Security and Privacy on the Web
OVER THE PAST few decades, software developed for the Web has become more intricate, requiring knowledge of and the use of multiple moving parts—that is, dynamic software development and deployment stacks.

At the same time, Web software is becoming more integral to the human experience. For most of us, our online personas and real identities are tightly integrated. Online systems augment and connect with physical systems to transform and accelerate prior analog-only mechanisms.

Mission-critical assets of corporations, infrastructure providers, governments, and average citizens rely more than ever on Web software. So, they’re more exposed to attack through software vulnerabilities and flaws. This makes the creation of secure, privacy-preserving Web software essential to our future.

Web software, its supporting stack, and the Web software-engineering process must all have security and privacy mitigations integrated into them.
Challenges
The Edward Snowden revelations have shown that Web software insecurity is more widespread than most researchers previously thought possible. From weakening seed generators in encryption standards to eavesdropping on all forms of Internet traffic, a wide spectrum of problems needs fixing.

The Stack
Interactions online start with exchanges with a Domain Name System (DNS) server. This critical piece of network infrastructure is the directory service that translates a webpage address into its machine-readable address; for example, www.google.com becomes 74.125.137.104.

An attacker might inject malicious DNS data into recursive DNS servers, take over DNS servers and redirect traffic, tamper with the DNS registration, and so on. Software developers working at this level must be aware of not only DNS security standards but also safe coding practices that reduce the likelihood of a successful DNS attack.

Once the DNS service facilitates a connection between the Web software and the software on the requested page’s webserver, various attacks can occur through the communication channel. Examples include Web spoofing, denial-of-service attacks, and man-in-the-middle attacks. Developers must be aware of the security and privacy controls necessary to protect user data from these attacks.

Developers leverage many programming frameworks to (agilely) produce software. The frameworks are a mish-mash of a variety of tools that eliminate configuration and deployment pain. They also make many assumptions about the security and privacy risks their users are willing to tolerate. Software developers should be cognizant of these assumptions and the frameworks’ mechanisms for improving security and privacy protections.

The focused villain will concentrate on the weakest link in the security chain, which is typically the human. This means finding creative ways to compromise a computer system by coercing the user or developer to produce information that can lead to circumvention of the software’s defenses. Developers usually are unaware of the risks and have very limited ability to combat such attacks. A powerful prophylactic is to factor human behavior into the security and privacy mechanisms embedded in software.

The Development Process
The development of secure, privacy-preserving software requires

- the specification and communication of functional and non-functional security and privacy requirements,
- the use of security- and privacy-preserving tools, and
- the application of security- and privacy-preserving coding best practices.

Unfortunately, too many firms focus on at most two of these techniques. This leads to software that appears safe—that is, appears to be secure, privacy-preserving code. However, the resulting code offers little real protection. This state of affairs heightens the need for security and privacy intercepts.

Each development stage requires security and privacy assessments. These critical checkpoints should be enabling, transformative, and user-focused to reach the right equilibrium between product value, security and privacy, and usability.

The Software
At the bare minimum, Web software developers must ensure that their code is sufficiently hardened to protect against URL interpretation.
attacks, input validation attacks, SQL injection attacks, impersonation attacks, basic inference attacks, buffer overflow attacks, and inadvertent data disclosure attacks.

Web software provides entry points into the rest of the underlying system on which the code is deployed. Any manipulation of these entry points to access or manipulate subsystems (or information) that aren’t intended to be exposed poses significant risk. Whatever the development language used, Web software should have measures to reduce the probability of a successful attack when it pertains to input points and behavioral or interactional analysis.

This special issue presents a glimpse into some of these problem spaces and shows prototype solutions. In “Application Screen Masking: A Hybrid Approach,” Abigail Goldsteen and her colleagues highlight how technology can be used to protect users’ privacy while they consume media. In “An Empirical Evaluation of Web-Based Fingerprinting,” Amin Faiz Khademi and his colleagues describe a mechanism to defend against attackers trying to uniquely identify Web software users and tracking their browsing history and interactions. Finally, in “PriView: Personalized Media Consumption Meets Privacy against Inference Attacks,” Sandhya Bhamidipati and his colleagues detail technology you can use to balance the need to share and privacy constraints.

Web software security and privacy will depend heavily on progress in three areas:

- the codification of principles from secure and privacy engineering into programming-language constructs and tools;
- the creation of programming languages that have security and privacy as foundational tenets; and
- the construction of mechanisms or tools that enable secure, privacy-preserving (business) operations.

If the programmer doesn’t know the secure-coding principles and is unaware of privacy-engineering methodology, the resulting software will likely contain too much risk.

You can have a secure design, but if no supporting language constructs exist, the systems will have unacceptable levels of security and privacy risk.

With Web software being integrated into ordinary household devices, making progress in these three areas becomes even more pressing and relevant.