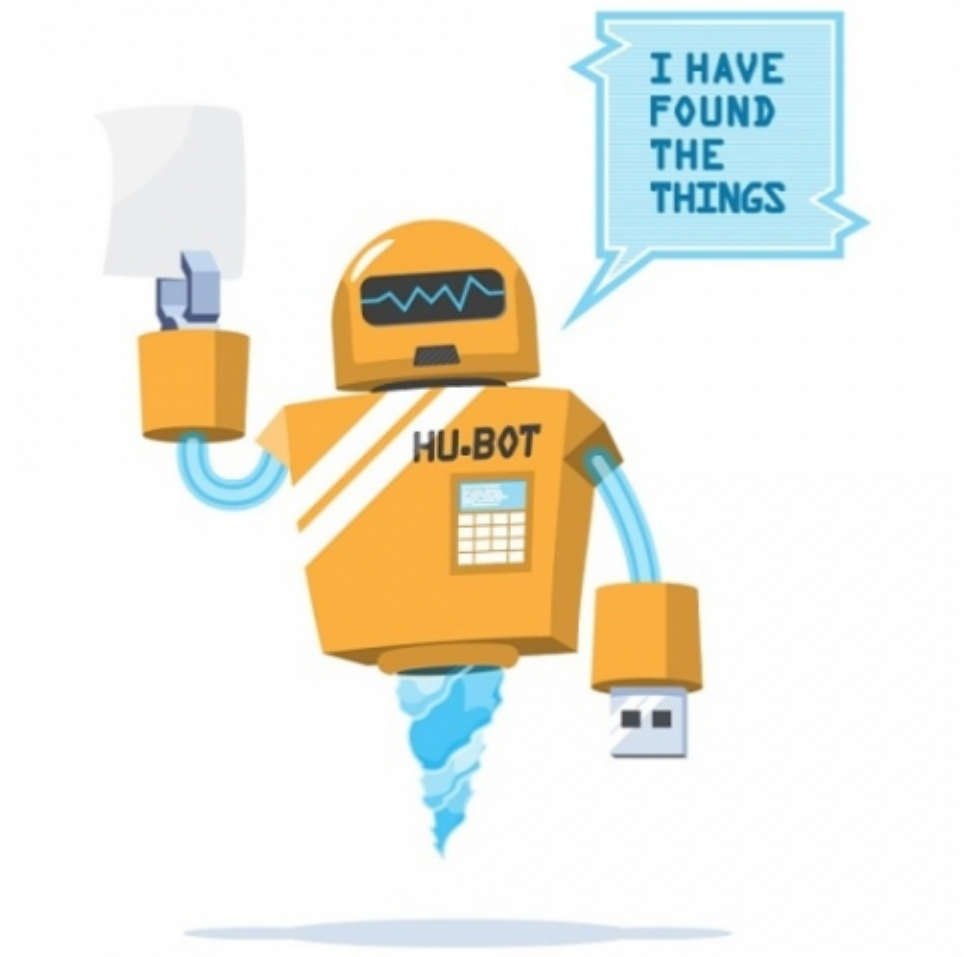
Why Exposed API Keys and Sensitive Data are Growing Cause for Concern

Internet security is a subject that is being covered more and more frequently by technology blogs and with good reason: the number of high profile internet security breaches has gone up substantially in the last few years. According to Semantec, 2013 saw a 62-percent increase in the number of breaches, a 91-percent increase in targeted attack campaigns, over 552M identities exposed via breaches, with 1 in 8 legitimate websites having a critical vulnerability. With these kinds of internet security threat statistics, internet security was a hot topic at many of the 2014 API conferences, including *ProgrammableWeb's* API conference in London (See SlideShare presentation by David Berlind, editor-in-chief of *ProgrammableWeb*).

Many of the high profile internet security breaches happened within the last year or so, some due to API security vulnerabilities. Companies that have had to deal with recent internet security breaches and/or API security vulnerabilities include Buffer, Pinterest, Facebook, and Snapchat. Many companies are now learning "the naked truth about internet security" and the lengths that hackers will go to to achieve their objectives. API security involves more than just securing the API itself: it involves protecting API Keys, cloud credentials, and other sensitive data from public exposure — security measures that are sometimes overlooked by developers.



Hackers have created an algorithm that continuously searches GitHub for exposed Amazon Web Services (AWS) API keys. The exposed keys are used to spin up hundreds of EC2 servers, which the hackers then use for bitcoin mining. - Image Credit: GitHub

Public Exposure of API Keys, Cloud Credentials is a Growing Problem

Public exposure of API Keys, cloud credentials, and other sensitive data is a serious problem that is happening more and more often. Developers are increasingly relying on cloud-based tools to automate building code and deployment of services, which is leading to far more instances of accidental public exposure of sensitive data.

Rackspace developer support engineer Kyle Kelley and Rackspace software security engineer Greg Anderson spoke at Def Con 22, explaining that developers are often too trusting of build pipelines. Build pipelines refers to a developer's source control (GitHub, Bitbucket, wherever code is being stored), continuous integration (the environment used for running tests), and upstream sources (npm, PyPI, RubyGems, etc.). Kelley and Anderson warn that when developers are too trusting of these build pipelines, credentials get leaked.

There are a lot of things that hackers can do with a developer's cloud credentials: spin up hundreds of servers, take down servers, "redistribute" DNS and load balancers, and much more. Accidental public exposure of credentials such as API keys, OAuth tokens, and app secrets is a mistake that can be made by both inexperienced and seasoned developers, particularly when it comes to source control. Right now there are thousands of exposed API keys on GitHub that can be found in just minutes using GitHub code search; these can be found in seconds by bots.

Bots are Constantly Scanning GitHub for API Keys, Credentials

One of the most recent examples of what can happen when a developer's API keys are accidentally exposed on GitHub can be found in a blog post titled "My $2375 Amazon EC2 Mistake," written by Andrew Hoffman. In the blog post, Hoffman explains how he had accidentally pushed code to GitHub that included his AWS API keys. Even though he had deleted the keys from GitHub within five minutes, bots had managed to obtain his API keys from that GitHub commit.

Hackers have created an algorithm that continuously searches GitHub for exposed API keys and credentials, AWS credentials in particular. Within the

five minutes that Hoffman's AWS API keys were publicly exposed on GitHub, hackers were able to obtain and use those keys to spin up about 140 EC2 servers on his AWS account. The servers were then used by the hackers for bitcoin mining. Five minutes of accidental API exposure on GitHub led to an AWS bill totaling $2375. Fortunately for Hoffman, Amazon customer service agreed to drop the charges.

Consider Your Data Compromised When You Push a Commit

When it comes to accidental exposure of API keys and other sensitive data on GitHub, GitHub states very clearly on the advanced Git help page that "once you have pushed a commit to GitHub, you should consider any data it contains to be compromised. If you committed a password, change it! If you committed a key, generate a new one." GitHub provides detailed instructions on how to purge a file from a GitHub repository's history.

Note: GitHub's warning refers to *public* repositories only, not private.

If your API keys, cloud credentials, or any other sensitive data has been accidentally exposed on GitHub, or any other public code repository/playground for that matter, take the necessary steps to remove that sensitive data from public view. Also be sure to generate new API keys and change all associated passwords immediately — don't wait.

Amazon Web Services is a Primary Target for Hackers, Secure Your AWS Credentials

Amazon Web Services is one of the cloud services companies most targeted by hackers, although just about every cloud services provider is a

prime target. Amazon takes security very seriously and the company provides tools that developers can use to securely control access to AWS services and to monitor AWS cloud resources and applications running on AWS.

AWS Identity and Access Management (IAM) is a free service that allows developers to control access to AWS services. Developers can use this service to create IAM roles, grant user permissions, generate temporary security credentials, use IAM roles for EC2 instances, reduce/remove use of root, and much more. Amazon CloudWatch can be used by developers to monitor AWS cloud resources and applications running on AWS. Developers can use the CloudWatch service to monitor EC2 instances, monitor Elastic MapReduce job flows, set up automated alerts, and more. One of the most useful features of Amazon CloudWatch is the ability to monitor AWS costs using billing alerts and notifications. When AWS usage exceeds the set thresholds, an email is sent to notify the AWS account holder.

The Amazon Web Services site also provides a lot of information regarding IAM best practices and best practices for managing AWS access keys. If you are a developer building applications that run on AWS, the AWS documentation for security credentials and IAM best practices is well worth reading.

Don't Let Hackers Gain Access to Your API Keys, Sensitive Data

Andrew Hoffman's blog post about what happened when his AWS API keys had been accidentally exposed on GitHub sparked conversations on several tech blogs and on reddit. There were a lot of suggestions by

developers on how to prevent API keys from being exposed on GitHub, tips on using AWS, and advice in general, which include:

- **General Advice**
  - Never commit API keys, passwords, secrets, and other sensitive data to a code repository (public or private).
  - Don't embed API keys, passwords, etc., directly into code, always keep these separate. Put API keys, passwords, etc., in a separate configuration file. If a configuration file is part of a project in an IDE, remove sensitive data before distributing or sharing.
  - Store API keys, passwords, secrets, and other sensitive data in a secure system with strong encryption. System should also utilize immutable audit logging (IAL).
  - Use a private GitHub repository or Bitbucket. Always keep the repository private.
  - Tutorial/education code should be developed locally on your own machine. Pushing commits to GitHub or using a cloud service like AWS should only be done when necessary, rarely for tutorial/education code.
- **Using AWS**
  - When using AWS, set up billing alerts and max budgets.
  - Never use root credentials with AWS, use limited access keys through IAM instead.
  - Use temporary security credentials instead of long-term access keys with AWS. This is done by creating IAM roles.
  - Use different access keys for different applications and rotate access keys periodically.
- **Using Git / GitHub**
  - Always double-check a commit before pushing to a server.
  - Always run Git status and Git diff before pushing a commit.
  - Scrub code completely before pushing a commit.

- Avoid using catch-all commands like git add with “-a,” “-A,” or “.”
- Use git-crypt. Git-crypt enables transparent encryption and decryption of files in a Git repository.

Conclusion

Hackers are using sophisticated algorithms and innovative hacking techniques to quickly exploit vulnerabilities in technology platforms and the APIs powering them. An algorithm and a five-minute API key exposure on GitHub was all it took for hackers to gain access to Andrew Hoffman’s AWS account and run up a $2375 bill.

Perhaps GitHub could improve the platform’s code search function so that API keys, secrets, etc., are not displayed in search results or searches for sensitive data like API keys are prevented altogether. Ultimately, it is the developer who is responsible for the security of his projects and the protection of API Keys, cloud credentials, and other sensitive data.

Never underestimate the tenacity and guile of hackers; consider your data compromised the second you push a commit. Always take the steps needed to prevent accidental exposure of sensitive data.