METHODOLOGIES & TOOLS FOR WEB APPLICATION SECURITY ASSESSMENT

ORY SEGAL, DIRECTOR OF SECURITY RESEARCH

A whitepaper from Watchfire

TABLE OF CONTENTS

PREFACE

Web application security assessments are a crucial phase in the development lifecycle of any web application. The process of assessing a web application should be handled using the same approach as any other testing (e.g., Unit testing, Quality Assurance, etc.). A well-documented methodology should be followed carefully, and in most cases, the use of automated tools will speed up the process.

This whitepaper suggests a methodology for web application security assessments, as well as an explanation on how to use automated tools for accelerating the assessment process.

ASSESSMENT METHODOLOGY

CUSTOMER INTERVIEW

Often, the job of assessing the security of a web application is outsourced. In such cases, it is imperative that all information related to the web application be collected in an orderly fashion, summarized and given to the person doing the assessment. It is recommended that the person performing the assessment interviews the customer, be it the person who ordered the assessment, the application developer(s) or the application administrator.

The interview should aim to retrieve any possible piece of information which may be relevant to the assessment process, as well as the following crucial data:

- Assessment date, time and location
- Website URL
- Application path limitations:
  - What parts of the application should be assessed? What parts of the application should be avoided?
- Assessment goals and expectations:
  - Is this a full-blown audit? What are the goals of the audit?
  - Is this just a proof of concept? What are the goals?
- Application technical details:
  - Web server type and version
  - Application server type and version
  - Database server type and version
  - Additional technologies (ASP, JSP, etc.)
  - Is the application protected by an IDS, Proxy or Firewall?
- Test account information: in some cases, it may be required to obtain more than one test account. This may become helpful when testing a multi-user system, and also in cases where the test account may be locked by the application during the assessment.

Note: In cases where the assessment is outsourced, legal documents should be signed before the assessment begins.

PRELIMINARY RESEARCH

After collecting all possible information about the web application, it is advised to scout the internet for information about possible vulnerabilities in third-party products that are being used. Several internet databases exist for this task:

- CERT: http://www.cert.org
- National Vulnerability Database: http://nvd.nist.gov/
- CVE (Common Vulnerabilities and Exposures): http://cve.mitre.org/

Downloading the products' official documentation may also be useful in order to conduct thorough research about other insecurities in the products, such as default accounts, etc.

At this point, you can use an internet search engine (e.g., Google), to scout more information about the web application (assuming that it is facing the internet). A search engine may help you to find simple vulnerabilities such as:

- Possible SQL Injection – search for SQL error messages
- Directory listing – search for directory listing strings (e.g., "index of," "parent directory," etc.)
- Credit Card numbers – search for credit card number formats

You can find more details on how to use search engines for penetration testing in the following presentation: http://www.blackhat.com/presentations/bh-europe-05/BH_EU_05-Long.pdf ("*Google Hacking for Penetration Testers*" by Johnny Long, from BlackHat Briefings, Europe 2005)

FINAL PREPARATIONS

- At this point, the person performing the assessment should validate that the audit will be performed at the date and time that was set between the penetration tester and the application owner. This is especially important for live/production applications, which are serving real-world users.

- It also may be useful – and will save time later on – to prepare a skeleton document which will be completed with information and findings during the assessment.

- Finally, it is important to prepare the tools that will be used during the assessment (this will be discussed in the next section of the whitepaper).

DURING THE ASSESSMENT

- Notify the site and application owners that the audit is about to begin

- Get to know the application – browse the application with a browser, and understand the structure, the scripts and their role in the application's logic and flow.

- Extract information about technologies found (programming languages, platforms, etc.)

- Map and understand the site's structure (what should be added to the skeleton report)

- Configure the web application security scanner, and run it
- Where needed, use manual auditing techniques (external utilities). This will be discussed in the next section of the whitepaper.
- Interpret the scanner's results and add them to your manual findings
- Go over the weak spots that were found by the automated scanner and continue to research them manually, using external tools, until the goal is achieved

WHEN THE ASSESSMENT IS DONE

- Create the final report. It is crucial that you go over the findings from the automated scanner and make sure they contain enough information so that the developer or administrator will be able to reproduce them.
- Take screenshots where possible – they will help the person reading the report to figure out what happened during the assessment.
- Notify the website and application owners that the audit is done

ASSESSMENT TOOLS

WEB BROWSER

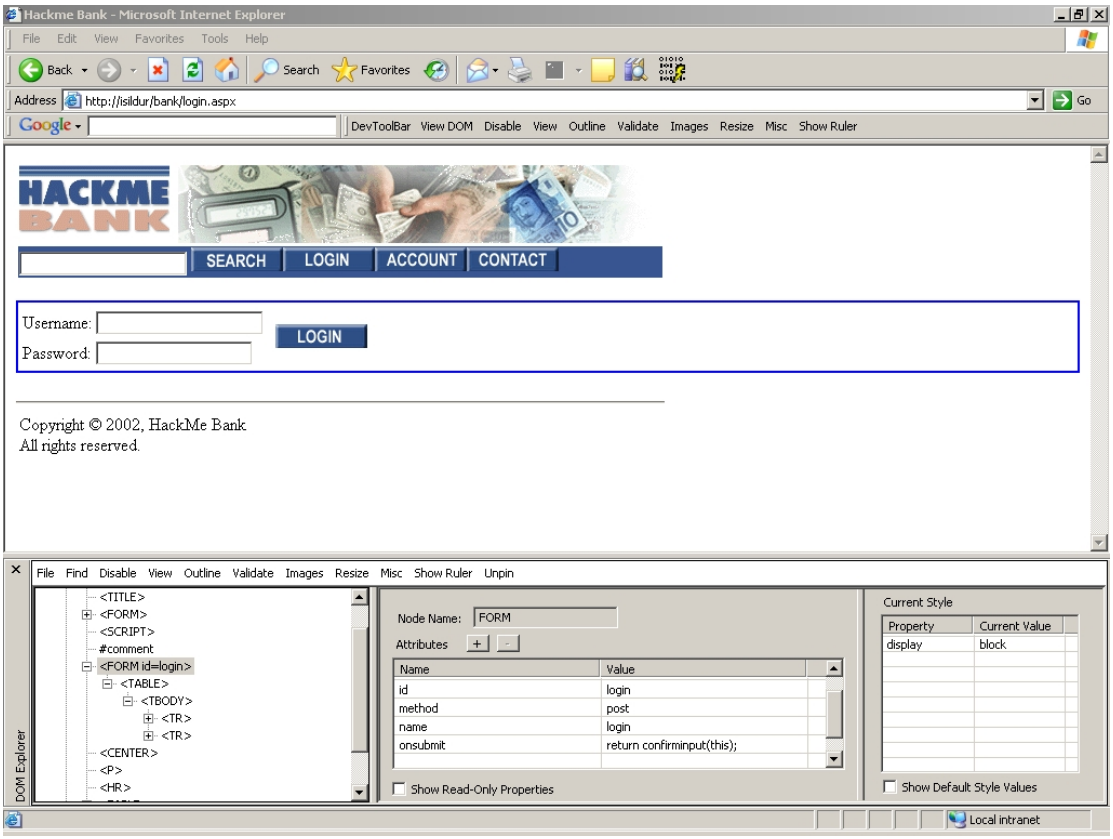
Since the main interaction between the user and the web application is done through a browser, it is crucial that the person performing the assessment browse the site using a regular web browser as well.

In simple scenarios, some of the tests can be performed inside the browser, by tampering with URL parameters, or sending tests to HTML forms using POST requests.

If the web application uses client-side validation scripts (e.g., JavaScripts), you may need to use an HTTP proxy (described later in this whitepaper) that will intercept outgoing requests and allow you to manipulate raw data, such as HTTP headers or parameters sent in the body of the request (POST requests).

Some add-ons can be used to improve the functionality of the web browser during a web application assessment, such as a DOM inspector. The DOM inspector allows you to view, traverse and dynamically update that HTML DOM directly in the browser window.

In the following screenshot, you can see the DOM inspector in action:



WEB APPLICATION SECURITY SCANNER

The web application security scanner is used to speed up the process of locating web application vulnerabilities.

Today, web applications are becoming extremely complex, and contain hundreds if not thousands of scripts, where each script accepts several parameters. If a person was to assess each parameter in each script, it would take weeks or months. An automatic scanner, on the other hand, can traverse sites much faster than a human can, and finishes covering vast areas in minutes. This is also true for the process of locating vulnerabilities such as Cross-Site Scripting, SQL Injections, Buffer overflows, etc.

Most automatic scanners operate by traversing the site (spidering), analyzing the site structure, the scripts, forms and parameters, and then sending tests to the application. The tests are then validated for success, according to predefined rules, which are based on the response of the application to the test.

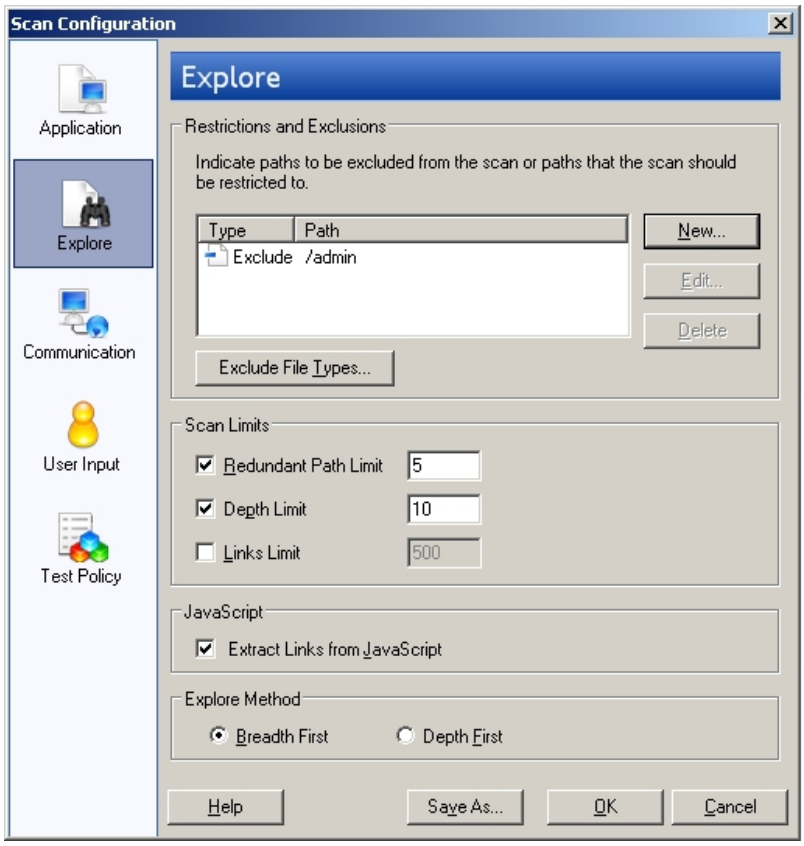
At the end of the scan process, a report is presented to the user.

Automatic scanner assessment best practices

- In order to find all existing security problems in a web application, it is crucial that the scanner traverse the whole site. This process often requires proper configuration that will fit the specific web application that is being tested. Here are some important things to watch, while configuring the automatic crawler (spider):

- **Starting point URL:** the main page of the application
- **Application path limits:** what areas of the site should be included or excluded
- **Crawling depth:** defined either by the amount of link-clicking from the starting point, or by directory depth
- **Redundant path limit:** since many forms and pages can link to the same resource (e.g., script), scanners usually have a limit on the amount of times that a specific script is visited. Some applications use a single script to provide all of the functionalities. In such cases, make sure that you either remove this limit or modify it to a larger value as required.
- **Amount of links to crawl:** define how many links the crawler should visit
- **Crawling algorithm**: can be set to either BFS (breadth first search) or DFS (depth first search). BFS is the algorithm used by most crawling engines, but in some cases, DFS may be required (e.g., applications that require that the person follows a specific flow of actions).
- **Automatic form filler configuration:** most professional automatic scanners include automatic form-filling capabilities which send real values to form fields just like a person would do. The form filling is required in order to properly traverse the site, and discover all possible areas. The automatic form filler often requires that you configure certain values for specific parameters (email addresses, usernames, password, etc.). Failing to configure the automatic form filler properly may result in areas of the site that are not scanned.
- **Exclude file types:** some applications contain large files that should not be downloaded during the crawling (e.g., MS Word documents, PDF, MP3, etc.).
- **Communication timeout:** some websites may respond slower than others. In such cases, it is crucial that you configure an adequate timeout so that the crawling process will retrieve all the pages and will not timeout on some of them.
- **Crawler proxy settings:** if you are accessing the web application through a proxy, it is required that you configure proxy settings. This may include proxy authentication credentials.
- **Authentication credentials:** in cases where HTTP authentication is required (e.g., Basic Authentication, NTLM, etc.), make sure that you configure the credentials before starting the crawling process.
- **Client-side certificates:** some sites require that the client install a client-side certificate in order to verify the user's identity.

Here is a screenshot of some of the crawling configuration items in AppScan:



- **Session Management:** most applications use a token in order to maintain user sessions. These tokens are usually either a cookie or a parameter. Once the user logs in, the application registers the token and follows it in order to keep the session state. When using an automatic scanner, it is crucial that the scanner detect the session tokens and refresh them when needed. In order to detect the session token, most scanners have a session management configuration which allows users to:

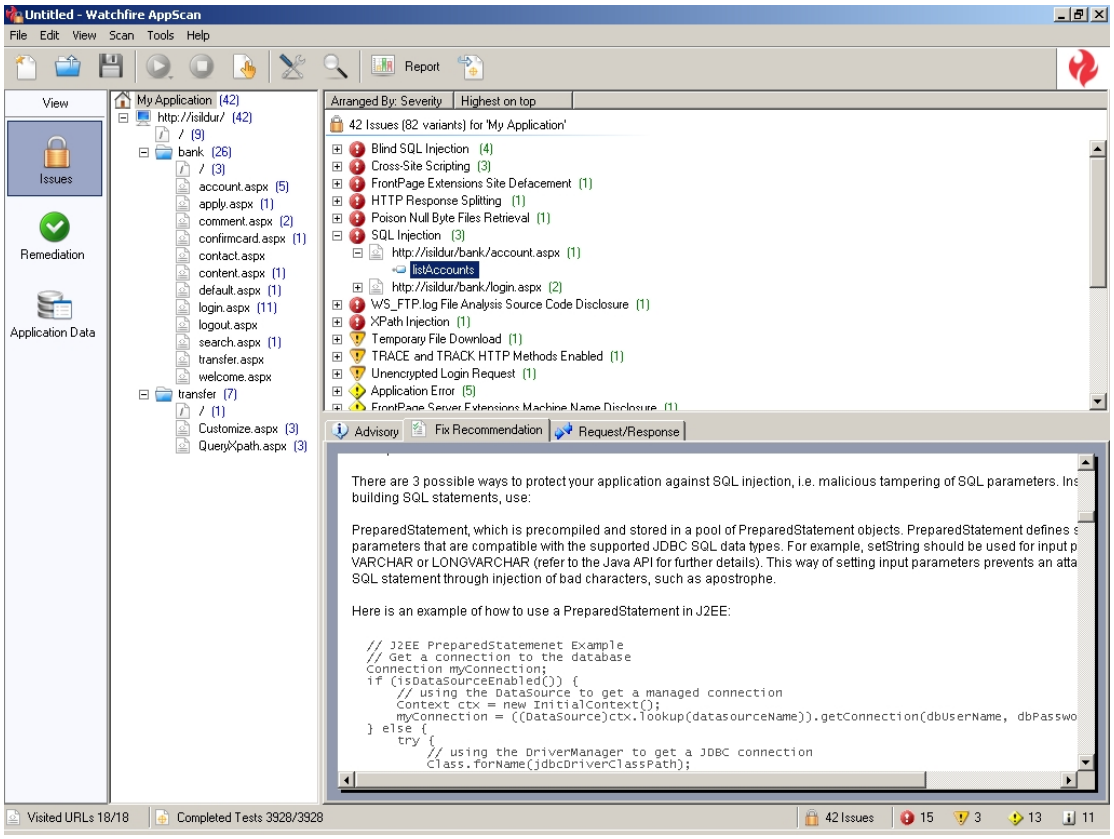
    - Define which parameters and/or cookies are session tokens

    - Record a login sequence which will later be replayed by the scanner in order to acquire new session tokens and refresh the test requests with proper tokens.

      Once the Session Management configuration is done, the scanner will use this information before sending tests to the site in order to refresh the tokens and make sure that the requests sent are actually considered "in session" by the web application. Without this ability, the scanner will be useless against web applications that track user sessions.

    - Since session tokens are sensitive, it is crucial that the web application scanner not tamper with their values during the testing of other parts of the application. In case the scanner contains tests against session tokens, these should be sent separately from the other tests.

- Once the crawling process is complete, it is critical to ensure that the whole site was traversed by the crawler. If not, make sure to check the configuration items as mentioned above. Not scanning the whole site means that you will likely miss security problems.



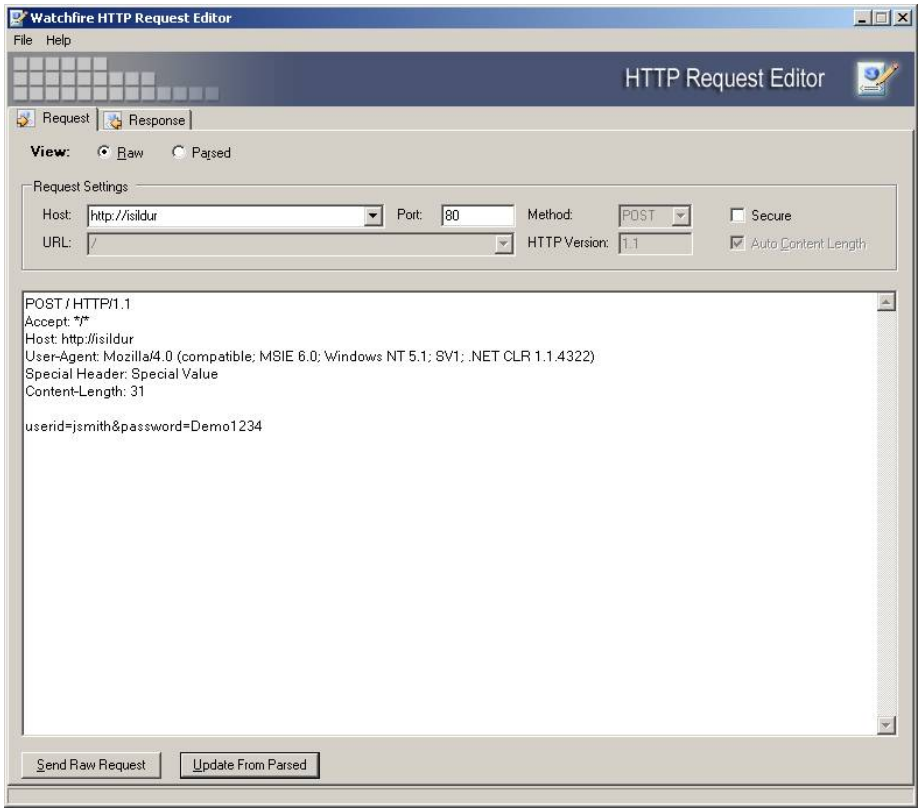
HTTP EDITOR

When performing a web application security assessment, it is sometimes required to perform manipulations on raw HTTP requests. Since a browser will not allow such manipulations, an HTTP request editor should be used.

Here are some sample uses for an HTTP editor:

- Sending HTTP requests without the restrictions of the browser's security model
- Crafting special HTTP requests which cannot be done in a browser (manipulations to HTTP headers such as the Referer header)
- Craft requests that require SSL encryption
- Create manual manipulations to test requests created by an automatic web application security scanner

Most HTTP editors allow the user to manipulate a request either in a raw mode (textual editing of the request) or through some sort of a form (parsed view).



Most HTTP Request editors contain the following useful features which may help to speed up the process of sending a manipulated HTTP Request:

- Raw editing of HTTP requests
- Parsed editing of HTTP requests
- Support SSL communications
- Calculate the "Content-Length" header automatically
- Change every part of the HTTP request such as: headers (e.g., cookies), method, version, port, parameters, URL, etc.

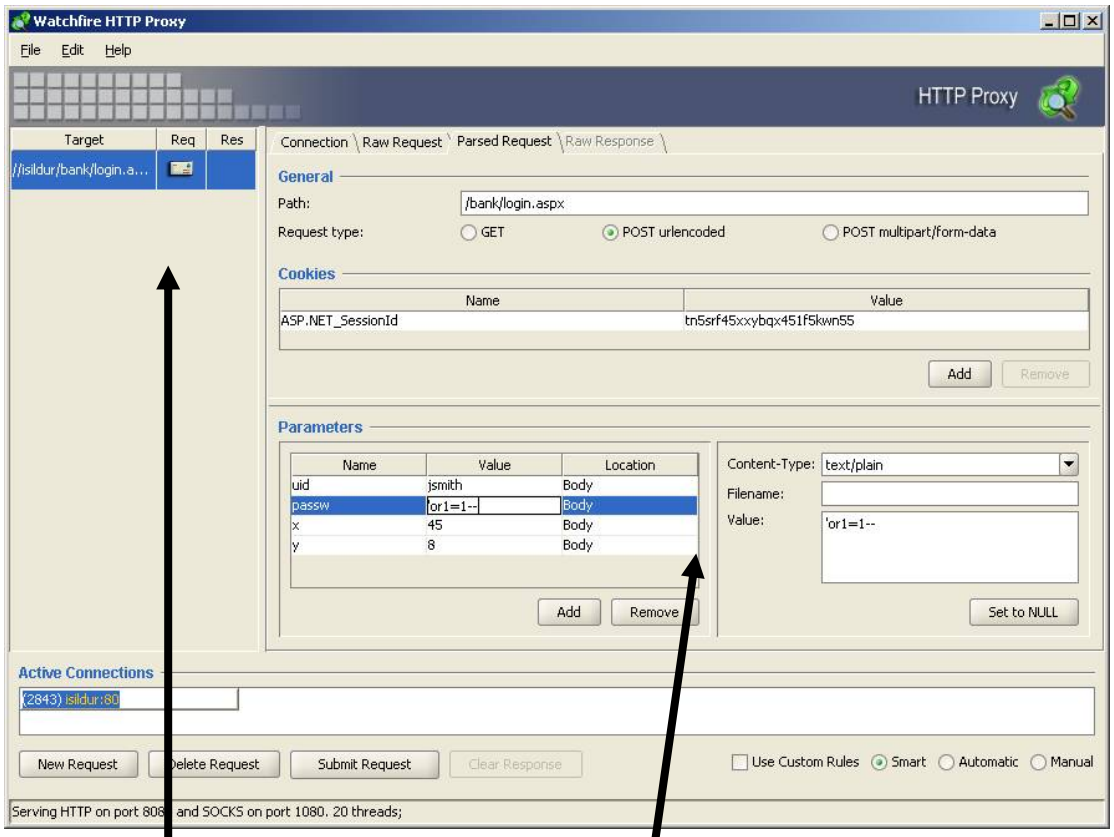
The results of the HTTP request can usually be viewed as raw HTML or through an embedded browser.

HTTP PROXY

Oftentimes, it is required to intercept HTTP requests and responses before they reach their target (either the web server or the browser). An HTTP proxy will perform such interceptions automatically, and will allow the person using it to manipulate the request or response as needed. Here are some scenarios in which an HTTP proxy may be useful:

- Changing HTTP requests and responses on the fly, when they leave the browser or before they arrive to the browser. The proxy usually allows performing manipulations on the raw request/response or through a parsed view (just like the HTTP editor). This is very useful to:
  - Overcome HTML and browser limitations (field lengths, path length, etc.)
  - Circumvent client-side scripts which validate user input (e.g., JavaScripts)

- Log HTTP requests/responses for further investigations
- Create automatic rules that will manipulate certain HTTP requests or responses automatically each time a condition is met (e.g., change a cookie value to a different one each time the request leaves the browser)



CONCLUSION

Performing a thorough web application security assessment is a complex task which should be approached like any other software analysis – with a methodology, testing procedures, a set of helpful tools, skills and knowledge.

Today's web applications span over thousands of web pages, and accept a vast amount of input from users, in many different locations. This requires the person assessing the application to go over *each* script and *each* parameter, and to test it for numerous possible security flaws. This tedious job of assessing each input field can be streamlined by using automated tools such as a web application security scanner, an HTTP proxy and an HTTP editor.

REFERENCES

Relevant Documents:

[1] Web Application Threat Classification (WASC project): http://www.webappsec.org/projects/threat/

[2] Web Security Glossary (WASC project): http://www.webappsec.org/projects/glossary/

[3] Watchfire Security Zone portal: http://www.watchfire.com/securityzone/default.aspx

[4] Web Application News (CGISecurity.net): http://www.cgisecurity.net/

Tools and helper applications:

[1] Watchfire PowerTools (Contains an HTTP Proxy, String Encoder/Decoder, Web Server Connection Tester, HTTP Request Editor and a Regular Expression Tester):
http://www.watchfire.com/securityzone/product/powertools.aspx

[2] Watchfire® AppScan® (Web Application Security Scanner):
http://www.watchfire.com/securityzone/product/appscansix.aspx

[3] Microsoft Fiddler (HTTP debugging proxy): https://www.fiddlertool.com/fiddler/

[4] Microsoft Developer Toolbar (contains DOM Inspector, among other helpful browser utilities):
http://www.microsoft.com/downloads/details.aspx?FamilyID=E59C3964-672D-4511-BB3E-2D5E1DB91038&displaylang=en

ABOUT WATCHFIRE

Watchfire provides software and services to manage online risk. More than 500 enterprise organizations and government agencies, including AXA Financial, SunTrust, Nationwide Building Society, Boots PLC, Veterans Affairs and Dell, rely on Watchfire to monitor, manage, improve and secure all aspects of the online business including security, privacy, quality, accessibility, corporate standards and regulatory compliance. Watchfire's alliance and technology partners include IBM Global Services, PricewaterhouseCoopers, TRUSTe, Microsoft, Interwoven, EMC Documentum and Mercury. Watchfire is headquartered in Waltham, MA. For more information, please visit www.watchfire.com.

