1 - First Vulnerability (Reference 1)

Overview (abstract, reference 1): The research paper throws light onto how Diffie-Hellman key exchange algorithm was found to be less secure than expected. This algorithm is primarily used for SSH and IPsec, also a viable option for Transport Layer Security (TLS). The statistics supporting the vulnerability are from the abstract of the paper and from *weakDH.org* where 82% of the vulnerable servers used a 512-bit group for export-grade cryptography (refers to reduced, breakable key size). This constitutes about 8.4% of the Top 1 million domains. Looking further into the 768-bit and 1024-bit primes can be broken by an academic team and a nation state respectively.

Vulnerability (Section-2 and figure-1, reference 1): The most efficient technique to attack Diffie-Hellman is based on compromising either of the private exponents by computing discrete log of the public value. This is in fact difficult for a single entity of discrete log. But the authors show that by largely precomputing for prime 'p', arbitrary discrete logs can be calculated at ease. The cryptanalysis used in the paper consists of 2 parts. The first being the precomputation that consists of the number field sieve algorithm (used to find the all prime numbers upto a given limit). The authors composed the attack by diving the precomputation into 3 stages: 1) Polynomial selection – finding polynomial f(z) for computation, it scales well and usually doesn't take a lot of runtime. 2) Sieving – tries to find relations between the elements. Each sieving checks for a possible 2^{2I} combinations ('I' being the parameter). This scales up well but is very expensive to compute because of the number of elements. 3) Linear algebra – builds a large sparse matrix of all coefficient vectors of prime factorizations found. Here, a kernel vector gives logs of the smaller elements and these serves as input for a further stage called descent. The prime focus here is that only the descent stage would be discrete for each attack and the previous stages can be used to target many outcomes. There can also be some flexibility on each of the precomputation stages by introducing a trade-off between the 3 stages.

Description of attack (section 3.1, reference 1): The attack on TLS is based on Logjam – which allows the man-in-the-middle to downgrade (break keys into smaller divisions) vulnerable connections. This exploits a flaw in the TLS protocol and affects modern web browsers, explained as follows. The TLS protocol started with a handshake between the two parties for deciding on cipher suite to use for communication. The client and server ensure an agreement and calculate a message authentication code (MAC) of the handshake history. In the 1990's, the U.S export restrictions on cryptography insisted on using DHE_EXPORT cipher suites being no longer than 512-bits. However, many servers maintained two groups i.e. 1024-bit key for regular key exchange and 512 for DHE_EXPORT. The attacker writes his own 'ServerHello' and 'ClientHello' messages. This way the client and server will have different handshake outcomes. The attacker can meanwhile derive connection keys and master key and further write application data, enacting the role of the server. The challenge here would be to determine the discrete logs in real time and delay the handshake completion until then. After precomputation, the descent stage takes about 70 seconds. The attacker can use command line clients that have a long timeout or no-timeout at all. To deal with some of the TLS web browsers that have shorter timeouts, TLS warning messages can be sent to reset timeout timer.

Scaling(section 4.1 and table-2, reference 1): The work shown on scaling the number field sieve to 768 and 1024 bits show that the 768-bit takes up to 2 days for the core-time for descent and 1024-bit takes up to 30 days. These durations open up feasibility of NSA being able to break the 1024-bit Diffie-Hellman very valid.

Solutions (section 5, reference 1): Recommendations given were as follows. Transition to elliptic curves because of the strength in removing advantage of precomputation. This would be the most feasible long-term option. Other options would include increasing minimum key strength to 2048 bits as precomputation becomes 10^9 times harder. Another solution would be to continue using the 1024-bits group, but to generate fresh groups and checking the server parameters.

2 - Second Vulnerability (Reference 2)

This section discusses two vulnerabilities presented in research paper titled “A Systematic Analysis of the Juniper Dual EC Incident”. These vulnerabilities were discovered in ScreenOS in juniper networks’ NetScreen VPN routers. First vulnerability is the administrative bypass to gain the access to the source code in the router which enabled the attacker to carry out the attack based on second vulnerability. We will discuss **the second vulnerability** in detail in the subsequent sections.

Overview (section 2, reference 2): NetScreen devices use Dual EC PRNG to generate pseudo random numbers that are used for Diffie Hellman(DH) Key Exchange and nonces. Dual EC was found to have a property which can be exploited by attacker to predict the subsequent states of the PRNG. Dual EC uses three public parameters: Elliptic curve and two distinct points P and Q to output a random number of 32 bytes. The state of the PRNG is represented by a 256-bit (32 bytes) value. Random number r_1 is the function of current state s_1 and P, $r_1= x(s_1Q)$. Next state(s_2) is a function of current state(s_1) and Q, $s_2= x(s_1P)$. Random number output is a function of current state value and P. If we can find an integer ‘e’ such that, $e = \log_P Q$ such that, $P = dQ$, attacker can recover state s_2. Then attacker can recover by reverse engineering the value of next state s_2 as follows $x(ds_1Q) \Rightarrow x(s_1P)$. This is possible because $r_1 = x(s_1P)$. Using the administrative access vulnerability present in the NetScreen devices, attacker could modify the value of curve parameters by fixing Q and making $P = dQ$ for some integer d.

Vulnerability Exposed (section 3, 4 reference 2): Juniper stated that even though Dual EC has this vulnerability, it doesn’t affect the pseudorandom numbers generated by ScreenOS as DUAL EC output is only used as seed to another PRNG FIPS X9.31. But a closer look at the decompiled Source code of the random generator function in ScreenOS revealed that the ‘if’ condition control variable for X9.31 was never set to true and the X9.31’s random number buffer was being used by Dual EC PRNG. In essence, X9.31 was never set to run.

How the vulnerability can be exploited (section 5, reference 2): Now let’s see how predicting the pseudorandom numbers can help in revealing the secrets keys and nonces. ScreenOS uses IKE (Internet Key Exchange Protocol) to facilitate VPN. IKE has two phases 1&2. Both phases make use of Diffie Hellman key exchange and nonces. Phase 1 generates keys for Phase 2. Phase 2 generates keys that are used other protocols like IPSec. Phase 1 generates a DH key as follows g^x(g is the field generator and x is generated by PRNG) and then a nonce (from PRNG). Using the nonce, attacker recovers state of the PRNG. If key was generated before nonce, attacker can generate keys for future key exchanges, otherwise attacker can generate key for the current exchange. By revealing the secret shared keys from Diffie Hellman, attacker can decrypt the VPN traffic passively.

Attack implementation (section 6, reference 2): Authors recreated this type of attack on a NetScreen device (SSG 500M) and supplied their own value Q and P such that dQ = P to the ScreenOS. Then they ran the VPN in several configurations: IKEv1 with PSK (Preshared key for authentication in Phase 1), IKEv1 with public key certificates for authentication in Phase 1 and IKEv2 with PSK and the attack succeeded in all but the first configuration. It didn't work on 1st configuration because the keying material used to generate shared keys also takes PSK as input which requires an offline attack to find the PSK. In IKEv2, PSK is no longer input for the keying material.

Key takeaways (section 8, reference 2): Authors could answer the reasons NetScreen devices were susceptible to the VPN decryption attacks as: Implementation of DUAL EC PRNG, Not cascading DUAL EC PRNG to X9.31 PRNG properly. Always reseeding the DUAL EC PRNG and use of 32 byte nonces If the nonces were only 20 bytes as in ScreenOS version before 6.2, attacker would have had to work on $2^{96}(2^{(32-20)*8})$ combination of values to find the correct value of x-coordinate on the elliptic curve.

Solution/Fix (section 9, reference 2): After the paper was published, Juniper reverted the ScreenOS parameters to the versions before 6.2 and removed DUAL EC and only used X9.31 as PRNG. It also reduced the nonce value to 20 bytes as it was before.

Conclusion: This raises the question of how secure the PRNG used in the network devices. Also, device manufactures should be able to demonstrate how secure the predefined parameters used by the PRNGs. Also, PRNG should strictly pass the randomness tests before they are implemented in the Devices.

References

[1] Adrian, D., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., Zimmermann, P., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J., Heninger, N., Springall, D. and Thomé, E. (2015). *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice*. [online] Association for Computing Machinery. Available at: https://dl.acm.org/citation.cfm?id=2813707 [Accessed 2 May 2018].

[2] Checkoway, S., Shacham, H., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R. and Rescorla, E. (2016). *A Systematic Analysis of the Juniper Dual EC Incident*. [online] Association for Computing Machinery. Available at: https://dl.acm.org/citation.cfm?id=2978395 [Accessed 2 May 2018].