A brief introduction to how and why it will go wrong

Security should always be present in your mind because it will be on someone else's. Even if you feel your site or service would not be a target for intentional hacking: do not relax. Many of the high stakes attacks on bank accounts, for instance, frequently start with hacks on low risk, low security sites. MyFavKitty.com may not handle your finances but many people use the same email address and a similar enough password to do their banking so hacking one site can mean walking into the next.

We have a sea of internet traffic, and there be pirates looking for your ports – port and url scanning

Tools like (ze)nmap give you an immediate look at a specific host or network and frameworks like metasploit use this to then suggest possible attack vectors after determining open ports and generally which software is providing a service and what version that software is. This approach can be completely arbitrary (ie. a port scan across the internet picking up your open box) or highly targeted (ie. once you have penetrated a network you can craft specific scans of servers to build both a host-to-service list and network topology). The main attack surface of web apps are of course the url responses and there are many dictionary url scanners (a good one is skipfish) that attempt to find responses from unlisted addresses in order to both gather more info and see if you have hidden useful pages.

http://map.ipviking.com/

http://cseweb.ucsd.edu/~clbailey/PortScans.pdf

Know your friends, close thine commoners and the revolt your enemy – fuzz testing

There are a bunch of fuzz testing tools, some open source, many proprietary but the premise is the same. Approach the valid endpoints of an application with arbitrary and often erroneous data and see what happens. Normally, not a lot, an error message (which can be useful too) or a blank response, but make enough of them in various forms and exceptions to the rule can be an opening. Lets say one of your inbound endpoints accepts POSTed xml, parses it, stores it then passes it to a worker to be processed. This may come from a trusted source and be filtered at the network level but what happens if that source is compromised. How does your application handle a million randomly crafted xml files? What responses does it give to malformed xml? Even GET requests with bonkers parameters can sometimes cause your app to behave unpredictably with the right data. Until recently rails did not handle certain things well; running fuzz tests against endpoints that at some point created a symbol from a parameter would bring down the server as symbols weren't garbage collected until Ruby 2.2 (25 Dec 2014).

-
- https://code.google.com/p/zaproxy/wiki/HelpStartConceptsFuzz
- https://www.owasp.org/index.php/Fuzzing_with_WebScarab
- http://pages.cs.wisc.edu/~bart/fuzz/

Dropping the ball – SQL injection

Hopefully this vulnerability is something that we are all aware of and one that doesn't need to be covered in detail... Until the next time you hand write a piece of SQL, which at some point somebody will. ORMs and the like have made us fear this problem much less than in the past. PHP and Perls popularity in early dynamic web development and their associated propensity for string interpolation lead to hilariously simple attacks. Typing in an apostrophe, a semi-colon and then some easily guessed SQL could give you entire customer databases. So remember as soon as you are hand writing SQL sanitize EVERYTHING, even if you can't immediately see how an attacker could fill that field with some Bobby Tables fun.

-
- http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/

The Good, The Bad and The Very Very Cross – XSS attacks

Cross site scripting is the cause irritation, fraud, fake likes and another damn good reason to remember to sanitise any data from a user, including urls. There are several types of attack that come under this heading, non-persistent being the simplest: Having a parameter of a url rendered straight to the next page (eg. puts "Hello #{params[:name]}") without sanitisation allows someone else to create a url that can execute their javascript on your page in the clickers context. Many web frameworks automatically sanitise this input and provide meta tokens to mitigate various types of attack, but they are not infallible and you need to remember that this can happen. The second main category is when your inputs aren't fully sanitised before storage either. eg. Leaving a comment on a blog post that contains javascript that's rendered to other users; can be used for all sorts of tomfoolery. The final main type of XSS is DOM based, using a similar vector to modify the responses of javascript to trigger events in the browsers rendered page. Many modern attacks based on this use a melange of techniques and play around with things like double encoding and the like to bypass attempts to suppress them.

- http://html5sec.org/
- https://www.exploit-db.com/papers/13102/
- http://www.slideshare.net/x00mario/the-innerhtml-apocalypse
- https://www.owasp.org/index.php/Types_of_Cross-Site_Scripting

Crazy horses and Warewolves – trojans, malware, hijackware, ransomware

Crafted install packages can infect any machine regardless of OS or setup. If you install code as you or a priviledged account that code can execute anything you could. You should only install from verified sources, and even then you need to regularly check that there are not broadcast security vulnerabilities in the verified code. Security auditing your open source libraries sounds immensely tedious but it is usually easily automated as many companies have to do so for compliance purposes.

- https://github.com/rubysec/bundler-audit
- http://www.offensive-security.com/metasploit-unleashed/Binary_Linux_Trojan

Your only as <strong> as your weakest url_for – social engineering and weak link devices

The only attack vectors that have a roughly 90% success rate are social engineering and commodity network hardware. One of my favourite security speakers came to a conference he wasn't invited to and gave a lightning talk on how to gain access to a conference using social engineering, showing his press pass and the 4 emails it had taken to get a free ticket to a $3000 conference. Pretty much every mainstream wireless router manufacturer has had some form of embarrassing vulnerability exposed over the last decade and that does not seem to be improving. And with regards to bad user practice combined with poor network hardware security leading to quite chilling results:

- http://www.networkworld.com/article/2844283/microsoft-subnet/peeping-into-73-000-unsecured-security-cameras-thanks-to-default-passwords.html
-
- http://www.theregister.co.uk/2015/03/05/broadband_routers_sohopeless_and_vendors_dont_care/
- http://www.techrepublic.com/blog/it-security/securitys-weakest-link-technology-no-match-for-social-engineering/

Tools

Tools

- https://code.google.com/p/zaproxy/wiki/Docker
- https://www.owasp.org/index.php/Appendix_A:_Testing_Tools
- http://portswigger.net/burp/intruder.html
- https://nmap.org/zenmap/
- http://metasploit.com
- https://code.google.com/p/skipfish/