SEE TEXT ONLY

Protect Your Secrets

Do you create complicated passwords? How hard is it for you to remember the new ones? Do you keep your credentials on a Post-it note near your monitor? If you do any or all of the above, we have an alternative that will help protect your privacy and make it easier for you to stay safe.

YubiKey Overview

A lot has been said about YubiKey, which is produced by Yubico. It has become one of the most popular solutions offering a secure 2FA — Two-Factor Authentication. It also may be used as a secondary password factor U2F — Universal 2nd Factor. It offers strong authentication and is easy to use.

YubiKey is a USB-like device. In the simplest use-case, when we connect YubiKey to a computer, it is detected as a keyboard in the operating system and the device allows us to store up to two passwords. Depending on how long we press the button on the device, it will release the first or the second password. In this simple use-case, we can use it to integrate with almost any web service.

In this scenario, we can use it to remember our passwords and forget about using Post-it notes. For more advanced users, this scenario can help protect their data. Rotating passwords is difficult, as it is problematic to remember new ones. Now, we can rotate our most often used key (for example access to our password vault) and not have to bother remembering it (although it is always a good idea to have backup). While keeping our primary password on the YubiKey device, we can still use other devices (such as a mobile phone) as a second factor.

By Jarosław Zurek,
Michał Borysiak,
and Mariusz Zaborski

Authentication Methods

Besides storing static passwords, some models of YubiKey also support more sophisticated authentication methods such as:

- **One-time Password (OTP)** – authentication mechanism, generating passwords that can be used once.
- **OATH – HOTP** – event token, generating 6- or 8-character OTP passwords using the HOTP algorithm.
- **OATH – TOTP** – 6- or 8-character OTP passwords, the TOTP algorithm is based on time function.
- **PIV (Personal Identity and Verification)-Compatible Smart Card** – enables the use of private RSA/ECC keys stored on YubiKey for signing and decryption. This mode works like a smart card.
- **OpenPGP** – encryption and signing using RSA and ECC private key, stored on YubiKey, using standard suits like PKCS #11.
- **U2F** – an open authentication standard that enables secure access to any number of online services. Only one single device, without additional drivers, or client software.

It is possible to use two-step verification with web services like Google, Facebook, GitHub, and Hotmail. This is useful because even if attackers steal the first factor of authentication (username and password to the account), they are not able to log in. We can ensure this by using a YubiKey device which supports U2F. Of course, we must enable two step verification on a chosen web service.

Models

There are a few different types of YubiKeys that can be used for various purposes. The most basic version is the *FIDO U2F Security Key*. It supports static password authentication and may be integrated with the most popular applications like Gmail and Facebook using U2F. It does not support methods like HOTP or OpenPGP. This device is the most affordable product they offer at the time of writing this article and the cost is US $18.

More advanced models, known as the YubiKey 4 generation (which also includes YubiKey 4C, YubiKey Nano and YubiKey 4C Nano) already support basic cryptographic methods like OTP or OATH modes. Differences are in device size and USB port type. Their use is definitely wider than the previously mentioned model and some examples will be explained in detail later in the article.

Another available option is YubiKey NEO. An additional feature supported by this model is communication via NFC. For example, after tapping the device, a smartphone can read OTP emitted by YubiKey. A summary of differences can be seen in Table 1.

	FIDO U2F Security Key	YubiKey 4 generation	YubiKey NEO
OTP		✔	✔
OATH – HOTP		✔	✔
OATH – TOTP		✔*	✔
OpenPGP		✔	✔
U2F	✔	✔	✔
Secure Element	✔	✔	✔
Smart Card (PIV)		✔	✔
Supports NFC communication			✔

**Requires additional app (lack of built-in Real Time Clock).*

Table 1: Comparisons of different YubiKey models

FreeBSD Tooling

YubiKeys come with a set of open source tools that are necessary to integrate YubiKey in a Unix-like environment. For FreeBSD users most of these tools are available in the ports collection as well as in the binary package repository. Two of the most interesting packages are `security/ykpers` and `security/yubico-pam`. The `security/ykpers` package contains several command line tools to manage YubiKey:

- `ykpersonalize(1)` —is necessary to program YubiKey; it handles configuration options for almost every YubiKey feature.
- `ykinfo(1)` —is useful for retrieval of basic information about YubiKey.
- `ykchalresp(1)` —allows for signing data using YubiKey in a challenge-response mode of operation.

GELI Full Disk Encryption

GELI is the most popular disk encryption method on FreeBSD. One of the most secure ways of using GELI and FreeBSD is to use two-factor authentication with a passphrase and a key file. The key file is kept on a memstick. When we boot our machine, we need to provide both factors. After decrypting our device, we

remove the memstick with the file. In that way, if somebody steals our computer they will also need to see our password and steal our memstick. If we want to be even more paranoid, we can keep a kernel and a key file on the memstick. In that way, even if somebody has access to our computer, it makes it impossible to integrate with our software. An intruder could still alter our hardware, but this is much harder.

FreeBSD has recently begun supporting full disk encryption, which means that even a kernel is encrypted—only the small boot loader partition is not. Unfortunately, in the current implementation, we do not support a key file, so we can only use a passphrase to encrypt our disk. Thanks to YubiKey, which can be detected as a simple keyboard, we can use it to provide a passphrase during boot. Using it in that way, we lose one factor. We can mitigate this by using a passphrase which we provide using a normal keyboard and a second part of the password which is much longer and is kept on YubiKey. In that scenario, somebody not only would need to see what passphrase we are typing, but also would need to steal our YubiKey.

To make the device to meet these requirements, we program the second slot of Yubico in static mode:

```
$ ykpersonalize -2 -o static-flag -o append-cr
```

Thanks to the `append-cr` flag we do not have to press Enter after each use of this slot. In the case of GELI, we would first provide our passphrase and then the second factor using YubiKey. By default, there is no output when typing a password in GELI. In our case, we would see that the passphrase was submitted because the factor provided by YubiKey would contain trailing ENTER.

Integration with FreeBSD Login

Yubico provides a PAM module which can be deployed within existing authentication systems. The module can be found in the `security/pam_yubico` package and it works in two modes: online and offline authentication. The former provides a second factor based on one-time passwords, but it delegates authentication to Yubico cloud services. Therefore, it requires a stable internet connection. On the other hand, it is very easy to setup a newly bought Yubico as a second authentication factor for your system account.

Every device has a secret on its first slot preset by the manufacturer. The slot works in so-called Yubico OTP mode. Each password generated from this slot consists of a static 12-character part and the remaining dynamic part. The static part never changes and it can be considered the device's public identifier. These factory defaults are already known by Yubico cloud services. All that we need to do is to retrieve the API token and the user ID using a special form available on Yubico's website.

The second mode of operation is more practical. It requires a YubiKey to work in a challenge-response mode when the device can be issued to sign data sent by a user application. In this particular example, our application is the Yubico PAM module.

First, we have to program the device to work in a challenge-response mode. In the example below, we will use the first slot:

```
$ ykpersonalize -1 -o chal-resp -o chal-btn-trig -o chal-hmac -o hmac-lt64 -o serial-api-visible
```

We want the device to wait for a user confirmation of each operation which is guaranteed by the `chal-btn-trig` flag. The user has to press the key located on the YubiKey while logging into the system, unfortunately twice. The first press is needed to check whether the challenge located in a file matches the device response. If so, the second press generates a new challenge-response pair and stores it for later use. We need to generate an initial challenge which is stored by default in `~/.yubico` directory:

```
$ ykpamcfg -1 -v
```

The next step is to configure the PAM module. We want to use the second factor together with a static password to protect any login to our computer. For this purpose, we will modify the `/etc/pam.d/system` file so the "auth" section will look as follows:

```
# auth
auth    required    pam_unix.so        no_warn try_first_pass
auth    required    /usr/local/lib/security/pam_yubico.so mode=challenge-response
```

Now issue the `sudo -s` command, type the static password and press the button on the device. It waits for a user action up to 15 seconds and an LED indicator is blinking during this time. Authentication will fail either when the user does not take action or if YubiKey is not connected to the USB port.

Integration with SSH

Another useful application of YubiKey is strengthened remote authentication to an SSH service. A commonly described method utilizes the Yubico PAM module using the online mode of operation. However, a Yubico token is compliant with OATH-HOTP standards and, therefore, it can work with any authentication server which supports this standard. For example, Wheel Cerb AS is such a multi-factor user authentication solution. We will use the `pam_oath` module which is included in the `security/oath-toolkit` package and does not require a connection to any external cloud service.

We need to reprogram our YubiKey to support OATH mode. Unfortunately, we have only two slots on our device so we need to overwrite one of the previous configurations:

```
$ ykpersonalize -1 -o oath-hotp -o oath-hotp8 -o append-cr
```

We want to use 8-digit long, counter-based passwords. The output of the command should contain a line with a key in hexadecimal form:

```
      ...
key: h:c621245c5f05eefec1d9f2960f34b865849dd074
  ...
```

We need to store the user's name and the key in the `pam_oath` database file on the target machine (of course "alice" and the key should be replaced by real values):

```
$ echo "HOTP alice - c621245c5f05eefec1d9f2960f34b865849dd074" >> /usr/
local/etc/users.oath
```

The next step is to modify the `sshd` PAM configuration in order to enable the `pam_oath` module. We can achieve this by modifying the `/etc/pam.d/sshd` file so the "auth" section will look as follows:

```
# auth
…
auth        required      pam_unix.so          no_warn    try_first_pass
auth        required      /usr/local/lib/security/pam_oath.so usersfile=/
                          usr/local/etc/users.oath window=16 digits=8
```

We need to ensure the `sshd` configuration which is placed in the `/etc/ssh/sshd_config` file contains a few options set to the appropriate values as follows:

```
ChallengeResponseAuthentication yes
PasswordAuthentication no
UsePAM yes
```

Finally, reload `sshd(8)` service and try to login to the remote server typing the static password and then using the dynamic password from the token.

Conclusion

YubiKey is a remarkable device that can be used in a corporation or by individuals to increase their security. This small device supports many different authentication methods and can be used with many popular web services as well as with programs like SSH. It also allows us to leverage some of the imperfections of tools that do not support a 2FA. It can be used as a second factor or to keep the primary password. It is an interesting alternative to other solutions like mobile applications. YubiKey also allows us to painlessly change our passwords without the need for any memorization. ●

JAROSLAW ZUREK is a software developer at *Wheel Systems* where he supports a project creating a privileged session manager. He is interested in cryptography, TLS, and low-level/hardware programming.

MICHAL BORYSIAK is a software developer at *Wheel Systems,* where he works on centralized authentication systems. He is fascinated by low-level operating system concepts, networks and cybersecurity.

MARIUSZ ZABORSKI is a lead software developer at *Wheel Systems*. He has been a proud owner of the FreeBSD commit bit since 2015. Mariusz's main areas of interest are OS security and low-level programming. At *Wheel Systems*, Mariusz leads a team that is developing the most advanced solution to monitor, record and control traffic in an IT infrastructure. In his free time, he enjoys blogging (http://oshogbo.vexillium.org).