Athabasca University

OpenBullet

Credential Stuffing for Script Kiddies and Career Criminals

Philip Kirkbride
3060988
COMP 601
Richard Huntrods
September 29, 2019

Abstract

OpenBullet is an open-source testing tool which has been growing in popularity in the amateur offensive security community. It is commonly used for credential stuffing. The application has grown around an underground economy which focuses on the combination of configuration files and credential dumps. In this paper I will discuss the environment that fostered the application, its use, and defensive measures which can be taken to protect against it.

Password leak lists are bought and sold on black markets, as well as configurations files for the program, as well as large lists of proxies which can be used to disguise the origin of the attack. Using OpenBullet everything that is needed to successfully conduct a credential stuffing attack can be purchased with very little or no technical knowledge being needed, aside from learning how to use the OpenBullet user interface.

The modularity and ease of use has bolstered the black-market. Several forums and non-public chats have sprung up, revolving around buying, selling, and giving tips related to everything needed to conduct credential stuffing attacks.

As this trend continues communities, governments, and website operators are looking for more effective ways to fight this trend. Solutions include implementing two-factor authentication, rate-limiting, captchas, and alerting systems for both users and website operators.

OpenBullet

A credential dump is created when a large website has user emails and passwords leaked, for example the 2012 LinkedIn hack which saw the leaking of nearly 6.5 million email password combinations. These lists become a raw resource for the underground economy, sold in bulk to other criminals who attempt to create value by exploiting the lists for financial gain. One of many vectors of attack used by these criminals to create financial gain is to attempt to use these passwords on other sites where purchases can be made for example sites like Amazon, banks, or even lower value targets such as Netflix (allowing unauthorized users to stream videos).

Of course testing even a fraction of these massive lists manually takes a very long time. It is no surprise that various forms of automation have been used. However due to the nature of the web and the multitude of applications which are targeted, often such automation scripts would be written in large part from scratch for each target. OpenBullet has emerged as an open-source program which takes a configuration file, either written in a domain-specific language called loli-script, or built with a GUI editor (creating loli-script in the background). For example a configuration file might be specifically written for McDonalds.com which when combined with a leaked password list will return a list of valid email/passwords for the website. These account combinations are in turn sold to lower level criminals who either resell or use them maliciously.

OpenBullet configuration files can then be shared, edited, and modified to target other sites or applications. The popularization of OpenBullet has significantly lowered the barrier to entry for credential stuffing and presents a significant threat to the public. A criminal with only basic knowledge of computers can buy a password list and a OpenBullet configuration file and combine them to get a list of account credentials without having to ever know anything about programming. With a little more knowledge configuration files can be created or modified to target other sites (a legal grey area in not directly used) these new configuration files are then distributed or used thus spreading the use of OpenBullet further.

One of the major barriers to credential stuffing is the use of rate limiting, that is limiting the amount of requests that can be made from a single IP [1]. As a way around this OpenBullet allows users to specify a list of proxies which disguise the origin IP of the attack. A large attack for example may make use of 1000 proxies from different geographical locations. Every few requests the program will change the proxy it's using thus appearing to be a completely different user.

Another common form of defence is the use of a captcha service, that is a small puzzle which pops up before the user can submit a request to login. However these too are commonly circumvented [1]. OpenBullet provides support for several captcha solving services including 2Captcha (prices currently starting at $0.87 per 1000 images) and Anti-Captcha (prices currently starting at $0.50).

Both of these captcha services employ real humans to solve these captchas. According to the 2Captcha website workers can expect to make $0.20-$0.80 per hour solving captchas. Other solutions using AI or creative solutions also exists, for example uncaptcha2 (no longer works) used Google's audio version of captcha (intended for the hard of hearing) and fed the results into a voice to text to solve captchas [2].

It should be noted that even though rate-limiting and captchas do not fully stop credential stuffing attacks, they do significantly raise the barrier to entry and cost. Proxies and captcha solving services require more technical knowledge to setup and increase the cost for the attacker (paid even if the attack is unsuccessful). Given the choice to target a site with rate-limiting and captchas or one without either, attackers will choose the later.

We can consider the economics by looking at the sale price of accounts to end-users or the value gained by a hacker using said account themselves. As an example by searching through private groups on Telegram I was able to find people selling Chipotle accounts for $3-$8 (credit card attached can only be used to order food). Presumably end buyers will use them for a cheap meal, likely racking up a charge between $20-$100.

By going to Chipotle login page we can see they don't currently employ a captcha service. It's not possible to test how many login attempts can be made per IP without launching an attack but given the widespread buying and selling it is likely that rate-limiting is lenient.

As mentioned previously rate-limiting and captchas can raise the barrier required for an attack but do nothing to stop them. When accounts contain something of value such as a

financial institution or high-value store account other methods of security should be used to prevent attacks from applications like OpenBullet. The use of any type of 2FA (two factor authentication) prevents these types of attacks completely.

To reduce the cost for both the user (in time) and website (in resources) the 2FA can be triggered upon a login attempt from a new device or IP. For example a website might require a user to click a link from their email when accessing the account from a new location or device. Despite the many features of OpenBullet it has no way of dealing with such a scenario.

2FA is an easy solution for a website owner looking to provide security, but what about individuals, governments, and other community organizations who want to protect citizens, but have little or no control over the websites they use? To prevent people from falling victim there needs to be a general education about password reuse and updating of passwords over time.

For this other methods need to be used to fight the rising popularity of programs like OpenBullet. Generally the most effective methods are education and early alert systems. Education focuses on teaching users the importance of not reusing passwords and updating passwords over time. For example it has been found that “... just-in-time fear appeals decrease password reuse; 88.41% of users who received a fear appeal subsequently created unique passwords, whereas only 4.45% of users who did not receive a fear appeal created unique passwords” [3].

The use of alerting systems can be used to target website operators or end-users. For example the website haveibeenpwned.com allows users to check if their email has been included in any popular password leaks. In the same way some services or individual activists have attempted to email such users directly suggesting that they change their passwords.

Amazon has taken an initiative with its AWS Cognito service which provides user-authentication management as a service. If a developer enables advanced security features they will see an option which says “You can detect and protect your users from using credentials that have been exposed through breaches of other websites”, by enabling this feature a user who uses an email/password combination that has previously been exposed will be prompted to choose another password.

The AWS Cognito option is left off by default in part because login systems want to be as quick and seamless as possible. This is doubly true with ecommerce, someone who is making a “splurge” purchase, for example expensive shoes, may second guess themselves when confronted with the request to be “responsible” with password reuse.

It is likely that issues related to credential stuffing can only be solved from the website operator perspective. Despite education and alerts end-users continue to make the same mistakes repeatedly. For high value sites two factor authentication should always be implemented, at a minimum for logins from a new device or location. When possible website

operators should alert users when they try to create an account using known breached email/password combinations, as has been implemented on Amazon’s login management system Cognito.

Citations

1. "I want my money back!" Limiting Online Password-Guessing Financially, Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), 2017, Santa Clara, CA, https://www.usenix.org/conference/soups2017/workshop-program/way2017/golla, USENIX Association
2. "ecthros/uncaptcha2", GitHub, 2019. [Online]. Available: https://github.com/ecthros/uncaptcha2. [Accessed: 29- Sep- 2019]
3. Jeffrey L. Jenkins, Mark Grimes, Jeffrey Gainer Proudfoot & Paul Benjamin Lowry (2014) Improving Password Cybersecurity Through Inexpensive and Minimally Invasive Means: Detecting and Deterring Password Reuse Through Keystroke-Dynamics Monitoring and Just-in-Time Fear Appeals, Information Technology for Development, 20:2, 196-213, DOI: 10.1080/02681102.2013.814040