Graham Dungan
January 30, 2025
UIN: 332001764

CSCE 413: Software Security
Self-Study: GoTo Fail

What is the Vulnerability?

In 2012, Apple released an update for IOS 6.x and OS X Mavericks in September of 2012 [2]. This update upgraded some SSL security features seen in files like `sslKeyExchange.c`. This file contained a call to a function `sslRawVerify` to verify the validity of a SSL certificate. This code, however, had a misplaced `goto` statement that rendered the call to `sslRawVerify` unreachable. What follows is an abridged code segment from a function `SSLVerifySignedServerKeyExchange` detailing the vulnerable code, obtained from GitHub @Pharap;

```
1 if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
2     goto fail;
3     goto fail;
4 err = sslRawVerify(...)
5 fail:
6     return err;
```

The error within this code is the extra `goto fail;` on line 3. Despite the indent, the code executes regardless of the if statement, such that the `sslRawVerify` is never executed. If the variable `err` passes the initial if statements, it will be a value 0. Because `goto fail;` is then called on line 3, and line 6 returns `err`, `SSLVerifySignedServerKeyExchange` will always return 0, indicating no error has occurred and that the SSL credential was verified.

What was the Root Cause?

It is a fact that the vulnerability was caused by the extra `goto fail;` statement on line 3. It is generally accepted that the vulnerable line itself appeared due to a previously deleted if statement between lines 2 and 3, or a mismanaged merge between two developers' code [1]. The root cause of this vulnerability was an erroneous `goto` statement that resulted in an unreachable code segment which was pivotal to the validation of SSL certificates [6, 4].

What is the Extent of its Impact?

Affected systems included any machines running IOS 6.x-7.x and OS X Mavericks from September 2012 to February 2014 [2]. The vulnerability was fixed days after it was first disclosed, and patched in IOS 7.0.6 on February 21, 2014. The bug affected any services on these devices that relied on SSL/TLS for security, as all certificates were essentially considered valid. Because of this, man-in-the-middle attacks could be easily staged on Apple devices. Since man-in-the-middle attacks are mitigated by encryption and certification, the failure of `sslRawVerify` allowed attackers to intercept traffic and to act as some authority. Since Apple patched this error quickly, there were no documented cases of this vulnerability being exploited.

How to Patch it?

The vulnerability was removed by deleting the erroneous `goto fail;` statement on line 3. The vulnerable code was infuriatingly simple and did not require much recourse. Other simple fixes would have included adding brackets to portions of the program or forgoing `goto` statements entirely, but this is a preventative measure rather than a patch.

How Could it Have Been Prevented?

Programmers have largely debated `goto` statements as potentially dangerous and syntactically confusing logical flow operators. Edgar Dijkstra has considered the `goto` statement a feature that fails to bridge the conceptual gap between the logic of static programs and dynamic processes [3]. Other programmers disagree, stating that the perceived danger from `goto` statements is due to conceptual design failures, rather than the statement itself [1].

Besides the notorious `goto` statement, many others have noted how code formatting, the use of brackets, compiling with `-Werror -Wall`, negative testing, defaulting the `error` variable to nonzero number, and manual review were all valid preventative measures that were seemingly skipped [5].

References

[1] Wheeler, A. D. (2014). *The Apple goto fail vulnerability: Lessons learned.* David A. Wheeler. Retrieved February 2, 2025, from `https://dwheeler.com/essays/apple-goto-fail.html#goto`

[2] Bland, M. (2014, June 3). *Goto fail, Heartbleed, and unit testing culture.* martinfowler.com. Retrieved February 2, 2025, from `https://martinfowler.com/articles/testing-culture.html#:~:text=The%20%E2%80%9Cgoto%20fail%E2%80%9D%20bug%20first%20shipped%20to,all%20communications%20between%20the%20other%20two%20systems.`

[3] Dijkstra, E. (1968). *Go To Statement Considered Harmful.* Communications of the ACM, 11(3), 147–148. `https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf`

[4] Fexl. (2014, March 4). *GOTO (Still) Considered Harmful* [Online forum post]. Hacker News. Retrieved February 2, 2025, from `https://news.ycombinator.com/item?id=7339800`

[5] Hacker News. (2014). *The Apple goto fail vulnerability: Lessons learned.* Retrieved February 2, 2025, from `https://news.ycombinator.com/item?id=25801019`

[6] Van Deursen, A. (2017, April 17). *Learning from Apple's #gotofail Security Bug.* Arie Van Deursen. `https://avandeursen.com/2014/02/22/gotofail-security/`