounts through a phishing campaign involving fake NPM management and login pages. This tactic enabled them to take over accounts for malicious activity and remains one of the most common and effective methods of credential theft.

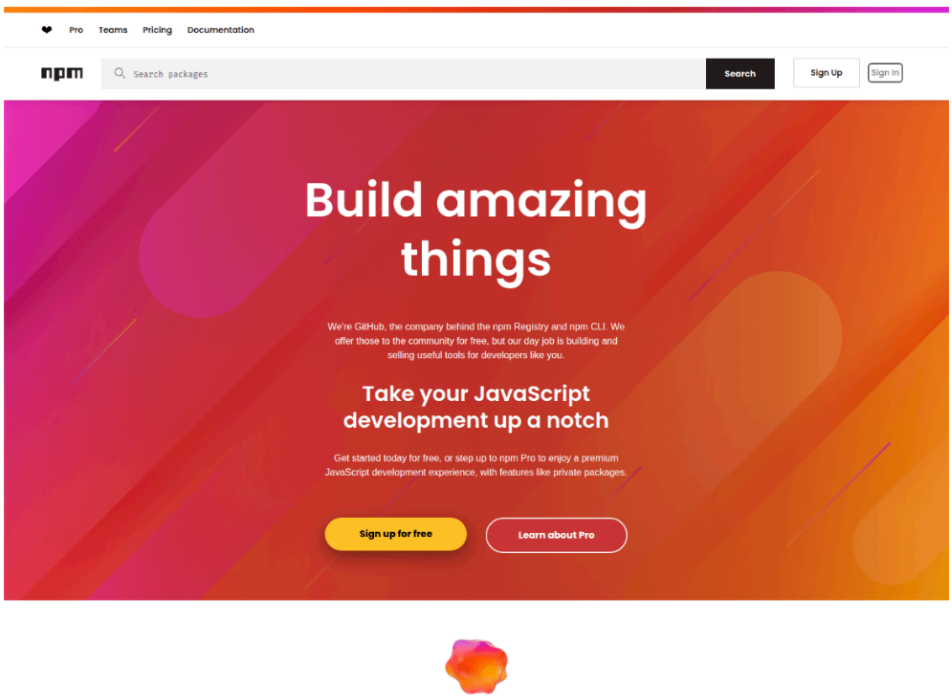
Details

NPJMS is the largest JavaScript repository, with two official domains: npmjs.com is the main site and npmjs.org is also an official NPM domain. Phishers have historically used variations of this domain to deceive users, leveraging common tactics such as typo-squatting through domains like “npnjs[.]com”, which are particularly easy to overlook when presented in lower case characters.

Examining a recently spoofed NPM login page configuration with the domain “npmjs[.]pro” demonstrates how the attack progresses through three distinct stages, each designed to capture a piece of information or deceive the user into the next step.

Stage 1: Homepage Lure

This is the initial landing page of the phishing site, designed to build trust and initiate the login flow.

npmjs[.]pro	

This is a relative sign-in link. On the malicious domain, clicking “Sign In” sends the user to the /login path on the attacker’s server, not the legitimate npmjs[.]com. The attacker’s server logs the request and serves the fake login page (Stage 2) in response.

```
<a href="/login" id="signin" class="_352269a0 pa1 lh-copy br2 f6 no-underline fw5 black dim">Sign In</a>
```

Stage 2: Initial Credential Capture

After being funneled from the fake homepage, the user is presented with the fake login form.

npmjs[.]pro/login	

The form's action="/login/" sends the submitted username and password to a script on the attacker's server. The attacker's server captures and logs the credentials. It then uses them to initiate a login attempt on the real npmjs[.]com, triggering a legitimate email OTP to be sent to the victim. At this point, the user's primary npm credentials (username and password) are compromised,and the next stage is to retrieve their MFA/OTP code.

```
<form id="login" method="POST" action="/login/" class="ma0">
    <div class="mb2">
        <input type="text" id="login_username" name="username" />
    </div>
    <div class="">
        <input type="password" id="login_password" name="password" />
    </div>
    <input type="hidden" name="csrftoken" value="SuFFIhbDu5alJd0wZDS1_Z0Xis7lFTLFRDu5AKK9wsn" />
    <div>
        <button type="submit" class="_24a1e9c7 dab9924c">Sign In</button>
    </div>
</form>
```

Stage 3: MFA / OTP Code Interception

The attacker's server immediately presents a page to intercept the second-factor authentication code.

This form captures the value from the name="otp" field and sends it to the /login/email-otp endpoint on the attacker's server. The user receives a real OTP via email (triggered by the attacker), which reinforces their

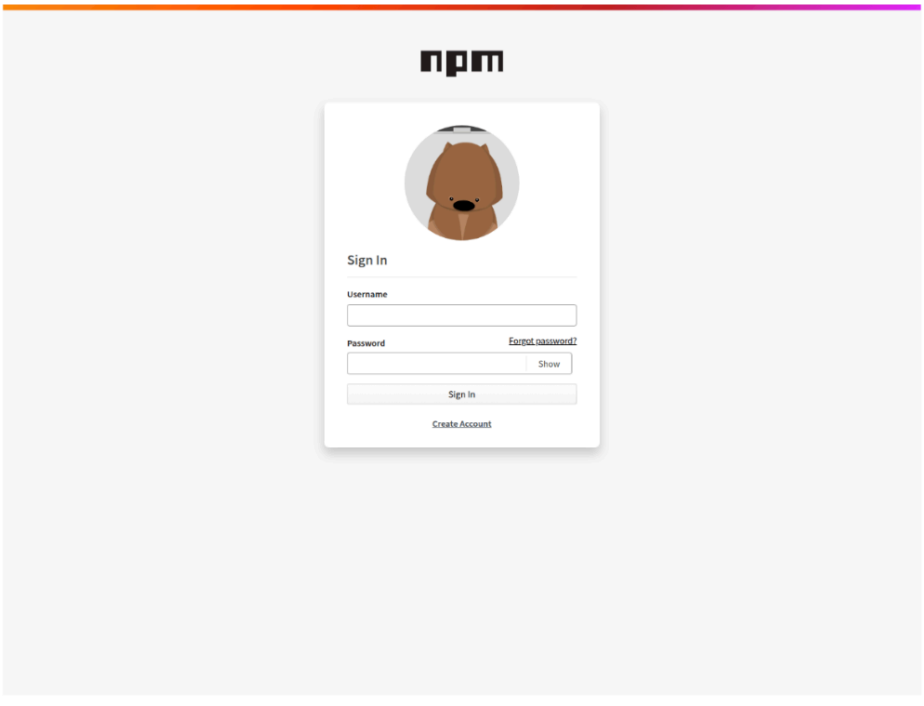
belief that the process is secure. The attacker's server receives the valid OTP and now possesses all information required to hijack the account.

```
<form id="login" method="POST" action="/login/email-otp?next=%2F" class="center">
    <h2 class="mt0 tc">Enter One-time Password.</h2>
    <div class="">
        <input type="text" id="login_otp" name="otp" inputMode="numeric" />
        <input type="hidden" name="username" value="" />
        <input type="hidden" name="password" value="" />
    </div>
    <input type="hidden" name="csrftoken" value="SuFFIhbDu5alJd0wZDS1_Z0Xis7lFTLFRDu5AKK9wsn" />
    <div>
        <button type="submit" class="_24a1e9c7 dab9924c">Login</button>
    </div>
</form>
```

Stage 4: Session Hijack and Evasion

This final stage is a server-side action to complete the attack.

The attacker uses the captured credentials and OTP to establish their own authenticated session on the real npmjs[.]com, then redirects the victim to avoid suspicion. The attacker now has full, authenticated access to the victim's npm account. The victim remains unaware that their account and session have been compromised. Their browser redirects them to the real npm sign-in page, making them believe the process did not complete.

npmjs[.]pro/login/email-otp?next=%2F	

Conclusion

This detailed attack flow for credential theft and account takeover shows that classic credential harvesting tactics remain highly effective. As our reliance on shared software supply chains grows, developer vigilance has never been more important. While multi-factor authentication (MFA) is an essential defense, this example shows that OTP codes are only as secure as the domain they are entered into. Always verify the URL in your address bar before entering credentials, and consider adopting phishing-resistant MFA, like hardware security keys, to truly secure your accounts.

IOCs

The provided IOCs are recently registered typosquatted domains of NPMJS.

npmjscdn[.]xyz
npmjs[.]us
npmjs[.]pro
npmjs[.]us
npmjs[.]pro
npmjs[.]us[.]org
npmjs[.]us[.]com
npmjs[.]se
npmjs[.]work
npmjs[.]online
npmjs[.]wtf
npmjs[.]help
npmjs[.]cam
npmjs[.]web[.]id
npmjs[.]support
npnjs[.]org
npnjs[.]com