

# How Attackers Can Misuse Sitemaps to Enumerate Users and Discover Sensitive Information

By adi peretz

Published: 2023-02-22 · Archived: 2026-04-05 14:05:20 UTC

## Introduction



Recently, I stumbled upon a [blog post](#) by Tal Folkman, a researcher at Checkmarks, about how they uncovered an army of fake user accounts using sitemap.xml. Her research highlighted how powerful sitemap.xml could be for attackers looking to enumerate potential victims. It got me thinking about the many other ways attackers can exploit sitemap.xml to gather information about a website and its users. In this blog post, I'll dive deeper into the risks of sitemap.xml and explore some of the techniques attackers can use to abuse this seemingly innocuous file.

## What is sitemap.xml

Sitemaps are XML files that contain a list of all the pages on a website, along with additional metadata about each page. They are commonly used to help search engines efficiently crawl and index a website. However, because sitemaps contain a comprehensive outline of a site's contents, malicious actors can also misuse them to gather sensitive data or enumerate private user information. In this blog post, we will explore three tactics that attackers may employ to take advantage of sitemaps and the risks that sitemaps pose if not properly secured. With a deeper understanding of these vulnerabilities, web developers can better protect their sites by avoiding common mistakes when implementing sitemaps. Awareness of how sitemaps can be exploited is crucial to reducing the attack surface and strengthening a website's security.

## Tactic #1: Enumerate User Pages to Guess Accounts

If a website has user profile pages or other content specific to each user, the sitemap may list out each of these individual pages. An attacker can attempt to guess or brute force the actual user accounts on the system by analyzing the patterns of the page names or IDs. For example, if pages are named "user123.html", "user456.html", and so on, an attacker can increment through numbers to discover other user page names and, eventually, user accounts. While this tactic requires some effort, it could be worthwhile for an attacker seeking to compromise multiple user accounts. To mitigate this risk, website owners should avoid naming user pages or IDs in a predictable, sequential pattern. Using randomized names, IDs, or other obfuscated systems for user pages can make it more difficult for attackers to guess or brute force their way to real user accounts.

## Attack Example

Suppose a website has user profile pages that are named using predictable, sequential patterns, such as “user123.html”, “user124.html”, and so on. An attacker can use a web crawler or a scraping tool to download the website's sitemap and analyze the patterns of the page names. Then, the attacker can use an automated script or a tool like Burp Suite to increment through the numbers and discover other user page names and, eventually user accounts. Once the attacker has obtained a list of user accounts, they can use this information to launch targeted attacks such as brute-forcing passwords or exploiting vulnerabilities in the website to gain access to sensitive information. The attacker can also sell the list of user accounts on the dark web or use it for other malicious purposes. The POC code below is looking for common usernames in sitemap.xml

```
import requests
from bs4 import BeautifulSoup

usernames = ["admin", "root", "guest", "test", "user"]

url = "https://www.example.com/sitemap.xml"
response = requests.get(url)

soup = BeautifulSoup(response.content, "xml")

locs = soup.find_all("loc")

for loc in locs:
    url = loc.text
    for username in usernames:
        if username in url.lower():
            print(f"Found common username '{username}' in URL: {url}")
```

## Mitigation

To mitigate this risk, website owners should avoid naming user pages or IDs in a predictable, sequential pattern. Instead, they can use randomized names, IDs, or other obfuscated systems for user pages to make it more difficult for attackers to guess or brute force their way to real user accounts. Additionally, website owners can implement rate-limiting and other security measures to detect and prevent brute-force attacks.

## Tactic #2: Discover Hidden or Private Pages

Sitemaps are intended to contain a comprehensive list of all website pages, including those not linked from elsewhere. As a result, sitemaps may inadvertently reveal “hidden” or private pages that are not meant to be publicly accessible. By combing through the sitemap, attackers can discover these unpublished pages and gain insight into potential vulnerabilities or other sensitive data on the site. For example, a sitemap could list an admin page, user data page, or other internal pages that search engines should not index or be accessible to the public. To reduce this risk, web developers should exclude private, unlinked, or internal pages from the sitemap. Only pages

that are meant to be indexed and crawled should be included. In addition, sitemaps should be protected from public access using authentication or other restrictions to prevent attackers from freely analyzing the sitemap contents.

## Attack Example

### Get adi peretz's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

an attacker might abuse the technique of discovering hidden or private pages by analyzing a sitemap:

1. The attacker obtains the URL of the sitemap for the target website.
2. The attacker downloads the sitemap and looks for hidden or private pages that are not meant to be publicly accessible.
3. The attacker analyzes the unpublished pages for potential vulnerabilities or sensitive data on the site.
4. If the attacker finds a hidden or private page vulnerable to attack, they can attempt to exploit the vulnerability to gain access to additional sensitive data or compromise the website.
5. Alternatively, the attacker could sell or leak the sensitive data they have found on the hidden or private pages for financial gain or other malicious purposes.

```
import xml.etree.ElementTree as ET

sitemap_path = 'sitemap.xml'

tree = ET.parse(sitemap_path)
root = tree.getroot()

namespace = {'default': 'http://www.sitemaps.org/schemas/sitemap/0.9'}

for url in root.findall('default:url', namespace):
    loc = url.find('default:loc', namespace).text
    for meta in url.findall('..//default:meta[@content="noindex"]', namespace):
        print(loc)
```

## Mitigations

To mitigate the risk of an attacker discovering hidden or private pages by analyzing a sitemap, here are some measures that can be implemented:

1. Use a robots.txt file: In the robots.txt file, web developers can specify which pages and directories should be excluded from search engine crawlers. This can prevent private or sensitive pages from being included in the sitemap.
2. Limit sitemap access: Sitemaps should be protected from public access by using authentication or other restrictions to prevent unauthorized access. Web developers can restrict access to sitemaps to specific IP addresses, users, or groups and/or add an authentication mechanism.
3. Exclude internal pages from the sitemap: Only pages meant to be indexed and crawled should be included. Internal pages that are not meant to be publicly accessible, such as admin pages, should be excluded.

### Tactic #3: Crawl Entire Website

With a sitemap that lists all pages on a site, an attacker can crawl the entire website and mirror its contents. This could allow the attacker to launch further attacks or thoroughly analyze the site for vulnerabilities at their leisure. The sitemap serves as a handy roadmap to exploring everything the website offers, for better or for worse.

```
import xml.etree.ElementTree as ET
import re

sitemap_path = 'sitemap.xml'

tree = ET.parse(sitemap_path)
root = tree.getroot()

namespace = {'default': 'http://www.sitemaps.org/schemas/sitemap/0.9'}

for url in root.findall('default:url', namespace):
    loc = url.find('default:loc', namespace).text
    if re.search(r'(\.php?)|(\.asp?)|(\.aspx?)|(/cgi-bin/)|(/myphpadmin.*|)/admin/)|(/wp-|
```

### Mitigation

Limit the breadth of pages included in the sitemap to mitigate this threat. Only include pages that need to be indexed by search engines rather than a full listing of every page on the site. In addition, implement protections on the website pages to prevent mass scraping and crawling. For example, use CAPTCHAs, rate limiting, or other controls to block automated crawlers from accessing pages.

### Conclusion

While sitemaps are useful for search engine optimization, they can also unintentionally aid attackers in discovering sensitive information or enumerating a website's contents. Web developers should ensure that sitemaps are designed properly and do not list any private or internal pages. Awareness of these security risks can

help reduce vulnerabilities from sitemap misuse by implementing best practices in sitemap creation and website protection. With a balanced approach to optimization and security, websites can reap the benefits of sitemaps while avoiding the pitfalls.

---

Source: <https://medium.com/@adimania/how-attackers-can-misuse-sitemaps-to-enumerate-users-and-discover-sensitive-information-361a5065857a>