Kud I Enter Your Server? New Vulnerabilities in Microsoft Azure

By Paul Litvak

Published: 2020-10-08 · Archived: 2026-04-06 01:15:44 UTC

Main Findings

We discovered two vulnerabilities in **Microsoft Azure**. They existed in a popular cloud service called **Azure App Services**—specifically impacting Linux servers—and should be on the radar of enterprise organizations that use cloud resources.

The first vulnerability enabled an attacker with access to the server to take over the App Service's git repository and implant phishing pages accessible through the Azure Portal. The second vulnerability allowed an attacker with an existing low-severity vulnerability on the application (SSRF) to upgrade to full code execution on the App Service and trigger the first vulnerability. We created a video demonstrating this:

The vulnerabilities were immediately disclosed to Microsoft and fixed prior to this publication.

Introduction

Migration to the cloud has rendered old security practices largely obsolete, as system administrators must learn how to adapt and defend this new platform. Cloud security is still relatively new, making it essential to research and document new attack surfaces that arise when using these services. The infrastructure underneath is somewhat

undocumented in some areas, as opposed to plain Windows or Linux systems which have been largely scrutinized by security researchers.

In this post we'll present two vulnerabilities we have found in **Azure App Services**, specifically in the Linux App Services administration component called KuduLite, and cover technical details regarding how Azure App Service works.

Azure App Services

Azure App Services is an HTTP-based service for hosting web applications and is available in both Microsoft Azure Cloud and on-premise installations. We will be referring to the cloud version specifically.

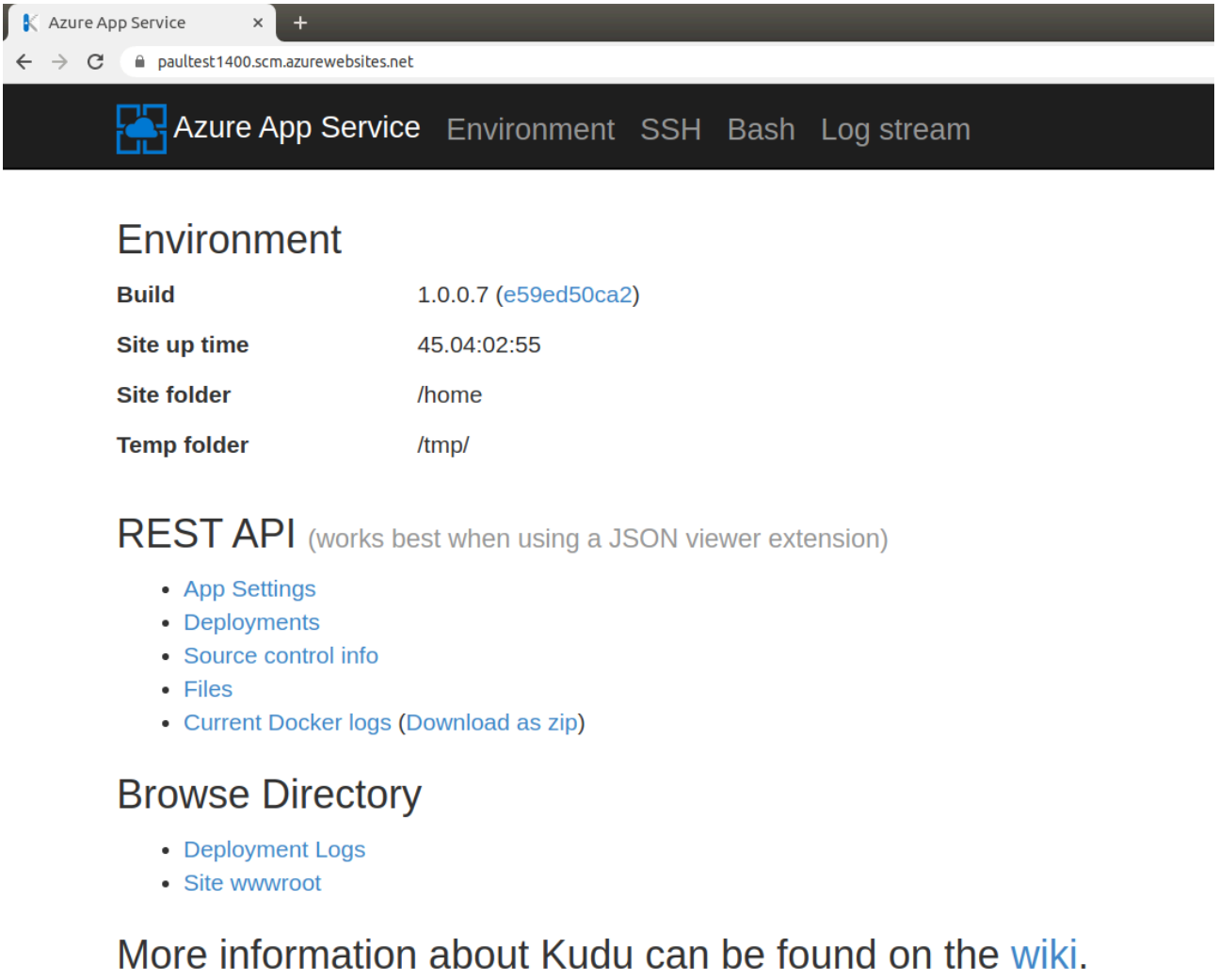
App Services is useful as it allows developers to simply write an application to serve HTTP and then push it to git. From there Azure will handle all pesky deployment details and provide an Azure-managed domain name.

To start using App Services, the user must first create an App Service Plan, which is the machine that App Services will use. This machine's main purpose is to host App Service containers.

Once a user creates an App Service, Azure creates a new Docker environment consisting of two container nodes: a manager node and application node.

Two domains are then registered:

- ***app.azurewebsites.net*** – pointing to the application's HTTP web server.
- ***app.scm.azurewebsites.net*** – pointing to the App Service's administration page provided by Azure.



The administration page is provided by a Microsoft open-source project called Kudu. For Linux, it's a lesser-known sibling project called KuduLite. The Kudu instance is hosted on the manager node, while the application itself is hosted on the application node. We will focus on the KuduLite variant.

The KuduLite instance offers the user diagnostic information about the system, including Docker logs, settings, and other environment information. If the user chooses to host the app's git with Azure, it is managed by this Kudu service.

Another useful feature is a web interface that runs interactive bash on the Kudu instance and an additional web interface to SSH into the application node (a separate Azure project named webssh).

The application inside the app node runs as root and we can SSH into it as root. When accessing the Kudu instance, however, we are given a low-privileged user:

```
Kudu Remote Execution Console
Type 'exit' to reset this console.
/home>whoami
a90b6e4e87d7503bc482aa84
/home>
```

This user is only meant to interact with /home and is unable to write to files in other directories. Interestingly, ClamAV is installed in this instance.

To recap, this figure describes the Linux App Services environment:


https://lh4.googleusercontent.com/ZzT_Hhskx4Q44VO5ypRVWPioICkyD4EacWCf54DYNokP4CwmYBcTtoDlI

Vulnerability 1: KuduLite Takeover/EoP

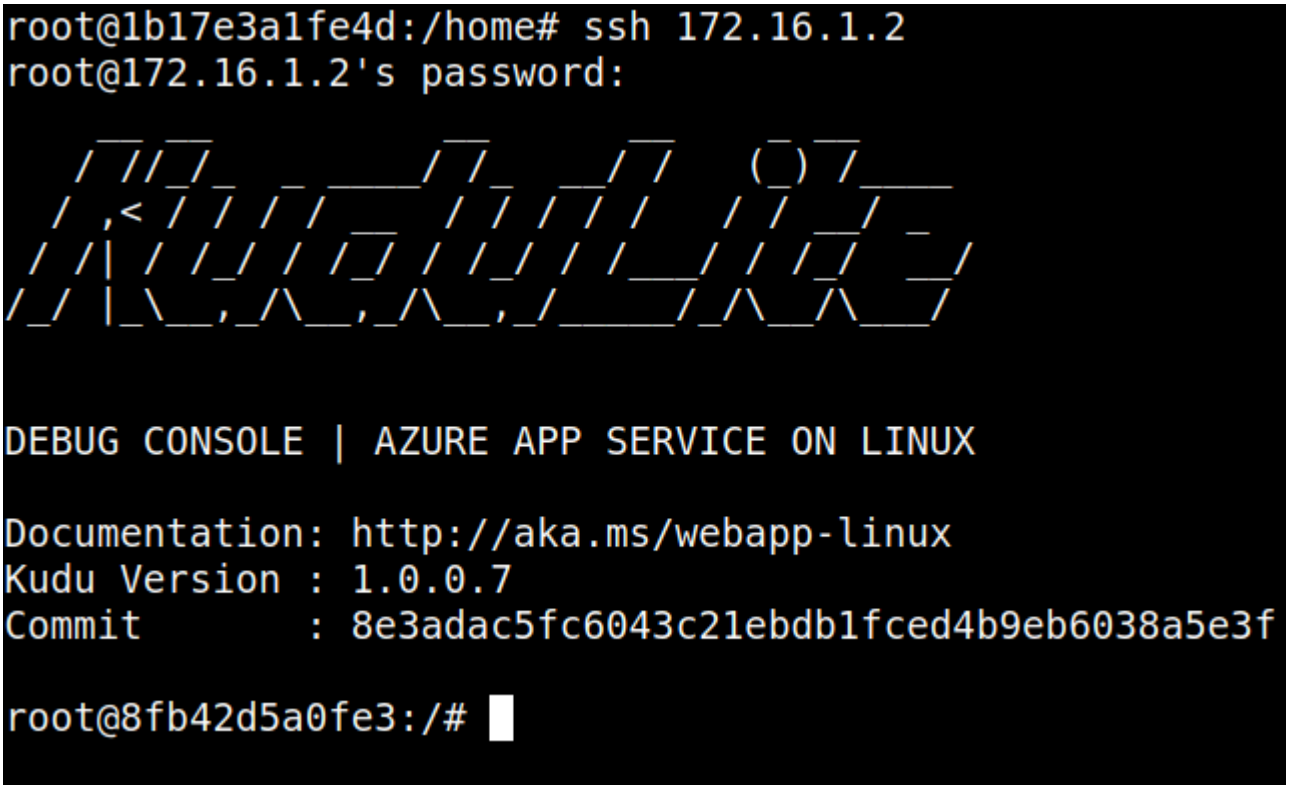
While investigating how webssh connects the web interface to the application node's SSH service, we noticed it uses hardcoded credentials "root:Docker!" to access the application node:

```
52      config.user.name = 'root';
53      config.user.password = 'Docker!';
54      if (req.originalUrl != null && req.originalUrl.includes("webssh/host")) {
55          if (req.query.debugconsolereq != null) {
56              config.ssh.host = "127.0.0.6";
57              config.ssh.port = 22;
58              config.ssh.kuduDebugReq = true;
59          } else {
60              config.ssh.port = 2222;
```

webssh credentials

This poses no danger since the application node's SSH port is not accessible from the internet.

We observed earlier the KuduLite instance also ran SSH, so we used the same credentials on the KuduLite instance and were able to log in as root:



As a reminder, the developers of the App Service KuduLite made sure admins were only able to log into it as a low privileged user, therefore we knew this was unintended.

Since we now controlled the KuduLite box, we had complete control over the SCM web server. We could listen to a user's HTTP requests to the SCM web page, add our own pages, and inject malicious Javascript into the user's web page.

Initially, we attempted to steal the SCM user's cookies from their requests to the server, however, we quickly found out there is an nginx intermediate that strips the cookies from the request before they arrive to KuduLite. Furthermore, the cookies had an *HttpOnly* attribute, which meant we weren't able to steal them with Javascript on the client's browser. These mitigations by Microsoft were very effective in limiting this vulnerability's potential damage.

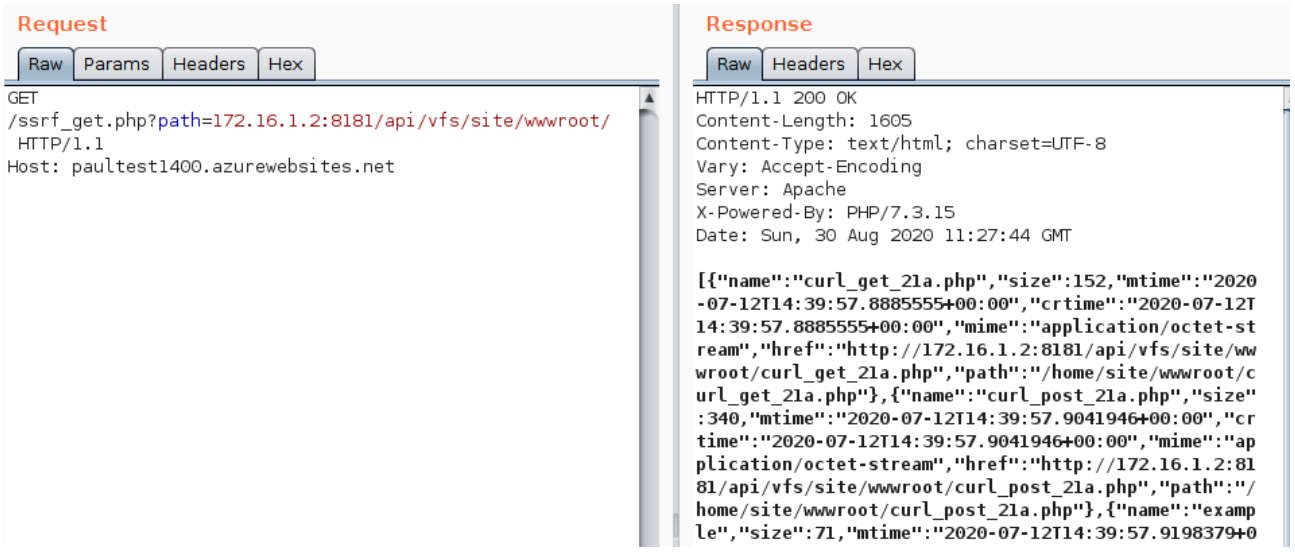
Despite the mitigations, an attacker could still inflict damage with this vulnerability and we presented such a scenario in the video in the **Main Findings** section, where an attacker uses the vulnerability to implant a phishing page in what is supposed to be the SCM web page.

The user may also choose to let App Services manage the git server, in which this will be handled by KuduLite. The attacker could then add malicious code to the repository to achieve persistence and spread to other instances using the same git server.

Vulnerability 2: Lack of Access Checks in KuduLite Allow Local File Inclusion or Remote Code Execution (LFI/RCE) to Attacker with SSRF

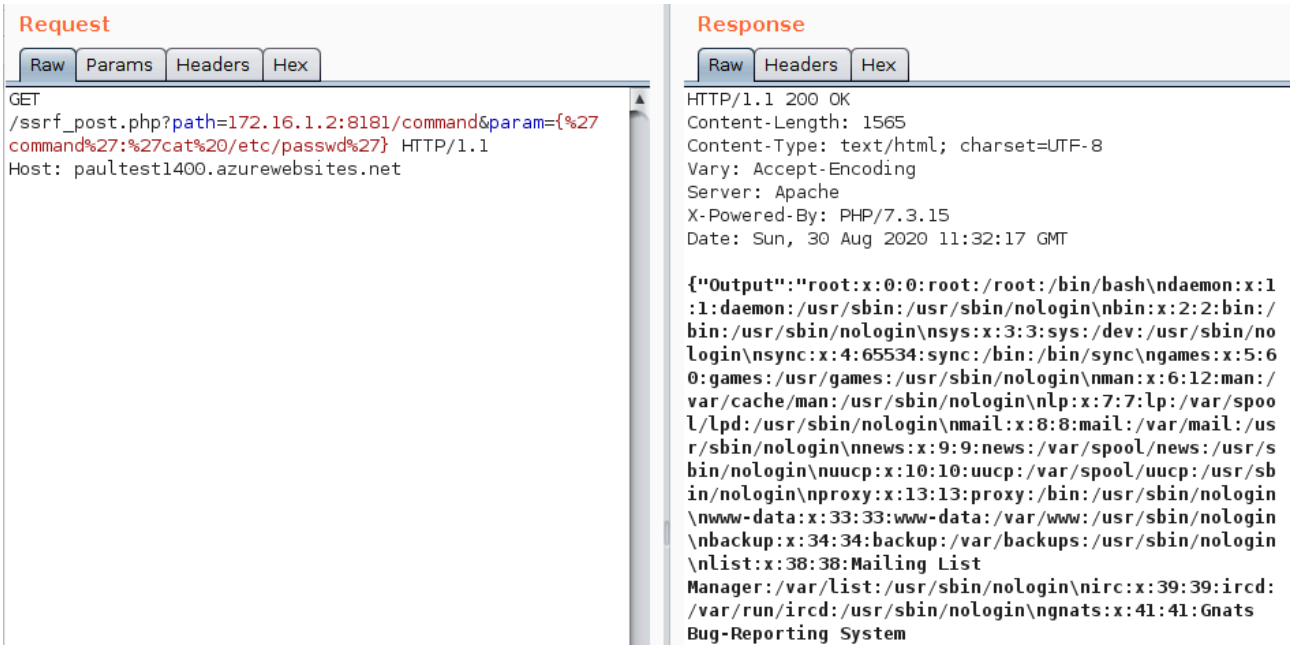
The second vulnerability resides in the KuduLite API, which is very similar to Kudu's API. The application node is able to send requests to the KuduLite API without requiring any access validation. This is especially problematic when considering a web app with an SSRF vulnerability.

An attacker who manages to forge a GET request may access the application node's file system via the KuduLite VFS API:



This would enable an attacker to easily steal source code and other assets on the application node.

An attacker who manages to forge a POST request may achieve remote code execution on the application node via the command API:



By contrast, in Windows (where Kudu is used), packets sent from the application node to the manager node are dropped.

Finally, these two vulnerabilities can be chained together, since once an attacker achieves code execution with the second vulnerability—provided they have an SSRF vulnerability—they can exploit the first one.

Conclusion

The cloud enables developers to build and deploy their applications at great speed and flexibility, however, often the infrastructure is susceptible to vulnerabilities out of their control. In the case of App Services, applications are co-hosted with an additional administration container and as we've seen in this post, additional components can bring additional threats.

We reached out to Microsoft with our findings as part of the responsive disclosure process and the vulnerabilities were quickly acknowledged and fixed.

As a general best practice, runtime cloud security is an important last line of defense since it detects malicious code injections and other in-memory threats that take place after a vulnerability has been exploited by the attacker.
