Blood in the Water

Are there Honeymoon Effects Outside Software?

Sandy Clark, Matt Blaze, and Jonathan Smith

University of Pennsylvania

1 Honeymoons

In a previous paper at this workshop (and in a forthcoming full paper), we observed that software systems enjoy a security "honeymoon period" in the early stages of their life-cycles. Attackers take considerably longer to make their first discoveries of exploitable flaws in software systems than they do to discover flaws as systems mature. This is true even though the first flaws, presumably, actually represent the easiest bugs to find and even though the more mature systems tend to be more intrinsically robust.

The software honeymoon effect is surprisingly pronounced and pervasive, occurring in virtually every kind of widely used software system, whether open or closed source and whether an operating system, word processor, graphical rendering system, or web browser. While the length of the honeymoon varies, far more often than not, the time between the discovery of the first zero day attack and the second will be considerably shorter than between the initial release and the first.

In a forthcoming paper, we will examine various factors that appear to influence the honeymoon, but the central observation is this: honeymoons occur because, at the early stages of a software system's life, the attacker's (lack of) familiarity with the system matters far more than the system's intrinsic security properties. As the first flaws are discovered, the community of attackers develops more expertise and becomes more efficient at discovering flaws, even after the "low hanging fruit" bugs are patched and eliminated (when, we would otherwise expect, flaws should become harder to find).

This leads us to wonder whether there are security honeymoons in other aspects of system security besides software itself. In particular, are there honeymoon effects in basic security protocols? Cryptographic algorithms? Security architectures? A cursory initial analysis suggests that the answer may be an emphatic "yes".

In the rest of this position paper, we examine representative examples in security protocols (Needham-Schroeder), crypto algorithms (hash functions), and security architecture (virtual machines), where an analysis of inter-arrival times of published papers discussing attacks suggests that honeymoons are enjoyed across a wide range of computer security defenses.

2 Protocols

On the surface at least, security and cryptographic protocols would seem to have very different properties from software. Security protocols, while complex

B. Christianson and J. Malcolm (Eds.): Security Protocols 2010, LNCS 7061, pp. 12–17, 2014.

to analyze, have far few steps than software systems have lines of code, they are almost always open source (or at least those discussed in the research community are), and the adversary is historically other members of the research community who have a strong incentive to publish their attacks.

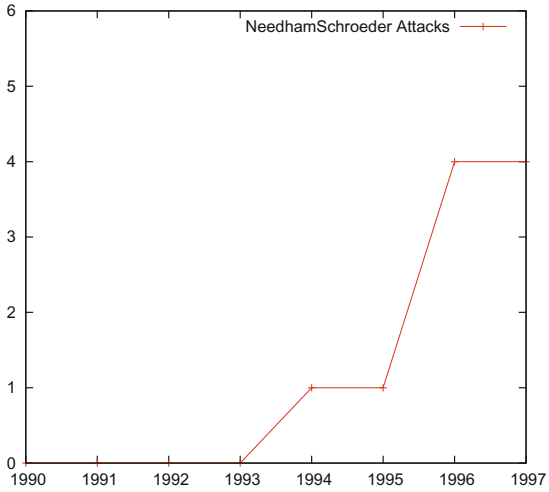

Fig. 1. Numbers of attacks on the Needham-Schroeder family of protocols by year

Consider the rate of vulnerability discovery in the original Needham Schroeder public-key protocol (and its patched successors). Its life-cycle appears to follow almost the same honeymoon curve as we found in a software systems. The protocol enjoyed a long honeymoon period, followed by a trickle of attacks and then a deluge of attacks against it. See Figure 1. For example, in 1994 Paul Syverson [6] outlined a taxonomy that replay and man-in-the-middle attacks would follow and indeed, the next year, Lowe published an attack on the protocol that followed the taxonomy. This was followed the next year by four new attacks and then by four more the year after [1,4] (see figure 1).

One major difference between the this attack life-cycle and the "classic" software system life-cycle is that the "midlife" phase (the post-honeymoon phase) of the cycle here was much shorter, with the attack papers coming in a rapid burst. We suggest that this might be due to a fundamental characteristic of security protocols that isn't present in software systems: a security protocol is typically designed to perform only one main function (key exchange) and is not subject to "feature creep" (we do not, after all, release *Security Protocol 2.0 – Now with More GUI*). When a flaw in a security protocol is found, either a fix is proposed, or the protocol is considered hopelessly broken and abandoned (or the problem ignored). It isn't subject to the same endless patch-revision, feature-addition, re-attack cycle as typical commercial software.

3 Security Architectures and Crypto Algorithms

Basic cryptographic algorithms, too, appear to exhibit a honeymoon, in which a relatively long period may pass before any structural cryptanalytic weaknesses are noticed. But once a first "chink in the armor" is found, this is often followed by a flood of increasingly serious attacks in rapid succession, sometimes culminating in the complete downfall of the algorithm (or class of algorithms). Crypto algorithms typically fall somewhere between security protocols and software systems in terms of size and complexity, but they share several things in common with protocols: they are mostly openly available to the attackers, the attackers are typically researchers motivated to publish, and mostly the tools for analyzing algorithms are not well understood.

Consider, for example, the attacks against broad classes of cryptographic hash functions such as SHA and MD5. MD5 enjoyed a very long honeymoon, followed by a recent succession of increasingly worrisome attacks. SHA(0) was discovered to have weaknesses by the NSA after its public release (we don't know how long this took, since the exact provenance of SHA inside the NSA is still classified), but enjoyed at least a two year honeymoon in the public cryptographic community before the attack was replicated [5]. The honeymoon is clearly now over for this class of hash function, with papers describing new and improved attacks being published in virtually every recent cryptology conference [2,7,8]. Examples of these are shown in figure 2.

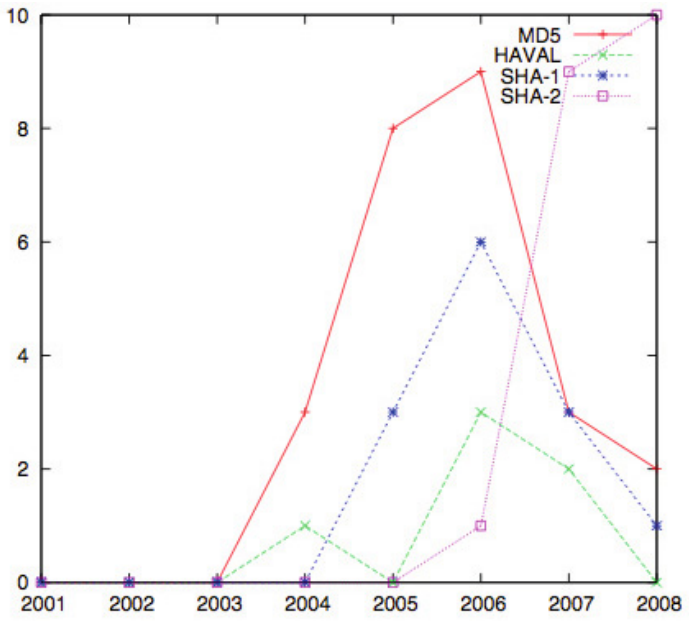

Fig. 2. Numbers of attacks on various hash functions by year

Honeymoons also appear to apply to security architectures as well as security protocols and algorithms. Virtual machines as an isolation mechanism appear to be a particularly salient example.

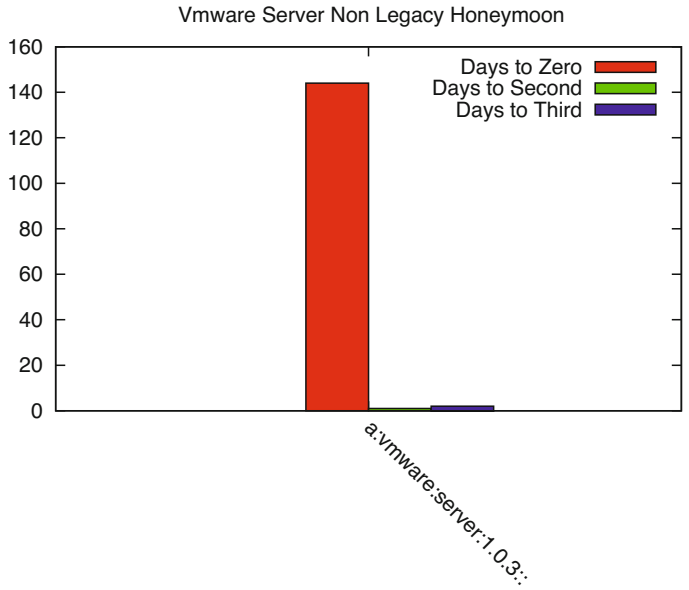

Fig. 3. Honeymoon (Days to first vulnerability/Days to Second Vulnerability for VMware Server

The concept of virtual machines has been in commercial use since the IBM 360 model 67. Since then it has been proposed as a panacea for a wide range of computer architecture issues including security. The idea of isolating systems in their own virtual worlds protected from outside interference or kept from interfering outside themselves is an important one, but even VMs must interact with their host systems in some way. For instance, virtual machines are software systems and therefore have the same security life-cycle properties as any other software systems. This includes the honeymoon effect. Figures 3 and 4 show the days to first vulnerability against the days to next vulnerabilities for one popular virtualisation product's server and workstation editions. Both graphs show a significantly longer period from initial release until the discovery of the first vulnerability than from the first vulnerability until the discovery of the second. As you can see from figure 4, the workstation edition also follows the exploit, patch (issue new version), re-exploit cycle common in most software products.

An additional security consideration for virtual machines is that the host/guest interaction itself is vulnerable. At Black-Hat Las Vegas 2009 Kostya Kortchinsky demonstrated that he could leap from the virtual machine to the host machine

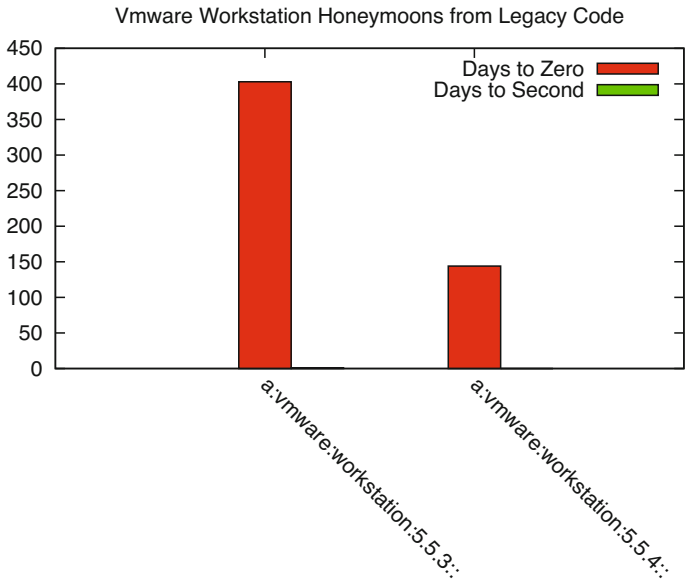

Fig. 4. Honeymoon for VMware Workstation: note that these represent honeymoons broken as a result of legacy code

and vice versa exploiting a memory leak in a shared frame-buffer [3]. In other words, for virtual machines, the honeymoon is clearly over.

4 Panic or Resignation?

Should we take comfort in this, or be frightened? Perhaps the answer is yes to both.

References

1. August, G.L.: An attack on the Needham-Schroeder public-key authentication protocol. Information Processing Letters 56, 131–133 (1995)
2. Chabaud, F., Joux, A.: Differential collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
3. Kortchinsky, K.: Cloudburst: A VMware guest to host escape story. In: BlackHat (2009), `http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchin%sky-Cloudburst-PAPER.pdf`
4. Lowe, G.: Some new attacks upon security protocols. In: Proceedings of the 9th IEEE Computer Security Foundations Workshop, pp. 162–169. IEEE Computer Society Press (1996)
5. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)

6. Syverson, P.: A taxonomy of replay attacks. In: Proceedings of the 7th IEEE Computer Security Foundations Workshop, pp. 187–191. IEEE Computer Society Press (1994)
7. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
8. Wang, X., Yu, H.: How to break md5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)