1. What is the paper about? What is/are the vulnerability? What causes the vulnerability?

The paper discusses a new defense strategy against cross-site scripting (XSS) attacks called Blueprint. According to the paper, the number one security threats for the Internet at large are XSS attacks. The attacks rely upon unauthorized script code injected into a web page to extract confidential and sensitive user data. The malicious script code is embedded in untrusted HTML and is executed on the victim’s web browser within the context of the vessel website. Enforcing a no-script policy for untrusted HMTL to prevent unauthorized script execution is the aim of defenses against XSS attacks. The strategy of the defense outlined in this paper and the goal for Blueprint is to minimize the burden on the browser in determining untrusted content.

1. What is/are the contributions of the paper? How was the vulnerability or insecurity discovered?

The paper discusses two main defense approaches: content filtering and browser collaboration. In content filtering application attempt to detect and remove all scripts from untrusted HTML while browser collaboration gets information from the browser as to which scripts are authorized.

Content filtering, also widely known as sanitization, uses specially designed filter functions on user input to protect against/remove any potentially malicious data or instruction. Simple sanitization, such as disallowing HTML syntax control characters, because every control character that can be used in a malicious attack code/script also has a legitimate use in many legitimate non-script contexts. Advanced content filters look at untrusted content and try to predict how the client web browser’s parser will interpret it. This not only makes the web application’s job of protection depend in part on the different web browser parsers, but also leaves the content filters open to exploit via any anomalous behavior by any of the browser parsers.

Browser collaboration involved tools to help browsers distinguishing authorized scripts from unauthorized scripts. Protocols are used to communicate a set of authorized/unauthorized scripts, which the browsers use to enforce policies denying execution of the unauthorized scripts. This unfortunately requires web application to use custom browsers to be able to communicate with the server containing the set of scripts; therefore, it lends itself more to a long-term solution, where all browsers already contain this communication by default, rather than a near-term one. In addition, the communication standards used by all browsers would have to be agreed upon, which in itself is a long arduous process.

1. The detailed techniques to solve the problem.

Blueprint claims to satisfy the three main objectives of preventing XSS attacks: 1) it is robust, protecting even with browser quirks, 2) it supports structure, benign HTML derived from untrusted user input, and 3) it is compatible with existing browsers current in use. Since the parsing behavior of browser can be unreliable, Blueprint effectively takes over and controls the parsing decisions instead of the browser. The Blueprint application uses information about the flow of untrusted HTML in a browser to create a “blueprint” or structural representation of untrusted web content, with XSS attacks removed. The generated tree is fed to the browser’s document generator bypassing the browser parser and avoiding its analogous and/or unreliable behavior. The implementation does not require knowledge of how the browser parses the data and instead enforces the application’s understanding of the web content on the browser. In doing this, the effect of the browser parser’s analogous behavior does not come into play.

1. What are the strength/weaknesses of the paper?

The first weakness I could see is for any application not written in PHP, an alternate version of Blueprint needs to be run, which has a separate process to communicate with the web application over TCP. The program already consists of a server-side component as well as a client side script library; therefore, adding the separate process makes things even more complicated and increases the possible points of failure. A second weakness that I saw is they ran their tests on popular web application using the most popular web browsers. While that will give the public at large an idea how well it works overall, many company’s application will be unique in the types of data and content that they deal with and may in some cases be using old/outdated browsers. The type of attacks that would work in those cases may never be exercised in the testing on with the most popular applications/browsers. A third weakness is their reported overheads. They report a range of overhead for memory consumption to be from 0% to 13.6% with the average being 5%. While this may seem “modest” to them a 13.6% overhead could impact overall performance. They also measured the overhead for page size and for WordPress it was a staggering 52.4%. This does not seem like a result that would be acceptable to any company even if it does offer the protection they advertise. Again for WordPress the average increase in processing time was a high 55%. This again would never be acceptable to companies that want their information to the user as quickly as possible.

The first strength of the program is that tests proved it is resilient against even subtle XSS attacks. The platforms used in testing embedded attack strings in a variety of different contexts for a template web page. The program was able to defend against all 94 different XSS attacks tested and there were no successful attacks due to exploits of browser parsing quirks. Since it was tested on eight of the most popular web browsers, which make up over 96% of the browser market, this goes a long way to show how effective it is against XSS attacks for the vast majority of users. Another strength is Blueprint preserves the rendering order of the page content by interpreting the models synchronously as the page is rendered. The content model is accepted as input to a model interpreter. The model interpreter constructs the parse tree and when complete, everything including the invocation script is removed. This means all the processing is done without leaving a trace that it was even performed.

1. What can you do better?

The main this that I think needs to be address to make this product better are the resource utilization. Overheads as high as 52.4% for page size, 13.6% for memory consumption overhead, and increase in processing times as much as 55% should be reduced. While the paper claims the overall latency is imperceptible to a user, there is a caveat that they read the web page in a continuous manner from top to bottom. They also suggest mitigations such as serving fewer comments per page. These “suggestions” seem to be hacks, rather than real solutions to problems incurred by using the Blueprint product. It is not a very good business model to ask the company using your product to change how they render web pages to overcome the shortcomings in your product. Address those resource utilization issues and I think the overall product would be greatly improved.