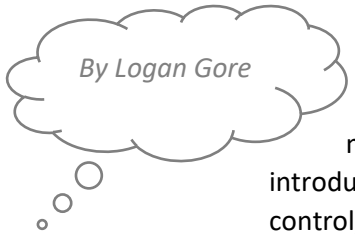
Why GMAC Solutions Should Be Presumed Insecure By Default



June 2, 2015

"The issue of nonce control in GMACs can't even be looked at as a best practice; it's a must practice that makes GMAC a fragile solution."

GMAC is a NIST standardized message authentication function that is designed for 128-bit block ciphers. GMAC is thrown around in forums and listed under product features as a most secure way to provide message authentication. *The cool kids are using GMAC*. Besides some hype, GMAC is popular because it's efficient in hardware and software and when used properly is a reasonably secure authentication mechanism.

A GMAC takes three values as input, rendering it fundamentally different from other authentication functions like CBC-MAC, CMAC, and HMAC. GMAC needs a key, a message, and a nonce. The nonce is used in generating the IV for GMAC's internal encryption function which, in most implementations, runs in CTR mode. [Cryptography Engineering]

The critical breakdown in GMAC security occurs upon nonce reuse. Because nonce reuse in GMAC relates directly to IV reuse, it exposes a vulnerability that can cripple the entire function. Recall the IV issue with WEP encryption in wireless systems.

This problem isn't even with GMAC, per se, or its process, but rather with how developers implement it. Most developers won't do what is necessary to prevent GMAC from breaking down. The extra measures that must be correctly introduced to handle nonce generation and control, make GMAC a more expensive and error prone implementation than simpler solutions, like HMAC. All too often, developers have little incentive to deal with the kind of hassle and additional overhead that secure nonce management can present. Therefore, solutions that run GMAC should be given a presumption of insecurity until it's determined that there is a *secure* system in place to handle the nonces. Still, even with sound nonce control, GMAC should only be seen as effectively providing 64 bits of security [Cryptography Engineering]. This falls short for engineers trying to maintain a true 128-bit (or greater) security level.

The issue of nonce control in GMACs can't even be looked at as a best practice; *it's a must practice that makes GMAC a fragile solution.* Developers looking to avoid the common pitfalls associated GMAC should consider an HMAC, like HMAC-SHA256, for a more secure, simple, and robust message authentication function that is not fundamentally dependent on a secure nonce system.

References

Schneier, Ferguson, Kohno. *Cryptography Engineering: Design Principles and Practical Applications.* Wiley Publishing, Inc. 2010.

About The Author

Logan Gore is a software engineer who has published several Open Source projects for C#/.NET and papers on current industry topics such as enterprise architecture, cryptography, and Big Data. He holds professional certifications in enterprise security, software and database development, cloud technologies, networking, and project management. Logan is currently involved in cloud, game, and mobile development.

loganlgore@gmail.com

<=512

Logan Gore's *<=512* is a series of articles and short papers addressing programming and IT topics in 512 words or less.