WHITE PAPER:

Preparing our software for a more dangerous Internet

AUTHORS:

Ken Kantzer
PKC Security Founding Partner

DATE:

May 30, 2018

Applying agile principles to software security

PKC has begun a series of software initiatives with its clients to take on custom software projects in a way that is safe and secure, and which prepares our clients for an uncertain age of internet security.

Now, we've had a chance to sit back and rethink how custom software development is done in enterprises, and how security fits into the picture.

The result is a set of five principles for agile software development. While these principles are meant to apply more broadly to our clients work, it just so happens that each principle has a security angle as well. We'll take a look at those angles here.

Principle #1 · Choose custom software as a last resort

While there are many financial benefits of going with an existing platform (custom software requires a lot of resources, and most general problems already have good solutions), there's also a security benefit in not going custom. As the internet gets more dangerous, it's becoming increasingly difficult for custom software shops that lack security expertise to keep up with all the new countermeasures and counter-countermeasures. Security must-haves like solid SSL encryption and 2-factor authentication are hard to implement on a software team with limited expertise in security or DevOps. That's why we first have our clients look for secure, vetted 3rd party products before deciding to go the custom route: we know the amount of effort required to secure a system and keep it secure.

Principle #2 · Practice agile development

Agility and resilience are key qualities in any secure system. Once you decide to go the custom route, you *have* to use this agility to your advantage: faster release cycles let you to quickly adapt to new threats, and tight-knit teams that communicate with each other often are more likely to catch bugs before they enter the wild.

Principle #3 · Follow the 3 F's, Fanatical Focus on user Feedback

If our products aren't used, then it doesn't matter how secure they are: their *net* security improvement is low. We *must* focus on user feedback because it makes better products, which means our security features get used more often. LetsEncrypt is an amazing example of the security benefits that can be reaped when a product gets magnificent adoption.

Principle #4 · Fail fast

Only the best ideas should be built custom, which means the not-so-great ideas have to fail quickly. "Fail fast" also happens to have a security-related corollary, "fail safe." Here's a helpful question that highlights the benefits of a fail safe system: would you rather have a broken traffic-light default to a green or a red light? Systems fail all the time, and that goes for security measures too. When a security measure fails, you want it to fail fast, and in a default-secure way.

Principle #5 · Hire the best people, and let them use the best tools

You want the best people, with the most up-to-date security knowledge, and the best tools, building software. You want people who are detail-oriented, and careful with how they design systems. These are developers who will independently bring up security issues to you *before they get exploited*. These are the developers who will take the time to add validation logic and get code reviews. That's what it takes these days to build secure custom software.

What we're doing to be more secure

Specifically, there's also a set of technical measures that we're baking into the security-sensitive custom software that we produce. Hopefully we can add this this list over time!

Secure Browsing

As surveillance equipment grows more prevalent, an encrypted TLS Internet connection is often the sole bastion of the average non-techie user against both oppressive regimes and opportunistic hackers. Our software projects use the latest browser security headers and modern ciphers for TLS to ensure that intercepted communications can't be spied on. This may seem trivial ("Just add HTTPS, and you're good!") but there are some hidden nuances here! Try copy/pasting your website URL here: https://securityheaders.com/ and see how you do.

Code reviews

We make sure that every code merge gets reviewed for security issues.

Credential management

We practice good secrets management. This means keeping credentials out of repositories, using password managers to share and store credentials, and making

sure that production credentials are not needlessly sitting on development or staging servers.

Secure authentication

Our enterprise software projects heavily rely on integration with SSO solutions like Microsoft's Azure Active Directory. This allows these projects to inherit good security measures, like Microsoft's two-factor authentication (if your organization has it turned on), and ensures that the same protections that are used for corporate email automatically get applied to our project.

Censorship-resistant technology

We are currently experimenting with next-generation VPN technology that will thwart even advanced, machine-learning dependent systems like the Great Firewall. These technologies are built straight into our projects, when needed, and won't require additional software installations or configuration.

Modern, secure-by-default web frameworks

All our projects use React.js, which has default-on security features that prevent client-side cross-site scripting (XSS) attacks, which are the most common type of website exploit seen in browsers.

What's coming next?

The above is just the beginning. We're also working on a few things that will prepare our clients even better for the next decade of internet security.

Bug bounties

As a security company, we need to take our own medicine—we do code audits for our clients, but to audit ourselves, we're looking into having bug bounty programs for our software. This allows us to see the most up-to-date vulnerabilities as they surface, and fix them before they are exploited.

Even better TLS encryption

As countries like Kazakhstan are imposing stringent requirements designed to undermine internet encryption[1], we are working on ensuring that all data connections in our projects are encrypted in transit with the very latest TLS security

[1] https://bits.blogs.nytimes.com/2015/12/03/kazakhstan-moves-to-tighten-control-of-internet-traffic/

features. This means taking advantage of cutting-edge features like Expect-CT and HSTS headers.

Management-less servers

We are looking into hosting all our projects on serverless infrastructure like Heroku, AWS ELB, Google App Engine, or Azure App Services. There are two security benefits to this:

1. It removes the need to remotely administer these servers via SSH or RDP - an attack vector that was exploited most recently to compromise Deloitte's networks.[2]

2. Management-less servers are patched automatically by the hosting provider, which means that there is less chance that a machine goes unpatched for long periods of time. In fact, these platforms often get fixed *before* the official patch gets released. This allows us to rule out a whole class of "unpatched systems exploits" attack vectors.

Have any questions or comments? Have any concerns? Feel free to reach out to us!

[2] https://krebsonsecurity.com/2017/09/source-deloitte-breach-affected-all-company-email-admin-accounts/