Position Paper

Web 2.0 Security and Privacy

December 08

enisa
European Network and Information Security Agency

www.enisa.europa.eu

About ENISA

The European Network and Information Security Agency (ENISA) is an EU agency created to advance the functioning of the internal market. ENISA is a centre of expertise for the European Member States and European institutions in network and information security, giving advice and recommendations and acting as a switchboard of information for good practices. Moreover, the agency facilitates contacts between the European institutions, the Member States and private business and industry actors.

Legal notice

Contributors

This paper was produced by an ENISA editor using input and comments from a group selected for their expertise in the area, including industry, academic and government experts. The content was collected via wiki, mailing list and telephone conferences. This paper should not be taken as representing the views of any company or other organisation.

List of contributors:

- Suresh N Chari, IBM, USA
- Andy Cirillo, Depaul University, USA
- Simon Grehan, National Centre for Technology in Education, Ireland
- Michael Hart, Stony Brook University, USA
- Rob Johnson, Stony Brook University, USA
- Ajit Jaokar, Futuretext, UK
- Jesús Jiménez Cordente, ISDEFE, S.A, Spain
- Robert Pajak, Interia, Poland
- Corin Pitcher, Depaul University, USA
- Robert Rachwald, Fortify Software, USA
- Zulfikar Ramzan, Symantec
- Thomas Roessler, W3C
- Paul Thompson, Dartmouth University, UK
- Daniele Vitali, Reply, Italy
- Andreas Wiegenstein, VirtualForge, Germany

Editor: Giles Hogben, ENISA (European Network and Information Security Agency)

Contact details:

For general enquiries about this Position Paper, please use the following details:
Email: positionpapers@enisa.europa.eu
Internet: http://www.enisa.europa.eu/

Executive Summary

Web 2.0 – user-generated content, rich user interfaces and co-operative, dynamic services – has also brought with it a new and extremely virulent breed of 'Malware 2.0'. A key motivation for this study is the link between Web 2.0 and the increase in 'drive-by' malware infections requiring no intervention or awareness on the part of the user. To give some idea of the threat posed, a Scansafe report analysing malware trends reports that risks from compromised websites increased 407% in the year to May 2008.

One of the most important sources of vulnerabilities in Web 2.0 is the inadequacy of access and authorisation frameworks used in Web 2.0 environments. In particular, this report highlights problems in policy frameworks governing the separation of control between web applications. These centre on the 'same-origin' policy, which sandboxes web applications coming from different domains, and the cases where this policy is either deliberately relaxed or circumvented for malicious purposes. Problems in access and authorisation frameworks often stem from the difficulty in finding a balance between allowing enough freedom for Web 2.0 applications to function and providing adequate security.

Web 2.0 has also brought a sea-change in the way knowledge and information is managed. One page contains content and even executable code from multiple sources including end-users, and information may be syndicated (eg, using RSS) and altered many times from its original source. This means in particular that:

- The increased opportunities for contributing content also provide more opportunities to inject malicious code leading to many vulnerabilities in the category of cross-site scripting, an important weakness exploited by Malware 2.0. This is exacerbated by very short development cycles and the fact that programmers often have little or no security training.

- Trust in information is more difficult to establish, making it easier to promote fraudulent information for criminal purposes (eg, distortion of stock prices in so-called 'pump and dump' schemes).

The vulnerabilities identified in this paper are extremely important because of the potential damage they cause through identity theft, extortion via botnets (we describe an attack where botnets are controlled via a Web 2.0 application), financial loss, loss of privacy and damage to reputation.

Technology can address many of the more immediate problems, but eliminating the more systemic risks requires a comprehensive approach to security involving people, process and technology. Some of the elements of such an approach include:

- Government policy – eg, secure development incentives such as lightweight certification schemes and the funding of pilot actions.

- Research – eg, usability of TLS/SSL, privacy-preserving means of establishing trust in information in Web 2.0 environments, and advanced Javascript security models.

- Awareness-raising campaigns – eg, lifetime of data on the web, use of stronger authentication in certain Web 2.0 scenarios, and the ineffectiveness of age-verification and content-rating schemes in Web 2.0.

- Standardisation – eg, further development of existing access-control and authorisation frameworks to provide improved security in access control and authorisation, standards for privacy-preserving establishment of information provenance and pedigree.

- Provider measures – eg, improvement of authentication measures and the use of encryption.

- Secure development initiatives – covering secure development processes for Web 2.0 and tools to facilitate those processes including built-in security features for IDEs and APIs, the implementation of anonymous strong authentication tools.

Summary of Risks

Access and Authorisation

Separation of control between applications: The same-origin policy, sandboxing applications from different domains, is the principal method of controlling access between web applications served by different providers. By their nature, however, Web 2.0 applications cut across domain boundaries. Web 2.0 has therefore produced many techniques for circumventing this policy, both for legitimate and illegitimate purposes (such as cross-site request forgery).

Excessive privileges: Many Web 2.0 services ask users to delegate access credentials to, for example, email accounts or bank accounts. Currently, users often have to give away the highest level of privilege, eg, unlimited, permanent access to all features of their email account rather than just time-limited access to their address book, to access a service. The lack of finer grained authorisation is a barrier to the use of such applications and a serious risk for those who do.

Distributed control: Web 2.0 applications such as mashups and feeds can be used as part of a distributed control system (eg, bot herding using blog posts, etc).

Insufficient sandboxing: Sandboxing features – especially those offered by the 'IFrame' HTML tag – are often insufficient and allow illegitimate cross-domain communication.

Weak authentication: Weak authentication makes it easy to steal Web 2.0 account details.

Information security risks for minors: Minors can be exposed to inappropriate content in Web 2.0 through the failure of age-verification techniques or content rating systems.

Server authentication usability problems: Users ignore SSL certificate errors or discrepancies between the server domain and the domain certified, enabling phishing and man-in-the-middle attacks.

Development process issues: Developer skills, the complexity of APIs, and typically short development cycles in Web 2.0 all increase the vulnerability of Web 2.0 applications. Cross-site scripting (XSS), one of the most well-known vulnerabilities, is caused by poorly validated user inputs and is an important example of a vulnerability whose root cause is poor development processes.

Inadequate specification of policies: Security and privacy guarantees are rarely specified in a precise way making it difficult to create systems with a specific level of assurance.

Method exposure in migration: Several toolkits automatically convert server-side to Web 2.0 Ajax applications. This can lead to the exposure of restricted features during migration.

Knowledge and Information Management

Fraudulent pedigree/provenance: Misinformation is easily propagated through syndicated news stories, blog posts, and social data, which provide few trust cues to users. This has very serious consequences such as stock price manipulation and the control of botnets via RSS feeds.

Vulnerabilities of collaborative knowledge systems: There is usually little or no means of establishing the trustworthiness, integrity or provenance of statements made in collaborative editing systems such as wiki articles. This can have dangerous consequences – eg, a survey showed that only 25% of people searching for health advice regularly check the source and date of the information they find to assess its quality.

Metadata attacks: Vulnerabilities in metadata, such as the ID3 media metadata format, are a key weakness of Web 2.0 applications.

Web 2.0-specific privacy threats: Theft or misuse of personal data was the top concern of end-users in a Web 2.0 context according to the ENISA survey (1). Privacy issues include indiscriminate image tagging, behavioural marketing and leakage of corporate secrets.

Terms of service problems: Web 2.0 service providers are often under pressure to intervene in inappropriate user-generated content. However in doing so, they give up the legal protections granted to 'mere conduits' (aka 'common carriers') which poses a difficult dilemma to many providers.

Low use of TLS/SSL: Many service providers transfer highly sensitive information, including passwords, without encrypting it or authenticating the service provider.

End-user related

Public-awareness problems: Examples include difficulties in data deletion, inappropriate trust in false or unsubstantiated information, and the meaning of SSL/TLS messages in browser UIs.

Lack of context in user interface: Users traditionally trust websites based on the name or URI. With many Web 2.0 applications, there is no comparable context to base trust on.

Automation attacks: The inability to detect whether a system is controlled by a human being or not is a serious vulnerability in Web 2.0 because many interfaces are publicly accessible, allowing over-exploitation of limited resources. An important weak point is found in CAPTCHA techniques.

General software and scripting vulnerabilities

Vulnerabilities of Web-enabled devices: Web 2.0 vulnerabilities can lead to attacks on network infrastructure, eg, pharming attacks using Web 2.0 applications.

Browser plug-in vulnerabilities: These are rapidly increasing and are often the weakest point in the browser security model.

JSON vulnerabilities: Several important vulnerabilities have been exposed which use JSON, a lightweight format for the exchange of the Javascript data commonly used in a Web 2.0 context.

Summary of Recommendations

Government policy recommendation

- Policy incentives for secure development practices such as certification-lite, reporting exemptions and the funding of pilot actions. These incentives are needed to address the large number of, eg, cross-site scripting vulnerabilities caused largely by poor development practice.
- Address/investigate Web 2.0 provider concerns about conflicts between demands for content intervention and pressure to maintain 'mere conduit' or 'common carrier' (US) status. This is considered a very important problem by Web 2.0 providers because of the strong user-generated content component.
- Encourage public and intergovernmental discussion on policy towards behavioural marketing (eg, by the Article 29 Working Party).

Research directions

We recommend that the following research topics be funded by industry and governments:

- Usability of provider authentication methods (eg, TLS/SSL) and encryption methods for Web 2.0 data;
- Improved usability of stronger user-authentication mechanisms (eg, one-time passwords);
- Piloting and assessment of the use of stronger authentication mechanisms in Web 2.0 scenarios;
- Defence against abuse of finite resources (eg, CAPTCHAs);
- Trust infrastructure using social networks (eg, key trust and reputation);
- Privacy-respecting means of establishing the provenance of information;
- Advanced Javascript security models;
- Licensing models and contracts for personal data; and
- Metrics and testing frameworks (code analysis, penetration testing) for security and privacy in Web 2.0 applications.

Awareness-raising

This section describes the issues we consider most important to be highlighted in Web 2.0 related awareness-raising campaigns. Examples include the lifetime of data on the web, how and why to use privacy features in Web 2.0 applications, the interpretation of TLS/SSL signs and errors in browsers, and (for developers) security threats and secure development practices.

Recommendations for standardisation work

- Emerging specifications, such as W3C's Access Control for Cross-Site Requests, OASIS XACML, oAuth, HTML 5, OpenID, Cardspace and Liberty should be further developed and promoted to provide, in particular, a comprehensive and standardised access and authorisation framework for Web 2.0 applications.
- Develop a consistent interpretation of security-related messages in browser environments.
- Develop standards for the privacy-respecting determination of information provenance and pedigree.

Provider issues

- Standardising the implementation of stronger authentication mechanisms in Web 2.0 environments.
- Use of authentication measures appropriate to data being accessed; eg, login and check account status with just a plain password but, for financial operations, re-authenticate using a one-time password token.
- Use of TLS/SSL based encryption wherever sensitive data are transmitted.

Developer issues/Browser vendors

- Secure development initiatives: secure development processes for Web 2.0, training and tools to promote such processes and security-by-design, and built-in security features in IDEs and APIs.
- Use of techniques and existing technologies for anonymous strong authentication.

Web 2.0 Security and Privacy

Audience

This paper is aimed at corporate and political decision-makers as well as Web 2.0 application-providers. It should provide an overview of the key threats and risks to the security of users of Web 2.0 and, most importantly, recommendations for actions and best-practices which mitigate those risks. It is also aimed at raising awareness of the legal and social implications of new developments in web-based threats.

End-user Survey

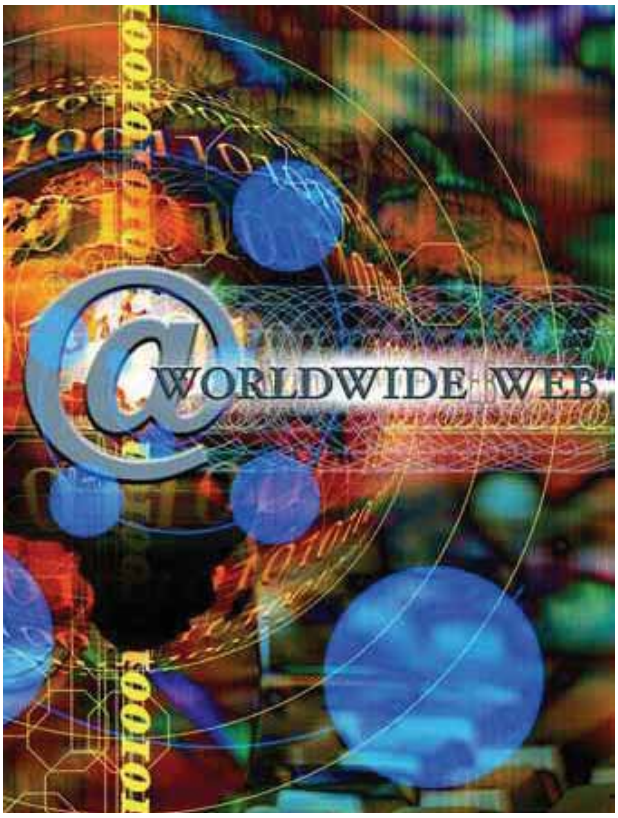
In conjunction with this study, we conducted a survey of 1,500 end-users of Web 2.0 sites and applications, which may be found at (1). We have cited parts of the result set throughout this paper where this supports our statements. We recommend, however, that the reader review the full results of the survey as an important addition to the findings presented here.

1 Scope

As a working definition and as a basis for analysing the security threats, we define Web 2.0 to include the following features:

- Rich browser-based applications including Asynchronous Javascript XML (AJAX) and flash applications.
- End-user-generated web content: content generated using a browser-based application rather than being uploaded directly to a web-server. Such content is often subject to radically different or less well-defined security and regulatory regimes than content generated and controlled directly by the service-provider. The user-generated content is also spread in multiple copies, referenced and historicised, making information removal almost impossible to obtain in reality.
- Client-side code, community-based widgets, user-defined code, community-based software, Ajax, IFrames, etc.
- Co-operative dynamic services deriving content and functionality from multiple sources, jurisdictions and legal entities. Examples are so-called mashups and dynamically composed web-services and content syndication, eg, OpenSocial, Google Mashups, etc.

Note that, throughout, we focus on risks specific to Web 2.0 rather than on general information-security risks (eg, identity theft, pharming, phishing, spam), unless there is a Web 2.0-specific variant.



2 Introduction

Web 2.0 – applications such as Flickr, YouTube, Digg, BlogSpot, Facebook and Wikipedia – has been the subject of much hype and controversy, but few people would disagree that there is a distinct and alarming new wave of 'Malware 2.0' . As this paper shows, Web 2.0 application vulnerabilities such as cross-site scripting have been an important contributor to the rise in Malware 2.0. To give some idea of the extent of Malware 2.0 attacks, consider the Scansafe Web STAT report analysing malware trends, which reported that between May 2007 and May 2008:

- The volume of threats confronting web surfers had increased 220%;
- Risk of exposure to exploits and compromised websites had increased 407%; and
- In May 2008, 68% of web-based malware exposure was via compromised websites (2).

Sophos has listed Mal/Iframe – the transmission of malware through IFrame vulnerabilities in web pages – among the top five malware vectors for every month in 2008 (3). Google researchers recently analysed the contents of several billion URLs in the Google cache. They discovered about 450,000 URLs were successfully launching so-called 'drive-by' downloads of malicious code (4).

Infection with malware is increasingly via such drive-by vectors where machines are infected simply by visiting a web page, without any further intervention by the user. Attackers are increasingly embedding attacks in bona fide web pages, often exploiting the vulnerabilities described in this paper. For example, in May 2008 hackers succeeded in embedding malicious scripts on Nature.com, which has an estimated 877,000 unique visitors per month. The size of the black market in drive-by infections demonstrates its power to infect users' machines with malware. Companies now offer fixed prices for a given number of web page infections of IFrame vulnerabilities (5).

Apart from providing opportunities for malware attacks, Web 2.0 has also brought a sea-change in the way knowledge and information is managed. In section [3], we describe the new paradigms in terms of architectural patterns. The most important paradigm shifts have been in collaborative content editing and syndication. In Web 2.0 scenarios, we commonly find that:

- One page contains content and even executable code from multiple sources.
- Information and executable code sources are likely to be private individuals. These individuals often cannot be traced and held accountable for wrong-doing.
- Trust in information is established through user-votes and reputation systems rather than brand names or PKI.
- Information may be syndicated (eg, using RSS) and altered many times from its original source, making the provenance and pedigree of information difficult to trace.

The most important security vulnerabilities arising from this paradigm shift are:

- Users can contribute content and even executable code into web resources. In combination with fast development cycles and the resulting poor validation of user input, this leads to many vulnerabilities (especially in the category of cross-site scripting – XSS [5.4.3]).
- Problems in separating control between Web 2.0 applications in a way which provides both adequate functionality and security. The 'same-origin' policy has traditionally protected web applications from accessing unauthorised resources but multiple applications or 'widgets' running within a single browser often need to communicate across domain boundaries and developers are therefore forced to circumvent access restrictions. On the other hand, the fact that there exist

[1] A term used by Kaspersky labs in http://www.viruslist.com/en/analysis?pubid=204791987

multiple loopholes in this policy, and that there is often no other basis for protecting sensitive resources, means that applications often have no means of protecting resources when they need to.

- The difficulty of tracing 'information pedigree' – the trustworthiness of information sources – means that it is much easier to promote fraudulent information for commercial (eg, so-called 'astroturfing'[2]) or criminal purposes (eg, distortion of stock prices in so-called 'pump and dump' schemes).
- Due to a change in business models, revenue streams are more often from 'behavioural' marketing based on personal information collected from private individuals participating in content creation.
- Data processing is increasingly between peers, rather than C2B, and is therefore not subject to the provisions of data protection legislation and far harder to regulate in any way (at least within Europe).
- Metadata, allowing processing of content, has a much more important role, meaning that attacks on metadata systems can have more impact through poisoning information sources and embedding malware in malformed syntax.
- Syndication mechanisms may be used as a means of extremely loosely coupled and distributed control for criminal purposes.

These new vulnerabilities are extremely important not only in terms of their increasing prevalence but also because of the potential damage they can cause and the inability to trace and detect their perpetrators. Some of the most important impacts are the installation of spyware for identity theft, the use of malware for extortion via botnets – we describe a Web 2.0 attack where botnets are installed and controlled via Web 2.0 applications [5.5.1.2] – financial loss, loss of privacy and damage to reputation.

We recommend initiatives in technology, research, development and standardisation to mitigate these risks, but we also emphasise that because no single technology or group of people is responsible for many of the vulnerabilities, they demand a comprehensive approach involving people, process and technology. Some of the elements of such an approach include:

- Government policy, such as secure development incentives including lightweight certification procedures and the funding of pilot actions;
- Research initiatives such as the usability of TLS/SSL, privacy-preserving means of establishing trust in information in Web 2.0 environments, and advanced Javascript security models;
- Awareness-raising campaigns on, for example, the lifetime of data on the web, the benefits of stronger authentication in certain Web 2.0 scenarios and the low success-rate of age-verification and content-rating schemes in Web 2.0 environments;
- Standardisation initiatives: for example, the further development of existing access-control and authorisation frameworks to provide improved security; we also recommend the development of standards for the privacy-preserving establishment of information provenance and pedigree;
- Provider measures, such as the improvement of authentication measures and use of encryption; and
- Secure development initiatives, covering secure development processes for Web 2.0 and tools to facilitate such processes including built-in security features for IDEs and APIs. We also recommend more widespread implementation of anonymous strong authentication tools.

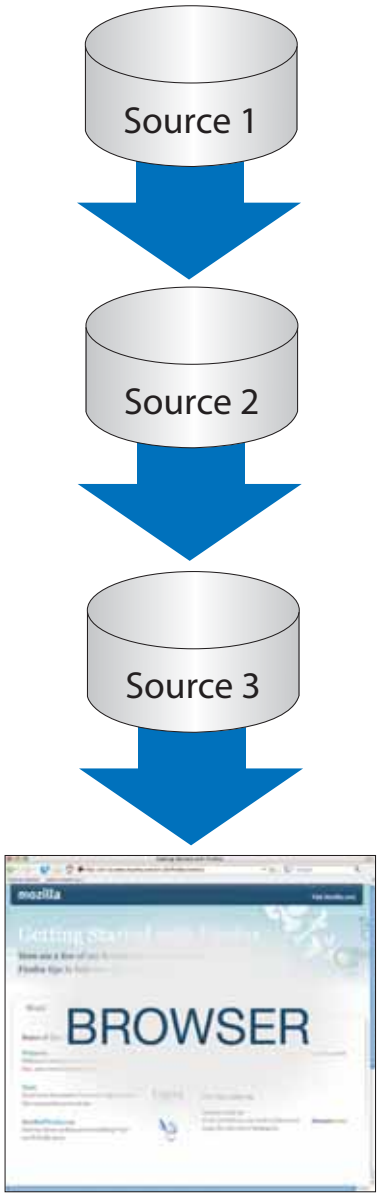
[2] http://en.wikipedia.org/wiki/Astroturfing

3 Web 2.0 Architectural Patterns

Web 2.0 is characterised by a set of new architectural patterns which underlie its operation. It is helpful in analysing its implications for security to review these patterns since many of the risks are introduced by changes in the architecture of the web.

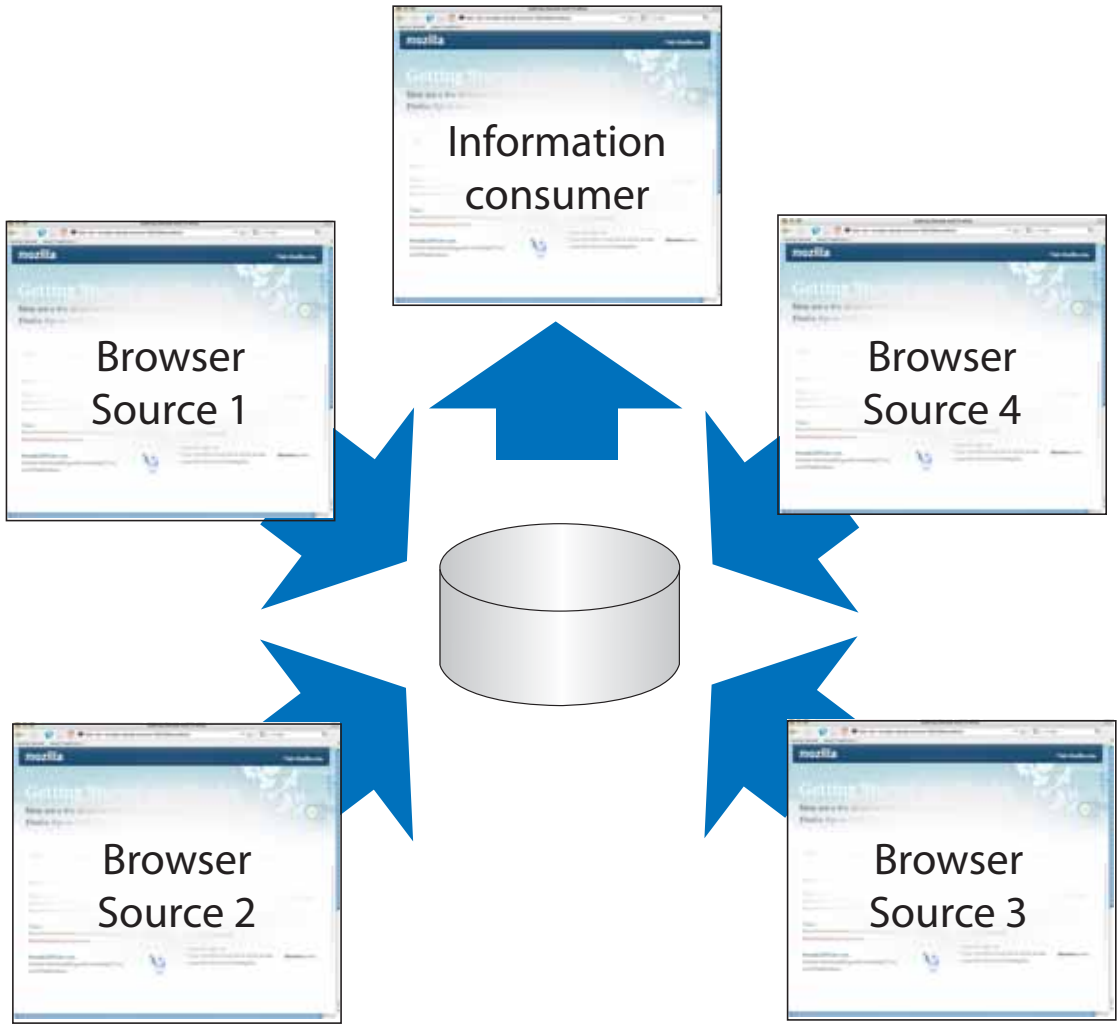
3.1.1 Information Syndication

Using syndication mechanisms such as RSS, information passes through multiple hosts, publishers and formats before arriving at the end-user. As we will discuss later in the paper, this makes the reliability of the information harder to establish and makes it easier to introduce fraudulent information into a system.



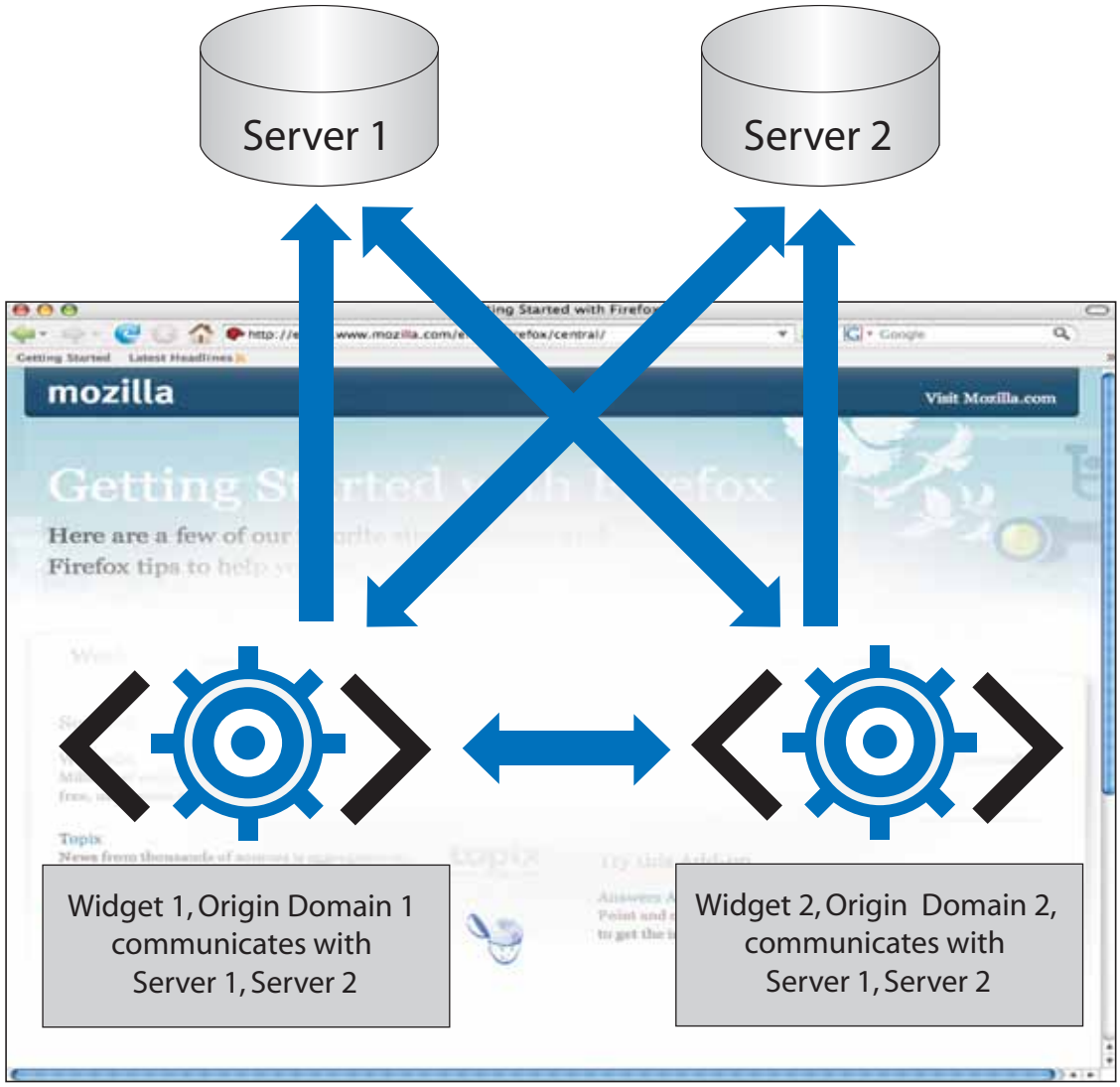
3.1.2 Collaborative Editing

Information which appears as a single source or article is edited by multiple (possibly unnamed and untraceable) users. The best-known example of this is Wikipedia.



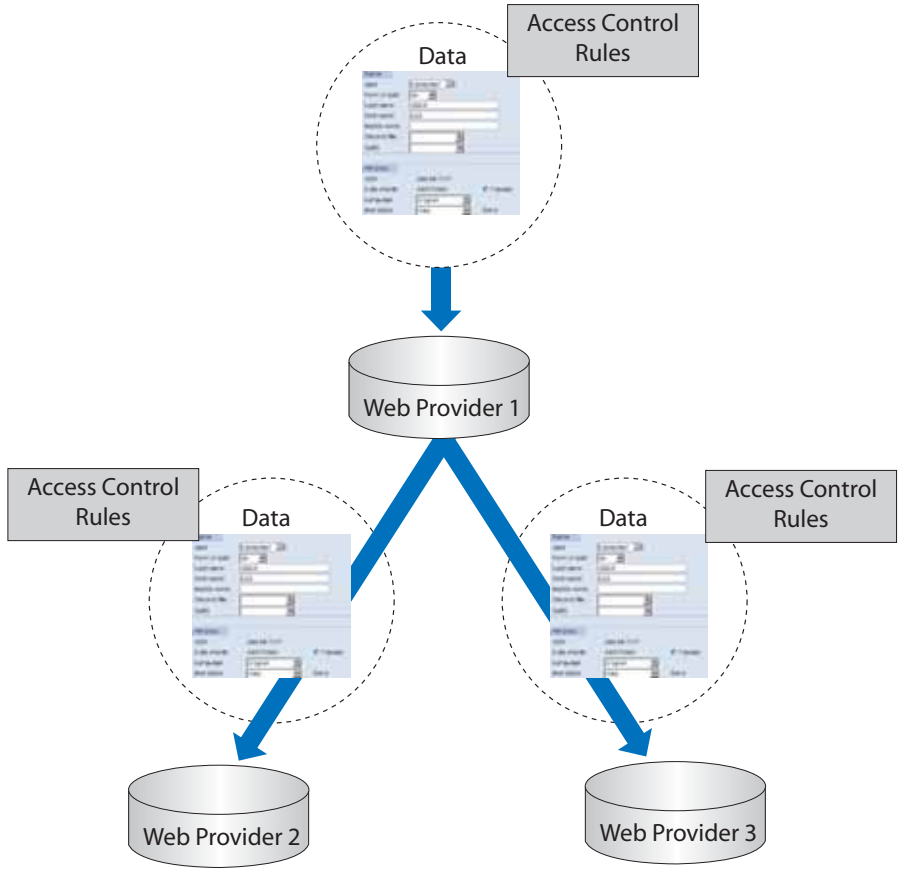
3.1.3 Embedded Widgets

A single web page contains Javascript or Ajax applications from multiple origins. Each of these may need to communicate with multiple servers or other widgets on the page in order to function. An example of this pattern is the widgets found in social networking applications such as Facebook and MySpace. Online advertising is another example since advertisements are usually served by a third party but need to take into account other content on the page in order to personalise the advertisements. Often widgets are developed with a limited life span (eg, complex Flash ads).



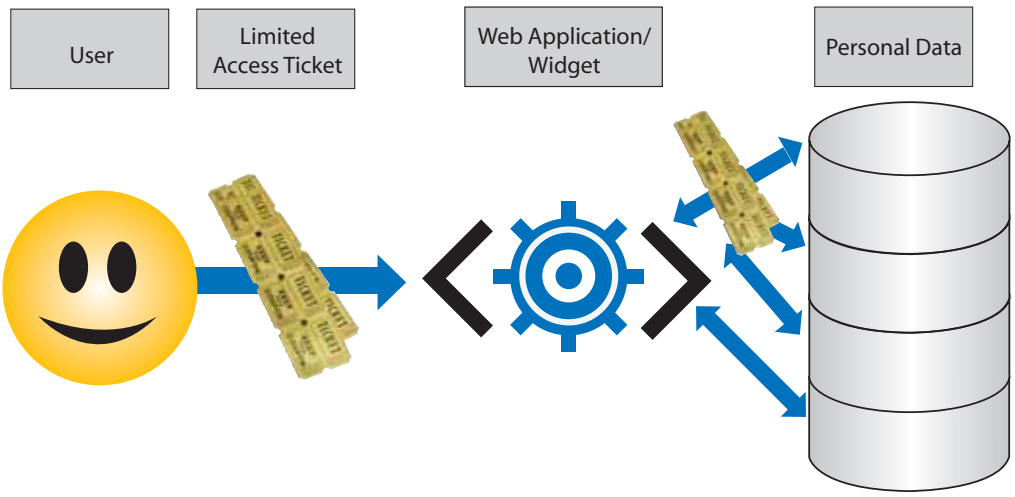
3.1.4 Portability of Access Rights

Any sensitive information which is syndicated through RSS or otherwise passes through multiple applications in a chain of Web 2.0 services may also be subject to access control and authorisation rules. A classic example of this is the export of social networking data where privacy rules and controls have been set.



3.1.5 Delegation of Authorisation Rights

Web 2.0 applications which aggregate other services often need authorisation to access sensitive information. A classic example is the delegation of authorisation to access an email address book to a social networking application. Another example is a service which aggregates different Internet banking services into one page.



4 Typical Web 2.0 Application Scenarios and their Security-relevant Features

4.1.1 Mashups

Mashups build new web services by combining data and components from existing web applications. When the underlying services require user authentication, a mashup user must grant the mashup access to his or her underlying account(s). For example, consider a mashup that reads the header information for all spam messages in a user's Yahoo Web Mail account and plots the geographic origin for each message using Google maps. To use this system, the user must currently give his or her Yahoo mail password to the mashup system (and hope that the mashup software behaves as advertised).

An early and well-known example of a mashup is Chicagocrime.org (6) which allows a user to search a database of crimes reported in Chicago and see where they took place, filtering results by location, type of crime, district, and date, and viewing details of the crime on the map. This is typically used by people considering buying a property in a specific area.

Note that online advertising networks may also be classed as mashups since they integrate data from many sources and deliver a service based on this data within a single page.

4.1.2 The Same Origin Policy – Example Scenario

An important feature of most Web 2.0 scenarios is the same-origin policy. This is a cornerstone of browser security. It prevents a document or script loaded from one origin from manipulating properties of or communicating with a document loaded from another site of origin. In this case the term *origin* refers to the *fully-qualified domain name, port* and *protocol* of the site hosting the document. The following table demonstrates how the same-origin policy would handle document manipulation by a script originating from http://www.example1.com/exampledirectory1/page.html.

Target URL	Outcome	Reason
http://www.example1.com/exampledirectory2/page.html	Success	
http://www.example1.com/exampledirectory1/exampledirectory2/page.html	Success	
https://www.example1.com/exampledirectory1/page.html	Failure	Different protocol
http://www.example1.com:81/exampledirectory1/page.html	Failure	Different port
http://host2.example1.com/exampledirectory1/page.html	Failure	Different host

The same-origin policy is one of the only features of a browser which provides separation of control between different application owners, ie, an application cannot take over control of another application for malicious purposes. For a classic example of a problem which could arise without this policy, consider a tabbed browsing environment where a user has an Internet banking application open in one tab and a page served by a hacker in another tab. If the malicious page were able to control content and actions on the Internet banking page, it would be able to make a transfer of funds into an arbitrary bank account (even if One-Time Passwords were used) by substituting a bank account number in a POST request.

On the other hand, there are important situations where the same-origin policy needs to be relaxed in order for web applications to function. POST requests are an important example of such necessary relaxation of the same-origin policy. In order to provide enough flexibility for applications processing HTML form data, there are no built-in controls in browsers forbidding the submission of cross-domain POST requests. On the other hand, this important design decision has led to vulnerabilities under the heading of cross-site request forgery [5.3.1.1.1.1].

4.1.3 Social Network Privacy Rules

Users often need to be able to set preferences such as who can tag photos with their profile, which of their friends can and cannot access profile information (which is now possible on Facebook, for example). Users need to be able to export their profiles AND their privacy preferences. This requires portable profile data and portable access control policies which travel with it. A typical scenario is as follows.

- John creates a social network (SN) profile on provider 1 with his name, birthdate, hobbies, workplace, etc.
- John closes the profile to public access.
- John sets granular preferences on viewing; only his friends can view his mobile phone data, but not his ex-wife.

Then EITHER

- John wants to join another social network provider, so he creates an account on provider 2.
- John exports his SN profile from provider 1 to provider 2.
- John exports his SN profile privacy preferences from provider 1 to provider 2.

OR

A social software application on another account John has already set up to be able to consume his profile data, eg,

- A web service independent of Facebook (eg, operating using OpenSocial), wants to be able to set up a party via SMS to John's friends.
- It contacts Facebook and asks for his network and their phone numbers.
- These are given out respecting the preferences set by the data subjects.
- It sends out invitations.

4.1.4 Online Personal Finance Management

Account aggregation allows an end-user to view details about transactions and balances in different accounts. Data might be entered by the user, downloaded from the institution where the account is held, or downloaded from an intermediary. Another example is found in *Mint – internet banking aggregation* (7), which provides Web 2.0 online account aggregation services (in the USA) – effectively an online equivalent of desktop accounting software. This kind of application is important from a security point of view because it involves the delegation of authorisation rights to access highly sensitive financial data.

4.1.5 Corporate IT Software and Browser-based Office Applications

It is now possible to get a 'desktop in a browser' offering the entire standard package of corporate applications working through the browser. Google Apps (8) is a good example. It includes spreadsheets and word processing to create corporate documents in the browser, hosted email, browser-based instant messaging, and a calendar. Businesses are also starting to outsource core functionality to web services, such as tracking sales leads, contacts, orders, etc, including payroll management. In future, a company may potentially outsource all its IT services except network infrastructure and hardware maintenance. An important feature is the fact that the application suite is given complete corporate branding while all data is usually hosted on a centralised repository provided by the service provider. Mostly the company has no contractual right to verify the security measures.

4.1.6 Collaborative Editing – Wikis

Wikis are collaborative knowledge-bases compiling content contributed by anonymous visitors and registered editors. In general, it should be emphasised that wikis support the open-source ethos encouraged by the European Union by providing easy access to contributors and consumers of knowledge. They encourage information sharing regardless of location or socio-economic standing.

Beyond the general quality of service (QoS) issues required of all websites, wikis present new and challenging security and integrity constraints. Semantic integrity is crucial to the success of wikis and must be factored into allowing access to contributors. Wikipedia, for example, strives to be a free encyclopaedia that (almost) anyone can edit. Beyond the feat of simply serving millions of articles, articles must conform to standards that ensure objectivity, factuality and relevance. It must prevent users from creating spurious articles, adding irrelevant text or expressing their own opinions contrary to the terms of service.

There are three main approaches:

- Allow unrestricted editing privileges: Although most visitors have good intentions, there are those who do not and this policy may foster rampant vandalism.
- Restrictive and bureaucratic editing procedures: Each time an edit is made, it will be reviewed and discussed thoroughly and be processed in a hierarchy. This approach is both slow and has lead to departures of very active editors.
- Autonomous agents: Scripts can reverse edits or provide notifications to human editors when edits are made. Although they are used extensively, these agents generally perform operations limited to regular-expression matching.

4.1.7 Blogging and Micro-blogging

Blogs – postings of regular user-generated content, usually maintained by an individual with regular entries of commentary, descriptions of events, or other material such as graphics or video – are already well known and need little explanation. Micro-blogging covers sites such as Twitter, Jaiku, Twitxr and Pownce which are a variant of social networking using short messages, usually along the lines of 'what are you doing now?' and even publishing images of the place or situation the user is currently in. The popularity of such applications is growing along with the possibility of posting these kinds of messages from any kind of device (web browser, mobile phone, IM, etc).

The subject matter of micro-blogging tends to involve posts and images taken in public, often involving unwitting (and unwilling) third parties. Furthermore the use of mobile devices can to add to the amount of location data made available. An important security concern is therefore the protection and proper usage of personal, temporal and geographical information.

4.1.8 Metadata – Tagging and Folksonomies

Image tagging: Typically used in social networking and photo sharing applications, users tag images with names or links to social networking profiles. They may tag images of themselves and crucially also of other people. EULAs and TOUs usually specify that tagging is opt-out, ie, if someone objects to being tagged, they must explicitly make a request to the tagger (whom they may not even know). Typically a user uploads an image and tags the image with metadata including the names, profile links or even email addresses of all the people in the image.

A key development in this area is the inclusion of face-recognition software in image upload tools. Such software uses face-recognition and machine learning to tag pictures automatically, based on a training period where the user 'teaches' the software to recognise the most common faces in their portfolio of pictures (9). As is described in more detail in *Security Issues and Recommendations for Online Social Networks* (10), this has important privacy implications. The widespread roll-out of such a feature will greatly increase the amount of tags applied to images.

RSS files: These files provide metadata on recently updated web content which allows users to aggregate multiple sources in a single reader, which displays summaries of new content instantly without the user having to check for updates.

Media metadata such as ID3 files: Media metadata provide information on media embedded in Web 2.0 applications.

Folksonomies and tag clouds: Users create community-based metadata schemes to which the members of the community can contribute. Tag clouds provide a visual representation of the relationships between the categories into which a piece of content falls.

4.1.9 Distributed Non-automatable Tasks

One of the uses of the new architectural paradigms is in harnessing the power of many distributed users to perform tasks which cannot be easily automated. This can be used for legitimate business purposes; Mechanical Turk is a good example (11). Such processes can also be used for criminal purposes such as breaking CAPTCHAs using a distributed user-base either paid a small amount for each CAPTCHA broken or otherwise enticed to enter the text of the CAPTCHA (12) (13).

5 Principal Risks

This chapter describes the most important privacy and security risks in Web 2.0. We focus on risks specific to Web 2.0 rather than general information-security threats (eg, identity theft, pharming), unless there is an Web 2.0-specific variant (eg, access control vulnerabilities), or the risk is increased or altered by some specific feature of Web 2.0 (eg, automation attacks). We have followed a typical risk assessment methodology. The analysis of risks is broken down into assets (the target of protection), threats (the potential negative impact), and vulnerabilities (the technical or systemic weaknesses which lead to the risk).

5.1 Assets

The assets at risk in Web 2.0 scenarios fall into a very few simple categories;

- Private information – can be stolen and used to harass, spam or slander a person;
- Financial assets – can be stolen via Internet banking or eCommerce portals;
- Corporate and personal reputation – can be damaged;
- Corporate secrets – can be stolen leading to financial and brand damages;
- Intellectual property – can be stolen;
- Computing and network resources – can be consumed leading to denial of service; and
- Physical security – can be compromised, eg, if an attacker is able to discover the physical contact details of a person.

5.2 Threats

Threats are potential damages to assets arising from the vulnerabilities outlined below. As these are generally not specific to Web 2.0, we only touch briefly to illustrate the consequences of not addressing the vulnerabilities. Typical threats and consequences include:

- Installation of botnets resulting in extortion by using the privileges of another application;
- Financial losses due to drive-by pharming, installation of malware, spyware or other vulnerabilities;
- Denial or corruption of reputation, or creating alternate identities in order to influence reputation, such as the well-known attacks on eBay's reputation system using a bot to create 1000's of fake accounts (14) or on a movie review site, using fake accounts to alter the popularity of a movie that will soon be released;
- Harassment using weakly authenticated accounts, eg, Twitter refuses to uphold terms of service (15);
- Internet user age verification threats, eg, bullying or sexual harassment of children as a consequence of discovering contact details;
- Spam from social networking sites or comment spam on blogs;
- Hiding the origin of a more serious attack by using an innocent victim;
- Consumption of user or corporate resources, eg, storage data in the user's account, denial of service, etc;
- Propagation of false and misleading information leading, for example, to stock market fraud or false medical diagnosis; and
- In banking scenarios, inaccuracies in data could lead to the imposition of financial penalties on users (not realising they are overdrawn).

5.3 Vulnerabilities

5.3.1 Access, Authentication, Authorisation

This group of risks concerns vulnerabilities which allow unauthorised access to applications and data.

5.3.1.1 Problems with the Same-origin Policy

The same-origin policy is one of the cornerstones of web security. In short, it prevents a document or script loaded from one origin from manipulating properties of or communicating with a document loaded from a different fully-qualified *domain name*, *port* and *protocol* (see [4.1.2] for more details). In Web 2.0, however, the common use of embedded widgets coming from third parties (see [3.1.3]) and the 'mashing up' of disparate data sources means that a single overall web application, operating in one browser instance, often needs to co-ordinate operations between executable code served from multiple domains. Conversely, executable code served from a single domain is often controlled by multiple mutually unknown and untrusted parties. As a result, in many Web 2.0 applications, the same-origin policy is seen as an obstacle to be circumvented.

Often a deliberate design decision has had to be been taken by browser developers to relax the same-origin policy in certain situations, in order to accommodate common requirements for cross-domain communication in web applications (even before the advent of Web 2.0 – eg, see POST requests described in [4.1.2]).

As a result, many techniques exist for circumventing the same-origin policy or abusing its deliberate relaxation, both for legitimate (eg, using Ajax Proxies, Flash, JSON, Script tags, etc) and criminal purposes. A common programming pattern is for http://a.com/sample.html to have code that adds a script tag like this to its Document Object Model (DOM):

<script src="http://b.com/data.js?callback=cb&param=foo"></script>

Ie, a script is loaded from b.com, and executed on a.com. That script is then expected to call the function cb, and passes the data that are retrieved to that function. Notably, execution control passes to the data provider, so the data provider (b.com) exercises full control over the web application running at a.com/sample.html (16).

The presence of such loopholes means *that even when the desired behaviour is to honour the same-origin policy*, it cannot be relied upon to protect an application from unauthorised access. Currently work is still in progress to develop a coherent technical policy framework for specifying when to allow relaxation of the same-origin policy, and a consistent and secure implementation of this policy.

5.3.1.1.1.1 Example of Same-origin Violation: Cross-site Request Forgery (CSRF)
CSRF is a common type of vulnerability which exploits weaknesses in the processing of HTTP requests (usually POST) coming from other domains. A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks and it is usually not an attack which can be detected by the requesting client since it exploits a legitimate and necessary feature of the client application (eg, the sending of a POST request from a third party application). A specific example of this kind of attack was the attack against Gmail (now patched) (17). In this attack, the victim visits a page in one tab of their browser while being logged into Gmail in another tab. Upon execution, the page performs an HTTP POST request to one of the Gmail interfaces and injects an email filter into the victim's filter list. This filter simply looks for emails with attachments and forwards them to an email of the attacker's choice. This filter will automatically transfer all future and existing emails matching the rule.

Note: this is an example of a more general class of 'confused deputy' problems (18).

5.3.1.1.1.2 Example of Same-origin Violation – Flash Cross Domain Policies
Adobe Flash plug-in software explicitly allows developers to relax the Flash implementation of the same-origin policy using a policy file. This in itself is NOT a vulnerability since it is part of a deliberate effort to provide a well-defined policy framework for defining when to relax the same-origin policy. The policy file is implemented on the server side by placing a single text file, named *crossdomain.xml*, in the root directory of the domain that hosts the web services potentially accessed by Flash files on another domain. Rules in the file specify which domains or IP addresses can make requests through to this server. (NB: it does not cover access to the DOM of the page the flash file is embedded in by flash files.) The vulnerability in this example lies in the fact that these rules also permit the use of wildcards (*) which allow *any* cross-domain request to be made to content served by this server. Web servers publishing crossdomain.xml files using wildcards are vulnerable to attacks, as described in *The Dangers of Cross-Domain Ajax with Flash* (19). The common development approach of 'how do I make it just work?' means that policy files with wildcards are extremely common. Most importantly, in a context where cross-domain access is unrestricted (ie, a wildcard has been used in crossdomain.xml), the browser's same-origin policy can be bypassed by loading a small SWF (Flash) file as an XmlHttpRequest proxy.

5.3.1.1.1.3 Online Advertising Networks
It is also common for online advertising networks to use various techniques to circumvent the same-origin policy. Such techniques include:

Http referrer headers: These headers, available to any embedded resource, give away the complete URL of the page the advert is embedded in – this allows advertisers to track surfers and even to extract personal information from URL variables. (This technique is known as web bugging).

Interception at the service provider: Emerging services such as Phorm (20) intercept and append advertisements based on the content.

Use of browser plug-ins: Some plug-ins do not apply the same-origin policy as strictly as the main browser. For example, Adobe Flash allows 'cross-domain policy files' which allow advertisers to circumvent the same-origin policy.

5.3.1.2 Excessive Privileges

Many Web 2.0 services ask users to delegate access credentials to, for example, email accounts, bank accounts, etc. Such services are often so compelling that some users are willing to suspend caution in order to participate, while the majority is put off by security concerns. ENISA's survey showed that over 70% of end-users are unwilling to use such delegation mechanisms and a majority would not use an online finance management service (aggregating many different accounts) for related reasons (1). This is a very rational choice because they offer users only one level of privilege and therefore if they want to do any kind of delegation, they have to give away the highest level of privilege (eg, unlimited, permanent access to all the features of their email account rather than just time-limited access to their address book). Moreover, this can cascade in surprising ways; an end-user providing their Google account information to an email aggregation service could also be providing access to other information, such as their Google documents. It is difficult for the majority of users to predict such implications.

Use of single linkable credentials also leads to leakage of information via tracking of the credential as a unique identifier. It also multiplies the vulnerability of all services to which the credential gives access. For example, social network aggregators reduce the security of social networking sites because a single password gives access to multiple repositories of personal data. Security is provided according to the weakest service.

The inability to provide fine-grained, limited authorisation and access-control to Web 2.0 applications is therefore a very serious vulnerability.

Another important issue related to the separation of control is the ability to use Web 2.0 applications in mashups as part of a distributed control system (ie, bot herding using blog posts, etc).

5.3.1.3 Insufficient Sandboxing

Sandboxing features offered by the 'IFrame' HTML tag are often insufficient. In particular, various features of IFrames allow cross-domain communication.

- Form controls within IFrames are accessible to the page in which the IFrame is embedded even if it is from a different domain.
- IFrames can 'spawn' new browsing contexts, modal dialogs and alerts in other domains.
- IFrames can navigate other browsing contexts.

See *Cross-Domain Communication with IFrames* (21) for more detail on these exploits.

5.3.1.4 Weak Authentication

Weak authentication and the existence of anonymous services, makes it easy to use Web 2.0 tools to steal account details (with all resulting threats) to impersonate people and publish 'on behalf of' (identity theft) to harass, insult or disparage people. The impact of this is augmented by the complexity of deleting published information and weak terms of service.

User Authentication
The standard way to authenticate the user is via a username/password web form. This is a very insecure method of authentication for several reasons:

- Passwords are easy to guess and are usually transmitted in clear-text (and are therefore subject to network eavesdropping and spyware attacks).
- If a user re-uses a password at multiple sites, then a malicious insider at one site can impersonate that user at other sites.
- There is no anonymity for users.
- Users are trained to type their passwords into potentially insecure web forms instead of an OS-level protected password box.
- Password recovery procedures are often weak.
- It is not uncommon to have different user bases on the same service with different levels of security (some really weak) while stronger authentication and security features apply to anybody who opens a new account. This generates a false sense of safety in the legacy end-user.
  For example, legacy accounts are allowed to change their passwords after a weak security question, whereas recent accounts are allowed to recover the password using a secondary email AND some stronger security questions.

In addition, more secure two-factor authentication schemes are not mature enough to be scalable. Many users are carrying an unacceptable number of bulky tokens. This means they are reluctant to take on further tokens from other providers. It is now a necessity to provide a 2-factor authentication scheme which can share a single (preferably credit-card form-factor) token among multiple providers. Tokens such as One-Time Passwords (OTP) and smart cards are also not sufficiently well-integrated into APIs and infrastructure.

Session cookies are another area where authentication can be weak. The use of easily guessable sequential numbers is an example of such a vulnerability.

Server Authentication
The standard way to authenticate a server is SSL/TLS. This results in multiple vulnerabilities for both users and service providers – for example:

- Users ignore SSL Certificate errors, enabling man-in-the-middle attacks to recover user passwords.
- Users do not notice domain typos and other anomalies, enabling phishing attacks.

Server authentication issues also play a role in some vulnerabilities in the same-origin policy [see 5.3.1.1]. The same-origin policy is based on DNS-resolved host names but network access ultimately uses IP addresses. When the browser starts loading network content, the host name is first resolved by the DNS system and after this the request is sent to the IP address, which defines the final destination. The origin of the content, for the purposes of evaluating the same-origin policy, will however still be determined by the DNS host name and the same-origin policy only works properly where there is no mismatch between the DNS host names and the IP addresses. Unfortunately DNS systems currently in use are vulnerable to a number of known attacks, such as DNS cache poisoning (see *Common Vulnerabilities and Exposures* (22)). Therefore, by using one of the well-known attacks on DNS, an attacker can circumvent the same-origin policy.

Once an attacker has exploited one of the above mentioned same-origin policy vulnerabilities, they have full access to the Document Object Model (DOM) of the compromised content, *even for SSL-protected documents*. This is because SSL protects the content on transport layer and is not applied to DOM elements once it is rendered in the browser.

A countermeasure against DNS attacks is to authenticate documents over an SSL-protected connection. However, there are situations where the attacker circumvents the same-origin policy despite the use of SSL. Consider the following example. Assume the browser loads document A from https://example1.com:443. The browser's security model grants another document the privilege to access this document through the Document Object Model, if the requesting document has also been loaded over an HTTPS connection from domain example1.com on port number 443. An important requirement at this point is that the SSL server certificate be valid. That means the certificate has been issued by a trusted third party for the domain example1.com. As discussed in [5.3.1.4], the average user does not properly verify server certificates. Thus, they may accept an invalid certificate (eg, issued for baddomain.com, from an untrusted third party) and the attacker then gains full access to document A. (For more information, see *Dynamic Pharming Attacks and Locked Same-origin* (23).)

Another problem arises from the retrieval of scripts. Consider again the example from Section [5.3.1.1]. Assume that document A from https://example1.com:443 loads the script <script src= "http://b.com/data.js?callback=cb&param=foo">. The purpose of the script tag is to load the script data.js from b.com. The browser's same-origin policy however treats this script like an object that has the privileges of https://example1.com:443. The crux is that the attacker may replace data.js by a malicious script because it is triggered over an unauthenticated channel. Thus, the attacker may inject arbitrary script code into the security context of https://example1.com:443 without the user noticing. In fact, this attack can be considered as a side channel attack where the attacker injects inline malicious script code into a valid SSL communication session. (For more information, see *Beware of Finer-Grained Origins* (24).)

5.3.1.5 Age Verification

Many services require age verification to access certain resources and this is a legal requirement in many jurisdictions. All currently available solutions present important unsolved vulnerabilities; for example (rising upwards in strength):

- Simple click-through in agreements, which require the user to state they are over 18, do not require a visitor to present any sensitive information that can be abused or stolen but are easily subverted because there is nothing other than the user's integrity to prevent false assertions. On the flip side, any other viable solution that guarantees the age of requestors, will infringe on their right to privacy.

- Email accounts issued at birth by government agencies can be used to verify age. But email accounts can be attacked with phishing or passwords can be delegated resulting in even more serious damage.

- Possession of electronic national ID cards (smart cards) underwritten by governments is proven by a PIN or password. But these can be delegated (experience with credit cards shows that people will delegate even highly sensitive tokens), and often disclose far more private data about the individual than just the fact that they are over or under the age of majority.

Note that verifying that someone's age IS greater than 18 may be easier than verifying that their age is less than 18, as there is a larger number of use-cases which require proof that a person has reached the age of majority and also a number of documents, such as driving licences, which are only available to people over a certain age. The key elements of a truly reliable age verification solution are:

1. Government underwritten registration procedure – registration and enrolment must be carried out by trusted personnel, without commercial motivation to fraud.

2. Strong proof of possession – it should not be possible for anyone except the person issued the token or secret to use it. Passwords are one solution here but people (especially minors) often give away both their token and its password to another party. One partial solution is to use the disclosure of more sensitive data as a disincentive to password delegation. For example, the use of a secure email address issued at birth has this effect because delegating a password would give access to all the user's mail. On the other hand this makes the damage much greater if it does happen.

3. Inexpensive – placing fingerprint readers in every key of a keyboard so you can see who is typing every key is possible, but prohibitively expensive (and a serious invasion of privacy).

4. Convenient – one way to get round age restrictions would be to remove home access to the Internet and force users to use certified work spaces if they wanted to use the Internet, but this too inconvenient.

5. Not intrusive – making people wear brainwave-scanners or have RFID chips in their hands would work, but is too intrusive.

Unfortunately no current solution satisfies all these requirements. Nevertheless, ENISA's survey (1) showed that only about 50% of Web 2.0 users believe that age-verification mechanisms are untrustworthy. See the recommendation on awareness-raising [6.1.3] for further discussion.

5.3.1.6 Content Filtering

A related problem (as it is another barrier which should protect minors) is that content filtering software is less effective in Web 2.0 environments because content is less predictable and depends on the culture of the user-generated content, which may change rapidly. Studies such as *Safer Internet* (25) have demonstrated this problem. *Half of the 23 filters we tested both in 2006 and 2007 have improved their filtering capabilities relative to non-sexual content. Unfortunately, eight vendors scored weaker than last year relative to sexual content ('porn-only'), partially because our test cases now included Web 2.0 content which is more difficult to filter*. Nevertheless many users place trust in such mechanisms in a Web 2.0 context; ENISA's survey showed that only 24% of users stated that they do not trust such tools in a Web 2.0 environment.

5.4 Development Process Issues

This group of risks concerns features of development processes which affect Web 2.0 applications.

5.4.1 Client-side Development – Complex APIs

Web 2.0 applications increasingly move security critical functionality to the client. When a mashup handles data from differently trusted sources inside the web browser then that client side code becomes security critical. The coding skills of developers, and the complexity of APIs, are not commensurate to the difficulty of this task. A predictor for the code quality in this area can be seen in various widget engines, which use web technologies as an environment for developing local (and security-critical!) applications. Examples of vulnerabilities resulting from poor code quality in widgets can be found in *Widgets – Web Vulnerabilities for All* (26), *Web Application Security Issues* (27), and *24C3 lightning talk* (28).

5.4.2 Short Development Cycles

In addition, because of the extremely competitive and fast-moving market in which Web 2.0 applications operate, most Web 2.0 applications have extremely short development cycles. This means that security is often overlooked – no or inadequate vulnerability assessments are performed and documentation is often lacking or incomplete (eternal Beta versions, etc). In fact, the end-users are becoming the new beta testers. Even in 'security aware' web companies, widgets are often developed in short time and with a low budget. Third parties serving widgets almost never do security checks on them.

5.4.3 Example: Cross-site Scripting (XSS)

Cross-site scripting (XSS) is probably the most widely publicised class of security vulnerability associated with Web 2.0 applications and one of the key drivers behind the increase in Malware attacks coming from Web 2.0 sources. XSS flaws occur whenever an application takes user-supplied data and sends it to a web browser without first successfully validating or encoding that content. In other words, the root cause is poor development practice, although it can be addressed to a certain extent by smarter IDEs or better type management in scripting languages.

XSS allows attackers to execute script in the victim's browser which can hijack user-sessions, deface websites and introduce worms. There are many flavours of XSS but we describe a simple scenario below for illustration. See *XSS Attack Information* (29) for many examples and more information.

5.4.3.1 Simple XSS Example

A common example of XSS in a Web 2.0 context is when angle bracket characters '<' (used for opening and closing tags) are not correctly escaped. If a Web 2.0 application allows users to enter comments on a page where the characters '<' and '>' are not escaped, any user can then enter a comment containing a <SCRIPT> tag (with arbitrary Javascript inside) which is then be executed in the browser of most visitors to the page containing the comments. Such a script can then perform all kinds of malicious actions such as creating a hidden IFrame, which constructs a URL with query parameters containing information taken from the parent page, and then sets the IFrame's 'src' to a URL on a server under the attacker's control, ie, it can steal information from another browser page or tab in the user's environment.

5.4.4 Inadequate Specification of Security and Privacy Guarantees and Assumptions

Security and privacy guarantees made by web applications or intermediaries (such as mashup or portal sites) are rarely specified in a formal or precise way. As a result:

- It is challenging for system implementers to be assured that their systems do meet their stated security guarantees, especially as code is maintained and new features are added.
- Developers who combine web applications and need to make assumptions about security guarantees may be forced to check informally-specified policies manually. Such policies may be hard to interpret, and subject to change.
- End-users are often not aware that their right to privacy is being infringed because insufficient, or insufficiently clear, information is given about data handling practices.

For example, when the Facebook Beacon service was introduced, there was initially an inadequate specification of the data collection and re-publication practices (30).

A related security problem for widgets is that the owner and developer does not know where the widget will be served, in what context and on what sites. This is a potential brand management issue: sometimes branded widgets are served in an inappropriate context.

5.4.5 Method Exposure in Migration

Several toolkits exist which automatically convert server-side applications to Web 2.0 style AJAX applications. In this case, server-side applications will invariably have so-called 'private methods' – features of the code which should not be accessed from the outside. Several toolkits exist which automatically convert server-side applications into dynamic client-server applications (eg, using AJAX); see Google web toolkit (31), Direct Web Remoting – Client-Server AJAX Framework (32), and ASP.NET AJAX (33). Automatic conversion processes which convert code without applying a clear human-assisted policy to protect private methods can inadvertently expose sensitive data and methods with harmful consequences. This risk is exacerbated by the fact that some frameworks (eg, Direct Web Remoting (32)) provide a public downloadable list of methods available.

5.5 Knowledge and Information Management

This group of risks relates to problems with information provenance and integrity and the management of personal information in Web 2.0 environments.

5.5.1 Fraudulent Pedigree/Provenance

The use of the syndication pattern described in [3.1.1], as well as intermediaries such as social bookmarking applications, mashups, portal services, or advertising or monitoring proxies on ISPs means that information is often processed, rendered and altered through multiple steps before it is finally displayed to the end-user. A lack of non-repudiation mechanisms and integrity checks in news stories, blog posts, and social data means that misinformation is therefore easily spread and propagated, often with very serious consequences. In addition, end-users do not use appropriate methods to determine the authenticity of content. ENISA's survey showed that most users will trust content simply because it is repeated at many URLs on the web (1). There are several serious attacks which exploit the inability to establish authentic provenance and pedigree:

5.5.1.1 Example 1: Stock Price Manipulation

False rumours spread via Web 2.0 applications have been used to manipulate stock prices in order to gain from price inflation due to rumours. For example, *Social Investing and Pump-and-Dump 2.0* (34) describes how so-called social investing software has been abused in order to inflate stock prices and *Could Digg be used for Suns stock manipulation?* (35) discusses how social bookmarking can be used for stock price manipulation. Price distortions can also happen due to less systematic attacks – for example, Apple Computer's stock price fell when technology blog Engadget posted a supposed fake 'internal memo' indicating a significant delay in the releases of the much-anticipated iPhone handheld device (36).

5.5.1.2 Example 2: Control of Botnets via Mashups

Botnets are networks of millions of small software programs installed by malware on users' machines which are used for extortion (usually financial) – for more information see *Botnets – the silent threat* (37). Botnets are normally controlled by channels such as Internet relay chat (IRC); however security researchers have recently identified attacks which use Web 2.0 features such as blog postings and email to RSS interfaces to control Botnets. The use of such channels for command and control of the botnets makes tracing the attacker much more difficult and, because of the open nature of the mashup architecture, it means that control interfaces are very difficult to shut down. Mashups are perfectly suited to massively distributed systems with untraceable control structures and are therefore likely to lead to a variety of related attacks (see *Use of Web 2.0 technologies to control botnets* (38) and *With Web 2.0,* a new breed of malware evolves (39) for more information).

5.5.2 Vulnerabilities of Collaborative Knowledge Systems

The most unique vulnerability of collaborative knowledge-bases is the editing mechanism. By default, articles on a wiki grant 'write' privileges to both anonymous and registered users. The access control is generally very coarse, falling into the following categories:

- Editable by anyone;
- Editable only by registered users; and
- Editable only by editors.

There is usually little or no means of establishing trust, integrity or the provenance of statements made. For example, there is often no validation of user-identity, no signature available for content and often no use of reputation-based systems. If reputation-based systems are used then the vulnerabilities in these – eg, collusion, whitewashing, Sybil attacks, etc – are often not addressed. For more information, see *Reputation-based systems: a security analysis* (40).

5.5.2.1 Example: Medical Data

The unreliability of information can have dangerous consequences – for example only 25% of users surveyed in *Online Health Search 2006* (41), who search the Internet for health advice regularly, check the source and date of the information they find to assess its quality. Tools such as Wikiscanner (42), which trace source IPs of collaboratively produced articles, show that the deliberate introduction of bias and misinformation is frequent.

5.5.3 Metadata Attacks

The use of metadata formats such as ID3 (for mp3), RSS and other RDF metadata is central to the operation of Web 2.0 applications. Metadata has several important security vulnerabilities:

- Integrity and trust infrastructure for metadata is virtually non-existent – ontologies, tag clouds and schemas can often be altered or substituted or used to provide an attacker with extra information via unintended inferences (43).There are also serious problems with digital signatures for RDF data such as ontologies and RDF, which means that tools for digital signatures over RDF are non-existent (44).

- There is no standardised way to add provenance data to most metadata formats – this lies at the root of many of the knowledge management vulnerabilities described above, eg, those related to syndication.

- Metadata misuse is often the source of phishing attacks because it can be used to mislead the user about where a web resource comes from, eg, a wrong link attached to an image.

- Hidden metadata on Web 2.0 pages and sources can disclose private information – as the case of Wikiscanner (42) shows – as well as metadata hidden in documents; see *Microsoft Word's Hidden Tags Reveal Once-Anonymous Peer Reviewers* (46) and *Psst! Your Metadata is Showing!* (47) for examples. Metadata is often the source of malformed syntax leading to buffer overflows – there are thousands of examples of such vulnerabilities – one example being CVE-2008-0065 (45). More web-media-related examples are described in *Exposing Vulnerabilities in Web Software* (48).

- Codecs, a binary format used to provide interoperability for media display formats, although not strictly speaking metadata, are another important related class of add-on to Web 2.0 media which contains considerable vulnerabilities, some of which are also described in (48).

5.5.4 Privacy Violations

Theft or misuse of personal data was the top concern of end-users in a Web 2.0 context according to the ENISA survey (1). This stresses the importance of upholding the fundamental rights to privacy, anonymity and informational self-determination enshrined in, for example, the European convention on Human Rights (article 8) (49) and European data protection laws such as Directive 95/46/EC (50).

5.5.4.1 Image Tagging and Search

The posting and tagging of images or reports by private individuals is an increasing problem where users publish images without the consent of the subject. This is exacerbated by the introduction of face-recognition software into mainstream photo-sharing applications such as Google Picasa (9), leading to a huge increase in the number of tags in photos which therefore means that photos are easy to find and link together. This has serious implications for privacy, particularly since they are distributed with EULAs which stipulate that tagging is on an opt-out basis (ie, if you don't like the fact that someone has tagged an image, you must explicitly ask the publisher to remove it). In fact, ENISA's survey (1) showed that a significant number of users are unhappy about the tagging of images in various scenarios: 60% are unhappy about a colleague tagging a photo with a social network profile and 44% unhappy about a friend doing the same. Such software could also be used for linking anonymous profiles to personal data by linking together similar images.

The issue of image posting by private individuals is being taken seriously by governments – Poland, for example, has a draft bill before parliament (as of August 2008) for a new penal code which contains an article introducing a penalty ranging from 3 months' to 3 years' imprisonment for distributing pornographic images of a person without their permission and a penalty from a fine to 2 years of imprisonment for distributing naked images without permission.

5.5.4.2 Behavioural Marketing

Behavioural marketing is a very common source of revenue in Web 2.0 applications where content and the exchange of personal data are often the main value proposition. Marketing companies generally operate on the premise that anything which does not include explicit personal data fields such as name, address, email or birthdate is not personal data – and therefore not subject to data protection laws.

We note that such a premise is extremely questionable with respect to the definition of personal data outlined by the Article 29 Working Party (51) whereby an IP address may be considered personal data. The issue of what really is anonymised data is a much more complex issue. Data on hobbies, friendships and search terms, etc, may be extremely personalised and may serve to identify an individual just as much as their name, address and birth date. Furthermore their knowledge by a third party may result in negative consequences on their personal or economic life and/or discriminatory exclusion from services. This leads to a situation where some providers are processing data which is in reality extremely personal data (by virtue of being specific to an individual and thereby able to identify them) but are technically not subject to data protection law. This is the subject of the debate around behavioural marketing.

Furthermore, only 7% of users surveyed by ENISA believed that such marketing was a positive development because it makes marketing more relevant, while 23% believed that it should be illegal, and 28% accept it because it is necessary to run the service but are not happy about the practice (1).

5.5.4.3 Other Privacy Issues

- Inference control: Beyond the possibility of inferring links between profiles from tagged images, another threat is the ability to make inferences among multiple distributed instances of data, none of which is personally identifying in isolation.

- Disclosure of highly sensitive personal data on social networks and blogs: Many users are not aware of the potential size or nature of the group having access to postings on social networks. This is discussed in more depth in *Security Issues and Recommendations for Online Social Networks* (10).

    - Data retention and persistence: Web 2.0 users typically post data on a large number of sites controlled by private individuals (eg, blogs, SN profiles), ie, in a very distributed pattern which makes rectification or deletion of statements very difficult. Another factor which makes deletion of data more difficult is the replication of data from one site to another, ie, one website takes data from another website, then republishes it as in the architectural pattern described in [3.1.1].

    - Leakage of corporate secrets via insecure corporate IT software working via a browser: The risk of data leakage associated with the use of browser-based software for enterprise operations are well-known. The two main vulnerabilities here are the insecurity of the browser environment as highlighted in [5.3] and the possible transfer and storage of corporate data to a web-based data centre.

5.5.5 Terms of Service Problems

Since Web 2.0 services host user-generated content, the service providers are often under pressure to police (censor) that content. For example, service providers may intervene for any one of the following reasons:

- A user has posted inappropriate or illegal material;
- A user has posted copyrighted material;
- A user has defamed or posted private information about another person;
- A user has made comments critical of the service provider; or
- A user has made comments that are off-topic for the forum.

Some of these interventions may be initiated by other users, eg, when a service provider removes copyrighted material at the request of the copyright holder, but others are initiated by the service provider, eg, removing statements critical of the service provider. Similarly, some of these actions are acceptable, while others clearly violate users' rights.

Many users want service providers to filter spam and other content but, as service providers become police-officers, they give up the legal protections granted to 'mere conduits' (in the US, known as 'common carriers') and create a host of conflicts-of-interest.

This is considered to be a particularly important problem by Web 2.0 providers because of the strong user-generated content component and the high financial and brand-related risks associated with related incidents.

5.5.6 Low Use of TLS/SSL

Many service providers transfer highly sensitive information, including passwords giving access to additional information, between client and server without encrypting it or authenticating the service provider using standard mechanisms such as TLS.

One of the major impediments to the widespread use of TLS is its poor usability. Users often have difficulty deciphering information given about certificates and especially error messages generated by non-verified certificates, or are simply overwhelmed by excessively frequent messages.

5.6 End-user Related

This group of vulnerabilities relates to the usage of Web 2.0 environments and the understanding of the security issues by end-users.

5.6.1 Public Awareness Problems

Lack of public awareness of the following issues undermines security and safe Internet behaviour in Web 2.0 environments – for a full list of issues recommended for awareness-raising campaigns see *The new users' guide: How to raise information security awareness* (52):

- Difficulty of data deletion – users do not appreciate the difficulty of removing all traces of such information in the face of replication of data from one site to another, e.g., where one website takes data from another website, then republishes it.
- Audience – as explained in (10), users of social networks are often unaware of the size or nature of the audience to whom content is available.
- Users (in personal, corporate, or governmental roles) may believe and replicate false or unsubstantiated information that is posted to the Internet, including social networking sites, by malicious parties or parties with commercial interests.
- Users do not fully appreciate or understand the various security mechanisms already in place on websites such as SSL/TLS certificates or CAPTCHAs (53).

5.6.2 Lack of Context on the User Interface

Users are used to trusting websites based on the name of the site, or the name of the company in the URL. If they trust the URL, they will interact freely with the website. With many Web 2.0 technologies there is either no comparable context that the user can base trust on, or if there is a context, it can be misleading.

Examples:

- Portals, mashups, IFrames, etc, incorporate content from multiple sources on a single page [see section 3]. The user may be used to providing sensitive information on other pages on that site (eg, Yahoo Stores) and continue that behaviour even when the overall service provider is presenting content from a source the user would not otherwise trust.
- Many mainstream websites provide a 'comments' section for anonymous users to add their own content to the main page. Many users scan long articles quickly, and can easily mistake a well-worded comment for part of the main article. This can allow spammers and advertisers to give their own content the authority or cache of the more reputable site.

This kind of vulnerability can be traced to a lack of well-established visual-design standards for indicating that content should not be trusted at the same level as the primary site or, alternatively, an over-reliance on the user to make trust decisions.

5.6.3 Automation Attacks (Pretending to be an End-user)

The problem of knowing whether a system is controlled by a human being or not is a serious vulnerability in Web 2.0. The ability to automate a process which is generally believed to be controlled by a human being can allow attackers to abuse trust and circumvent controls to prevent access via brute force attacks. Examples of attacks of this nature are:

1. One of the main defences against unwanted automation is a CAPTCHA – an image with text which has been rendered in such a way that it is difficult for a software program to read but is readable with reasonable ease by a human being. It is used to protect systems such as messaging and account creation which should not be automatable.

   Various high-profile attacks on CAPTCHAs have highlighted the vulnerability of such systems. For examples, see *Yahoo's CAPTCHA Security Reportedly Broken* (54), and *Spammers crack Gmail CAPTCHAs* (55). The ability to break a CAPTCHA (or other anti-automation techniques such as filtering) opens up the possibility of spamming, Sybil attacks on reputation systems (40), and infiltration of social networks, as well as harvesting personal data. For example, Friendbot (56) allows the creation of automatic 'Friends' on MySpace which allows the controller to send spam, harvest personal data, etc. Future trends could also include tools to automate the creation of online personas across distributed sets of evidence (blogs, forums, etc) for the purpose of fraud or political gain – see *Investigating individuals and organisations using open source intelligence* (57).

2. Social engineering using autonomous chat agents where individuals engage the bot as if it were a real person and divulge sensitive information (58) (59).

3. Automation that solicits humans to solve CAPTCHAs as part of an attack (60).

5.7 General Software and Scripting Vulnerabilities

This section relates to vulnerabilities in software (not including access control and authorisation). We do not discuss general vulnerabilities of infrastructures such as networks and web servers which affect Web 2.0 applications in the same ways as any other web application.

5.7.1 Vulnerabilities on Web-enabled Devices

A wide number of devices are web-enabled and run de-facto web servers, even though we may not think of them as such. One example along these lines is home broadband routers. Because these devices are web enabled, there is a risk that they are vulnerable to Web 2.0 based attacks, such as cross-site request forgeries (CSRF). One type of attack that was discovered along these lines is drive-by pharming (61). In a drive-by pharming attack, an attacker can include a specific piece of HTML code on a web page (or even email message). When the code is rendered on the victim's machine, it surreptitiously logs into the victims home broadband router and modifies its DNS settings. The attacker can either specify an entirely different DNS server than the one the victim was using before or the attacker may simply modify specific host to IP mappings (such as those associated with the victim's bank). This particular threat has actually been observed in the wild. In a particular case, victims in Mexico were being targeted and the specific router model was associated with one provided by a large Mexican ISP. The attacker's HTTP request to the victim's router modified the host to the IP address mapping of a large Mexican bank.

Another interesting class of Web 2.0 specific vulnerability affecting infrastructure is the possibility of port-scanning using XMLHttpRequest (XHR). For example, *Online port scanner* (62) offers a legitimate, consent-based service for vulnerability testing including an XHR based online port scan test.

5.7.2 Browser Plug-in Vulnerabilities

Web browsers have frequently been forced to withstand the brunt of web-related attacks; typically, attackers try to exploit one or more vulnerabilities on a web browser so that they can use it as a conduit to get malicious software onto the victim's machine (in the form of a drive-by download). However, attacks on web browsers now seem to be going out of favour and are being supplanted by attacks on browser plug-ins. With users constantly looking for a richer browsing experience, more and more are augmenting their browsers with popular plug-ins.

To provide some perspective, according to volume XIII of the Symantec Internet Security Threat Report (63), during the second half of 2007, there were 88 vulnerabilities reported in Mozilla browsers, 22 in Safari, 18 in Internet Explorer, and 12 in Opera. In the previous six month period, Internet Explorer was subject to 39 vulnerabilities, Mozilla to 34, Safari to 25, and Opera to 7. However, Symantec documented 239 browser plug-in vulnerabilities in the last six months of 2007, compared to 237 during the first six months. During the second half of 2007, 79% of these vulnerabilities affected ActiveX components, compared to 89% in the first half. These numbers clearly show that browser plug-in vulnerabilities far outnumber traditional browser vulnerabilities. Just one example of a plug-in based vulnerability is described in [5.7.2].

5.7.3 JSON Vulnerabilities

JSON – JavaScript Object Notation is a lightweight format for exchanging Javascript objects data between client and server. There are several important vulnerabilities using JSON. They fall into two classes:

1. Attacks exploiting the fact that JSON requests can be included in SCRIPT tags which evade the same-origin policy and allow data to be extracted from objects. This allows Javascript to be requested from an arbitrary domain and the results of inspecting objects returned can then be sent to an arbitrary domain. This allows an attacker to request personal data about an arbitrary client (requested by that client from a malicious web page) and then send it to an attacker's web server using, for example, an XMLHttpRequest. This vulnerability was used in 2006 when an attack allowing the theft of Gmail contacts was discovered (64). Another example is CVE-2008-1318, an attack on Mediawiki (65). This is a particularly important vulnerability for applications installed on a corporate intranet which can leak data should a user visit a rogue site.

2. Attacks exploiting the fact that JSON objects are sometimes executed directly on the client without validation using the eval() function. This is done for legitimate purposes to load an array into memory, but an attacker can include malicious code in the JSON request/response instead of just an array constructor.

6 Recommendations and Countermeasures

In this section, we make recommendations for mitigating the risks outlined in the previous chapter. In order to link to the vulnerabilities, we refer in each recommendation to the vulnerabilities it addresses.

6.1.1 Government Policy Recommendations

6.1. 1. 1 Policy Incentives

- In order to address the issues raised in [DEVELOPMENT PROCESS ISSUES], governments should offer policy incentives for secure development practices. Examples of possible incentives could be:
  - Certification-lite: Standard security certification schemes (see *Information Security Certifications* (66)) such as ISO 27001 and Common Criteria are too expensive and time-consuming to be feasible for the smaller enterprises often at the heart of Web 2.0 application development. Certifications-lite, cut down and therefore cheaper versions of security certification schemes, suitable for micro-enterprises or SMEs are therefore particularly appropriate to foster better security practices among Web 2.0 providers.
  - Reporting exemptions: Another useful model for policy incentives is the Swiss federal data protection act, which sets out voluntary certification requirements and motivates businesses to comply by providing exemption from expensive reporting.
  - Funding of pilot actions integrating security features in Web 2.0 applications: An example of a pilot action already in progress is the STORK project (67).
- To address issues raised in [TERMS OF SERVICE PROBLEMS]: Governments should investigate and address Web 2.0 provider concerns about conflicting obligations brought about by legislation on mere conduit or common carrier (US) status. In order to qualify as a mere conduit of information, service providers must not intervene in any way in the content served. This has led to considerable uncertainty among providers about when and how to engage in any kind of content filtering. This is of special relevance in a Web 2.0 environment because of its strong user-generated content component. Web 2.0 providers would therefore benefit from a government led initiative to clarify this area.
- Behavioural marketing – see vulnerability [BEHAVIOURAL MARKETING]: Bodies such as the European Commission, the European Parliament's ITRE and LIBE committees and the Article 29 Working Party on Data Protection encourage open public and intergovernmental discussion on the appropriate balance in Web 2.0 applications between, on the one hand, identification by marketers, law enforcement and private individuals and, on the other hand, the privacy and anonymity of end-users.

Note that the next section on research directions is also relevant to governments.

6.1.2 Research Directions

[Related Vulnerabilities: All]

We recommend that the following should be the subject of research funded by industry and governments:

- Usability of TLS/SSL – see vulnerability [LOW USE OF TLS/SSL]: As described in [5.5.6], TLS/SSL and encryption of data transferred in Web 2.0 applications are not used widely enough. This research is needed in order to increase use of end-to-end encryption and improve provider authentication in Web 2.0 applications.

- Usability of stronger authentication mechanisms (eg, one-time passwords) – see vulnerability [WEAK AUTHENTICATION]: Stronger authentication helps to protect sensitive user data and should therefore be encouraged in certain contexts. (Note it is only appropriate where sensitive data is to be protected.) One of the most important obstacles to its widespread adoption is its burden on the user. If strong authentication is to become a practical alternative in Web 2.0, a number of issues must be solved:
  - It is not practical to have a different physical token for each service. Research is needed into the means for sharing tokens between multiple services which do not necessarily share the same user data.
  - Users have difficulty with the current plethora of user experiences (eg, mobile phone based systems, key fobs, credit card form factor tokens, etc). Research is needed into the means for standardising the user experience in strong authentication.
- EU funded projects should pilot and assess the use of stronger authentication mechanisms such as OTPs, Smart cards, SMS based passwords, etc, in Web 2.0 environments in order to encourage stronger authentication environments; eg, the EU funded project STORK already plans to pilot a safer chat application. (See vulnerability [WEAK AUTHENTICATION].)
- Defence against abuse of finite resources: Web 2.0 applications need to offer open public interfaces for the input and exchange of user-generated content, the creation of accounts, etc. These are often abused for unintended and malicious purposes such as spamming (see [5.6.3]). Technologies such as CAPTCHAs, which restrict the usage of such resources to legitimate patterns (eg, humans only) are a serious weak point in Web 2.0 environments which needs to be addressed in research. (See vulnerability [AUTOMATION ATTACKS (PRETENDING TO BE AN END-USER].)
- Trust infrastructure using social networks: Social networks represent a considerable potential for gathering trust information using novel trust schemes. For example social networks could be used to create highly scalable public key trust using a web-of-trust (68) style scheme. Aggregated trust embodied in reputation scores collected from social networks can also help to filter content and authenticate assertions.
- Means of establishing provenance of information – see vulnerabilities [KNOWLEDGE AND INFORMATION MANAGEMENT]: In order to mitigate the risks described in [5.5], research is needed into mechanisms for tracing and assessing the pedigree and trustworthiness of information sources. Web 2.0-specific requirements for such as standard would include:
  - Respect for the privacy and possible desire for anonymity of the source – efforts in a related area have been criticised for ignoring this; see reference (69);
  - Ability to trace similar versions and their morphology; and
  - Trust metrics for content (eg, based on web topology).
- Advanced security models for ECMAScript – see vulnerability [GENERAL SOFTWARE AND SCRIPTING VULNERABILITIES]: Javascript has been the source of a large number of vulnerabilities [5.7]. A number of initiatives already exist (eg, Caja (70)) and while we do not favour any particular initiative we emphasise the importance of research in this area.
- Licensing models and contracts for personal data – see vulnerabilities [PRIVACY VIOLATIONS]: Any initiative which increases the rights of end-users over the use and re-use of their data or any content they generate should be encouraged. There are currently few legal or contractual mechanisms available to regulate the non-commercial use of a private individual's data even though several serious threats are posed by such scenarios.

- Metrics and testing frameworks (code analysis, penetration testing) for secure and privacy-respecting Web 2.0 applications: Research is neede here in order to encourage and facilitate secure development practices. (See vulnerability [DEVELOPMENT PROCESS ISSUES].)

6.1.3 Awareness-Raising

Awareness-raising is a very important countermeasure to many of the threats outlined above. More information on how to conduct awareness-raising campaigns can be found in *The new users' guide: How to raise information security awareness* (71). In general, it is important to use in-context methods to raise awareness. For example, when logging in, a link could be provided to a video about how to detect account compromise, or about the advantages of stronger authentication methods. When creating user-generated content, a link could be provided to a user-friendly explanation of the user's IP rights. This section describes the issues we consider most important to highlight in Web 2.0 related awareness-raising campaigns.

- Lifetime of data on the web – the fact that even when (as is not common) it may be easily deleted, user-generated content may persist in caches, back-end data stores, archives, etc. ENISA's survey (1) showed that a significant number (59%) of users had wanted to delete data after they had provided it and half that number had even decided to ask the service provider to help them do so.

- The data which may be revealed through search results in a Web 2.0 context (eg, profile data exposed in social networks).

- Use of access control – how to use privacy features and other access control features in Web 2.0 applications.

- Promote the use of existing etiquette guidelines on web 2.0 behaviour (eg, *Get Safe Online Guidelines* (72) or in a corporate context (73)).

- Benefits and usage of stronger authentication methods and username/password alternatives.

- All users should be made aware that age-verification mechanisms currently have a low success rate and therefore that they should not place too much faith in the claimed age of other users.

- Minors should be educated on how to detect and respond to inappropriate behaviour. The following is a suggested check-list:
  - Never give away your physical contact details;
  - Never agree to meet anyone you have met online in person, especially without consulting an adult first;
  - Do not respond to inappropriate (bullying, obscene, or offensive) messages and report them to an adult;
  - Never give away your account password or username; and
  - Be aware that other players may give false information about real-world characteristics.

- Parents and teachers should be made aware of all the above risks in order to better protect their children. This information should be transmitted through multiple media channels since many parents do not use Web 2.0 applications.

- Parents should be made aware in particular of the risks of relying on age-verification mechanisms to protect children and of the consequent need for non-automated monitoring of Web 2.0 usage – see vulnerability [AGE VERIFICATION].

- Parents and teachers should be made especially aware of the risks of relying on content filtering tools – see vulnerability [CONTENT FILTERING].

- Personal data risks and protection – these are described in more detail in [5.5.4] and *Security Issues and Recommendations for Online Social Networks* (10).
- Privacy enhancing technologies – alternatives available and their functionality. For more information see *Privacy and Identity Management for Europe* (74).
- Interpretation of TLS/SSL signs and errors in browsers; For example, the meaning of expired certificate errors, certificate not issued to this web server, etc. (See vulnerability [LOW USE OF TLS/SSL].)
- Developers of Web 2.0 applications and browser vendors should be made aware of security threats and secure development practices.

6.1.4 Standardisation

- Emerging specifications, such as W3C's Access Control for Cross-Site Requests (75), OASIS XACML (76) and oAuth (77), OpenID (78), Cardspace (79), Liberty (80), HTML 5 (81), etc, which are working towards the provision of better separation of control, better access control and better authorisation functionality in web applications should be further developed and promoted. In particular, the following are urgently needed:
  - A comprehensive framework to specify policies governing the exchange of data and access to function calls between web applications and widgets: It is very important that plug-ins should be included in such a framework – see vulnerability [PROBLEMS WITH THE SAME-ORIGIN POLICY] – and that appropriate default policies should be encouraged.
  - A granular authorisation framework allowing, in particular, the delegation of access in Web 2.0 applications – see vulnerability [EXCESSIVE PRIVILEGES]:
    - for a limited period,
    - to limited fields,
    - to a specific individual,
    - with traceability in case of abuse (but nevertheless respecting privacy).

    (It may be that XACML could fulfil this role but there is not much uptake in Web 2.0 environments).
  - Consistent interpretation of security related messages in browser environments – for existing related work, see *Web Security Context: User Interface Guidelines* (82).
- Privacy preserving information provenance and pedigree – see vulnerabilities: [KNOWLEDGE AND INFORMATION MANAGEMENT]: While digital signatures offer an existing means of establishing some degree of information pedigree and provenance, they are not appropriate for Web 2.0 content because:
  - They rely on a public key infrastructure which is not in place;
  - They usually identify the author strongly; and
  - They rely on canonicalisation of content which may be difficult in the case of related Web 2.0 content.

  There are emerging standards for information provenance such as *The EU Provenance Project* (83), but nothing exists in the context of Web 2.0. Requirements would be similar to those described in 6.1.2]. In this area of standardisation, we stress the possible privacy risks which must be addressed, ie, the privacy and possible desire for anonymity of the source should be respected.

6.1.5 Provider Issues

- Standardise and consolidate the implementation of stronger authentication mechanisms wherever appropriate. (See vulnerability [WEAK AUTHENTICATION].)

- Use authentication measures which are appropriate to the data being accessed: for example, login and check account status using just a plain password, but to do financial operations, re-authenticate using a one-time password token. (See vulnerability [WEAK AUTHENTICATION].)

- Use TLS/SSL based encryption wherever sensitive data such as passwords and personal data are transmitted. (See vulnerability [LOW USE OF TLS/SSL].)

6.1.6 Developer Issues/Browser Vendors

There already exists quite a large body of development best-practice and descriptions of common pitfalls so, rather than re-inventing the wheel, we would refer the reader to the following as examples: *The OWASP Guide to Building Secure Web Applications* (84), *Learn to Work Safely with Web 2.0* (85), and *Security Survival Tips for the Web 2.0 World* (86). We do, however, recommend some more general initiatives for mitigating risks arising from poor development:

- Develop and encourage secure development processes for Web 2.0: as in all areas of software development, recent experience has shown that code security can only improve by using a comprehensive approach throughout the development process, eg, by training in security-by-design, threat modelling, code review and penetration testing.

- Provide built-in features for IDEs to facilitate secure development: development environments (eg, Google Mashup Editor, Eclipse, CodeGear IDE, Komodo IDE) used for creating Web 2.0 applications should incorporate defaults, code templates and hints which aid secure development. (See vulnerability [DEVELOPMENT PROCESS ISSUES].)

- Security features of APIs: security features (eg, access control features) and models should be built, by-design, into APIs (eg, jQuery, script.aculo.us, dojo) shipped for Web 2.0 applications. (See vulnerability [DEVELOPMENT PROCESS ISSUES].)

- Anonymous strong authentication: strong authentication is traditionally opposed to anonymity since it binds a particular identifier to a claim made about that identifier in a number of different contexts, thereby increasing the ability to trace the actions of a user. Also certificates used in authentication contexts typically contain a fixed set of fields (name, date of birth, ID number, etc), which are always disclosed in every authentication event, no matterwhat the actual needs of the service. There are however privacy-preserving authentication technologies such as Credentica (87)and Idemix (88) which offer both strong and privacy preserving authentication. Such technologies offer strong authentication in combination with unlinkability between transactions and selective attribute disclosure. Very few services actually use such technology however and considerable development effort, building on prototypes developed by projects such as PRIME (89), is needed in order to integrate them successfully into Web 2.0 applications. In particular, effort is needed to make them useable and intelligible to users. (See vulnerabilities [WEAK AUTHENTICATION, PRIVACY VIOLATIONS].)

6.2 Concluding Remarks

Overall, Web 2.0 is a very positive social and technological development which has created many opportunities both for businesses and for interaction among private individuals. It supports the open-source ethos encouraged by the European Union by providing access to knowledge regardless of location or socio-economic standing. It has provided greater efficiency through allowing browser access to application features previously confined to the desktop and a wealth of previously undreamt of possibilities for social interaction.

Its rapid development has however, led to many security threats and vulnerabilities which have not yet been addressed. By the adoption of a comprehensive set of measures aimed at addressing these vulnerabilities by governments, service-providers, standardisation bodies and developers, the threats presented by the current wave of Malware 2.0 and other Web 2.0 related problems can be significantly reduced in order to realise the full benefits of this new technology.

7 Commonly Used Terminology and Abbreviations

AJAX	Asynchronous JavaScript and XML (Web development paradigm)
API	Application Programming Interface (Code libraries)
Blog	Web log – website with regular entries of commentary, descriptions of events, etc
Bot	Software program which runs autonomously without human intervention; used as part of botnets
Botnet	A large number of compromised computers that are used to create and send spam or viruses or flood a network with messages as a denial of service attack
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart – test used to ensure that a response is not generated by a computer
Codec	Metadata specifying how to encode and/or decode a digital data stream such as video or audio
Common carrier	Term describing the legal status of an IT service provider which provides only data transport services without intervening in content delivered (what exactly constitutes intervention is often a matter of debate)
CSRF	Cross-site request forgery
CVE	Common vulnerabilities exposure – database of public information security vulnerabilities
DNS	Domain Name System – specifies bindings between domain or host names and IP addresses
DOM	Document Object Model
Drive-by	Of malware infections – requiring no human intervention or knowledge
ECMAScript	A scripting language, standardised by Ecma International in the **ECMA-262** specification – most commonly used in web applications
EULA	End-user licence agreement
FOAF	Friend of a Friend – a machine-readable ontology describing persons, their activities and their relations to other people and objects; FOAF allows groups of people to describe social networks without the need for a centralised database
Folksonomy	Collaborative creation and management of tags to annotate and categorise content
GET	Type of Http Request
Host	In the context of this paper, the part of a URI preceding the domain name, eg, http://host.example.com

HTTP	Hypertext transfer protocol
IDE	Integrated Development Environment – software used to aid development
IFrame	HTML element which allows the embedding of one HTML document inside another HTML document
IRC	Internet relay chat
ISP	Internet service provider
JSON	JavaScript Object Notation – lightweight data interchange format for Javascript
Malware	Malicious software installed on a computer without the owner's consent
Mashup	A web application that combines data and services from more than one source
Mere conduit	Term describing the legal status of an IT service provider which provides only data transport services without intervening in content delivered (what exactly constitutes intervention is often a matter of debate)
Micro-blogging	Form of blogging using brief text updates or media such as photos or audio clips
PKI	Public Key Infrastructure
Plug-in	Browser add-on
POST	Type of HTTP request used for submitting forms
QoS	Quality of service
RDF	Resource Description Framework – metadata and semantic web format
RFID	Radio frequency identifier – wireless identification tags
RSS	Really Simple Syndication – syndication format
Sandboxing	Removal of any control functions from a system component so that the rest of the system cannot be accessed and is therefore not vulnerable to attack from that component
SME	Small to medium-size enterprise
SN	Social network
SOP	Same-origin policy
SSL	Secure Sockets Layer – server authentication and encryption protocol
SWF	Acronym used for Flash file forma – acronym of 'Shockwave Flash'
Syndication	Publication of material to multiple other content consumers – commonly used in conjunction with web feeds using RSS providing a summary of recently added content

Tag cloud	A visual depiction of user-generated tags used to describe the content of websites. The importance of a tag is shown with font size or colour.
TLS	Transport Layer Security – server authentication and encryption protocol – successor of SSL
TOU	Terms of use
Widget	Portable application that can be installed and executed within any separate web page – widgets often take the form of on-screen tools (clocks, stock market tickers, flight arrival information, daily weather, etc)
Wiki	Page or collection of web pages designed to enable anyone who accesses it to contribute or modify content, using a simplified mark-up language
xFolk	Open format for publishing collections of bookmarks
XHR	XML Http Request – scripting function used to transfer data between a web server and a browser asynchronously to the original http request it is returned in
XML	Extensible Mark-up Language (W3C Specification)
XSS	Cross-site scripting (attack)

8 References and Links

All web resources verified in November 2008.

1. ENISA Survey of end-user attitudes to security-related issues among Web 2.0 users. http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_survey_web2.pdf
2. Scansafe Security Brief. http://www.scansafe.com/threat_center/threat_alerts/stat_security_brief.
3. http://www.sophos.com/security/top-10/.
4. http://www.usenix.org/event/hotbots07/tech/full_papers/provos/provos.pdf.
5. IFrameDollars.biz. http://www.informationweek.com/news/security/vulnerabilities/showArticle.jhtml?articleID=163700819.
6. Chicago Crime Mashup. http://chicago.everyblock.com/crime/.
7. Mint – internet banking aggregation. http://www.mint.com/.
8. Google Apps. http://www.google.com/a/help/intl/en/index.html.
9. Picasa Refresh Facial Recognition Software. http://www.techcrunch.com/2008/09/02/picasa-refresh-brings-facial-recognition/.
10. **ENISA.** Security Issues and Recommendations for Online Social Networks. 2007. http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_social_networks.pdf.
11. Amazon Mechanical Turk Beta. http://www.mturk.com/.
12. Google's CAPTCHA busted in recent spammer tactics. http://securitylabs.websense.com/content/Blogs/2919.aspx.
13. Spammers Using Porn to Break Captchas. http://www.schneier.com/blog/archives/2007/11/spammers_using.html.
14. **Keizer, G.** New Bot-Powered eBay Scam Uncovered. http://www.techweb.com/wire/security/showArticle.jhtml?articleID=191600603.
15. Twitter refuses to uphold terms of service. 5 2008. http://arielwaldman.com/2008/05/22/twitter-refuses-to-uphold-terms-of-service/.
16. **Roessler, Thomas.** XDR API Security Impact, W3C mailing list post. http://lists.w3.org/Archives/Public/public-appformats/2008Apr/0068.html.
17. Gmail CSRF exploit. http://www.gnucitizen.org/blog/google-gmail-e-mail-hijack-technique/.
18. The Confused Deputy. Hardy, Operating Systems Review, Vol 4, 22, 1988.
19. **Shiflett, Chris.** The Dangers of Cross-Domain Ajax with Flash. http://shiflett.org/blog/2006/sep/the-dangers-of-cross-domain-ajax-with-flash
20. Phorm – online marketing tool. http://www.phorm.com/.
21. **Mahemoff, M.** Cross-Domain Communication with IFrames. 2008. http://softwareas.com/cross-domain-communication-with-iframes.
22. **(CVE), Common Vulnerabilities and Exposures.** CVE-2008-1447 DNS Insufficient Socket Entropy Vulnerability or the Kaminsky bug. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1447.
23. **Chris Karlof, J.D. Tygar, David Wagner, Umesh Shankar**. Dynamic Pharming Attacks and Locked Same-origin. 2007. http://www.cs.berkeley.edu/~daw/papers/pharming-ccs07.pdf.

24. **Collin Jackson, Adam Barth.** Beware of Finer-Grained Origins. Web 2.0 Security and Privacy (W2SP 2008). 2008. http://crypto.stanford.edu/websec/origins/fgo.pdf.

25. Safer Internet, Protecting Our Children on the Internet using Content Filtering and Parental Control Techniques. 2007. Page 8. The filtering solutions we benchmarked cannot sufficiently distinguish innocent from harmful content in these highly dynamic environments. http://www.sip-bench.eu/Reports2007/SIP%20Bench%202007%20-%20Synthesis%20Report.pdf.

26. Widgets – Web Vulnerabilities for All. http://www.w3.org/2008/Talks/0425-devtrack-tlr/slides.pdf.

27. Web Application Security Issues. http://www.w3.org/2008/Talks/0424-w3ctrack-tlr/0424-w3ctrack-tlr.pdf.

28. 24C3 lightning talk (video). http://youtube.com/watch?v=G7wcfu367T4.

29. XSS Attack information. http://www.xssed.com/.

30. 'Privacy Policy – Third Party Advertising' – applying to Facebook Beacon (HTML). Retrieved on 2007-12-04. http://www.facebook.com/policy.php.

31. Google web toolkit. http://code.google.com/webtoolkit/.

32. Direct Web Remoting – Client-Server AJAX Framework. http://directwebremoting.org/.

33. ASP.NET AJAX. http://www.asp.net/ajax/.

34. Social Investing and Pump-and-Dump 2.0. http://www.jacksonfound.com/2008/04/01/social-investing-and-pump-and-dump-20/.

35. Could Digg be used for Sun stock manipulation. http://siliconvalleysleuth.co.uk/2006/03/digg_is_used_fo.html.

36. Welcome to the era of gullibility 2.0 – stocks tumble on Engadget blog post. http://news.cnet.com/Welcome-to-the-era-of-gullibility-2.0/2100-1025_3-6185075.html.

37. Botnets – the siilent threat. http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_botnets.pdf.

38. Use of Web 2.0 technologies to control botnets. http://www.owasp.org/images/0/02/OWASP_Day_Belgium_2007-pdp.ppt.

39. With Web 2.0, a new breed of malware evolves. http://www.computerworld.com.au/index.php/id;850210034.

40. **ENISA.** Reputation-based systems:a security analysis. http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_pp_reputation_based_system.pdf.

41. Online Health Search 2006. http://www.pewinternet.org/pdfs/PIP_Online_Health_2006.pdf.

42. Wikiscanner. http://wikiscanner.virgil.gr/.

43. Web data and application security. http://www.cse.sc.edu/~farkas/csce813-2006/lectures/csce813-lect30.ppt.

44. **Jeremy J. Carroll,** HP Labs, Bristol. Signing RDF Graphs. 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-142.pdf.

45. Winamp metadata vulnerability. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-0065.

46. Microsoft Word's Hidden Tags Reveal Once-Anonymous Peer Reviewers. 2006. http://chronicle.com/free/v52/i33/33a04101.htm.

47. Psst! Your Metadata Is Showing! Dark Reading. http://www.darkreading.com/blog.asp?blog_sectionid=447&doc_id=145225.

48. Exposing Vulnerabilities in Web Software. Thiel, David. Amsterdam : s.n. BlackHat Europe 2008. http://www.blackhat.com/presentations/bh-europe-08/Thiel/Whitepaper/bh-eu-08-thiel-WP.pdf.

49. European convention on Human Rights (article 8 on privacy). http://conventions.coe.int/treaty/en/Treaties/Html/005.htm.

50. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm.

51. **Article 29 Working Party.** Opinion 4/2007 on the concept of personal data. 06 2007. http://ec.europa.eu/justice_home/fsj/privacy/docs/wpdocs/2007/wp136_en.pdf.

52. **ENISA**. The new users' guide: How to raise information security awareness. http://www.enisa.europa.eu/doc/pdf/deliverables/new_ar_users_guide.pdf.

53. *The Emperor's New Security Indicators.* **Stuart E. Schechter, Rachna Dhamija, Andy Ozment, Ian Fischer.** 2007. IEEE Symposium on Security and Privacy.

54. Yahoo's CAPTCHA Security Reportedly Broken. http://www.darkreading.com/document.asp?doc_id=143620.

55. Spammers crack Gmail Captcha. http://www.theregister.co.uk/2008/02/25/gmail_captcha_crack/.

56. Friendbot – automation tool for creating myspace friends. http://www.friendbot.com/.

57. **Roelof Temmingh, Chris Böhme.** investigating individuals and organisations using open source intelligence. http://www.blackhat.com/presentations/bh-europe-08/Temmingh-Bohme/Presentation/bh-eu-08-temmingh-bohme.pdf.

58. **Fried, Ina.** Warning sounded over 'flirting robots". CNET news. 2007. http://news.cnet.com/8301-13860_3-9831133-56.html.

59. Russian Computer Program fakes chat room flirting. ABC News. http://www.abc.net.au/news/stories/2007/12/13/2118477.htm.

60. Solving Captchas for Cash. http://ha.ckers.org/blog/20070427/solving-captchas-for-cash/.

61. **Sid Stamm, Zulfikar Ramzan, Markus Jakobsson.** Drive-by pharming report. Symantec. http://www.symantec.com/avcenter/reference/Driveby_Pharming.pdf.

62. Online port scanner – note this has not been tested by the group. https://www.grc.com/x/ne.dll?bh0bkyd2.

63. Symantec Threat Report, Volume XIII. 2007. http://www.symantec.com/business/theme.jsp?themeid=threatreport.

64. **Grossman, J.** Advanced Web Attack Techniques using GMail. 2006. http://jeremiahgrossman.blogspot.com/2006/01/advanced-web-attack-techniques-using.html.

65. CVE-2008-1318 JSON based attack on mediawiki. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-1318.

66. **Report, ENISA.** Information Security Certifications. http://www.enisa.europa.eu/doc/pdf/deliverables/inf_sec_certification_2008.pdf.

67. EU Stork Project: Large-scale eID pilots. http://www.eid-stork.eu/.

68. Explanation of Web of Trust. http://en.wikipedia.org/wiki/Web_of_trust.

69. http://news.cnet.com/8301-13578_3-10040152-38.html.

70. Caja – capabilities javascript. http://code.google.com/p/google-caja/.

71. The new users' guide: How to raise information security awareness. http://www.enisa.europa.eu/doc/pdf/deliverables/new_ar_users_guide.pdf.

72. Get Safe Online Guidelines. *Safe Social Networking*. http://www.getsafeonline.org/nqcontent.cfm?a_id=1459.

73. **Yahoo.** Yahoo Employee Blogging Guidelines. http://jeremy.zawodny.com/yahoo/yahoo-blog-guidelines.pdf.

74. Privacy and Identity Management for Europe, Tutorials. https://www.prime-project.eu/prime_products/reports/tuto/.

75. Access Control for Cross-Site Requests – W3C Specification. http://dev.w3.org/2006/waf/access-control/.

76. XACML – eXtensible Access Control Markup Language (OASIS). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

77. oAuth specification. 12 2007. http://oauth.net/core/1.0/.

78. Open ID. http://openid.net/.

79. Introducing Windows Cardspace. http://msdn.microsoft.com/en-us/library/aa480189.aspx.

80. Liberty Alliance. http://www.projectliberty.org/.

81. HTML 5, W3C Draft. http://dev.w3.org/html5/spec/Overview.html.

82. Web Security Context: User Interface Guidelines – W3C Working Draft. 07 2008. http://www.w3.org/TR/wsc-ui/.

83. The EU Provenance Project. http://www.gridprovenance.org/theproject.html.

84. The OWASP Guide to Building Secure Web Applications v2. http://www.owasp.org/index.php/Category:OWASP_Guide_Project.

85. Learn to Work Safely with Web 2.0. http://career-resources.dice.com/job-technology/learn_to_work_safely_with_web_2_0.shtml.

86. ComputerWorld: Security Survival Tips for the Web 2.0 World. http://www.computerworld.com/action/article.do?command=viewArticleTOC&articleId=285367&specialReportId=9000283&intsrc=article_sp_feat_bot.

87. Credentica website. http://www.credentica.com/technology.html.

88. Idemix website. http://www.zurich.ibm.com/security/idemix/.

89. EU Prime Project (Privacy and Identity Management for Europe). https://www.prime-project.eu/.


enisa
European Network
and Information
Security Agency

PO Box 1309, 71001 Heraklion, Greece, Tel: +30 2810 391 280
www.enisa.europa.eu