Future Internet **2012**, *4*, 700-718; doi:10.3390/fi4030700

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

Raising Risk Awareness on the Adoption of Web 2.0 Technologies in Decision Making Processes

Marco Prandini [1,*] and Marco Ramilli [2]

[1] DEIS, Università di Bologna, Viale del Risorgimento 2, Bologna 40136, Italy

[2] DEIS, Università di Bologna, Via Venezia 52, Cesena 47521, Italy; E-Mail:marco.ramilli@unibo.it

* Author to whom correspondence should be addressed; E-Mail: marco.prandini@unibo.it; Tel.: +39-051-209-3867; Fax: +39-051-209-3073.

Received: 27 April 2012; in revised form: 14 July 2012 / Accepted: 19 July 2012 / Published: 9 August 2012

Abstract: In the recent past, the so-called "Web 2.0" became a powerful tool for decision making processes. Politicians and managers, seeking to improve participation, embraced this technology as if it simply were a new, enhanced version of the World Wide Web, better suited to retrieve information, opinions and feedbacks from the general public on subjects like laws, acts and policies. This approach was often naive, neglecting the less-obvious aspects of the technology, and thus bringing on significant security problems. This paper shows how, in the end, the result could easily be the opposite of what was desired. Malicious attackers, in fact, could quite easily exploit the vulnerabilities in these systems to hijack the process and lead to wrong decisions, also causing the public to lose trust in the systems themselves.

Keywords: Web 2.0; e-government; security; decision making

1. Introduction

In 1999 Darcy DiNucci introduced the term Web 2.0 in her article entitled: "Fragmented Future" [1]. This seminal work described a new pervasive way of interaction between people through devices like TV, cell phones, and even home appliances. The article did not spur a significant discussion until five years later, when Tim O'Reilly stroke the term Web 2.0 during the O'Reilly Media Web 2.0 conference. Initially, the term referred neither to specific technical improvements nor to pervasive scenarios. It was meant as a way to describe new modes of interaction between people through the browser. Tim

O'Reilly's perspective was then refined in the "What Is Web 2.0" paper [2]. Later on, the term Web 2.0 was redefined by Bart Decrem [3] as the "Web-as-participation-platform" (or the "participatory Web") which can be seen as the ultimate improving of Tim Berners-Lee's "read/write Web" vision [4].

From the practical point of view, Web 2.0 represents the possibility for users to interact in a more natural and effective way. Thanks to the user-interface improvements and to the communication dynamism, Web 2.0 encourages the users to add value to the applications as they use it, to share what they learn, to see real-time changes and then to participate to collaborative groups.

For such reasons Web 2.0 finds a natural application in the electronic government (eGOV) environment [5–8]. The keys of government modernization (simplicity, user-friendliness, participation and inclusion, transparency, accountability, efficiency and innovation, *etc.*) can be implemented in a easy and fast way using such a technology.

Newer developments of the Web 2.0 potential, like Open Data, fit right into this approach: by standardizing the formats for representing information, and by licensing the data so that every citizen can use it, private and public institutions are fostering the spontaneous development of applications that are often surprisingly useful (See for example the success of Hackathlon initiatives). Open Data could represent the "killer app" leading to the long-since announced transition to the Web 3.0, or Semantic Web. The Semantic Web is characterized by having meta-data coupled to each piece of data, to give it a machine-processable meaning. The concept was introduced back in 2001 [9], and the academic community was prolific in defining the needed ontologies, description languages, and automated reasoning theories. Now, the sheer volume of data put in the public domain, and the widely distributed skills that the Web 2.0 has brought to scores of enthusiastic users, could really put them to use.

As Web 2.0 platforms gained popularity among users, decision makers started being more sensitive to the new opportunities for interaction with them. The technology behind brand-new applications such as social networks was enthusiastically applied also to a wealth of Web 1.0 applications (e.g., web interfaces to services regarding tax and revenue, motor vehicles, immigration), caught in the wave of modernization in the effort of making them fulfill the long-standing promise of bringing the public administration up-to-date with the digital age.

When compared to its ancestor, Web 2.0 exhibits an architecture that is potentially much more articulated. The many-browsers-to-one-server model is replaced by a many-to-many one. Aggregators, mash-up applications and similar content-gathering solutions can share between the browser and the web server(s) the role of building the page shown to the user, by contacting many sources of information. There are at least three main concerns emerging from this evolution: accessibility, reliability and security.

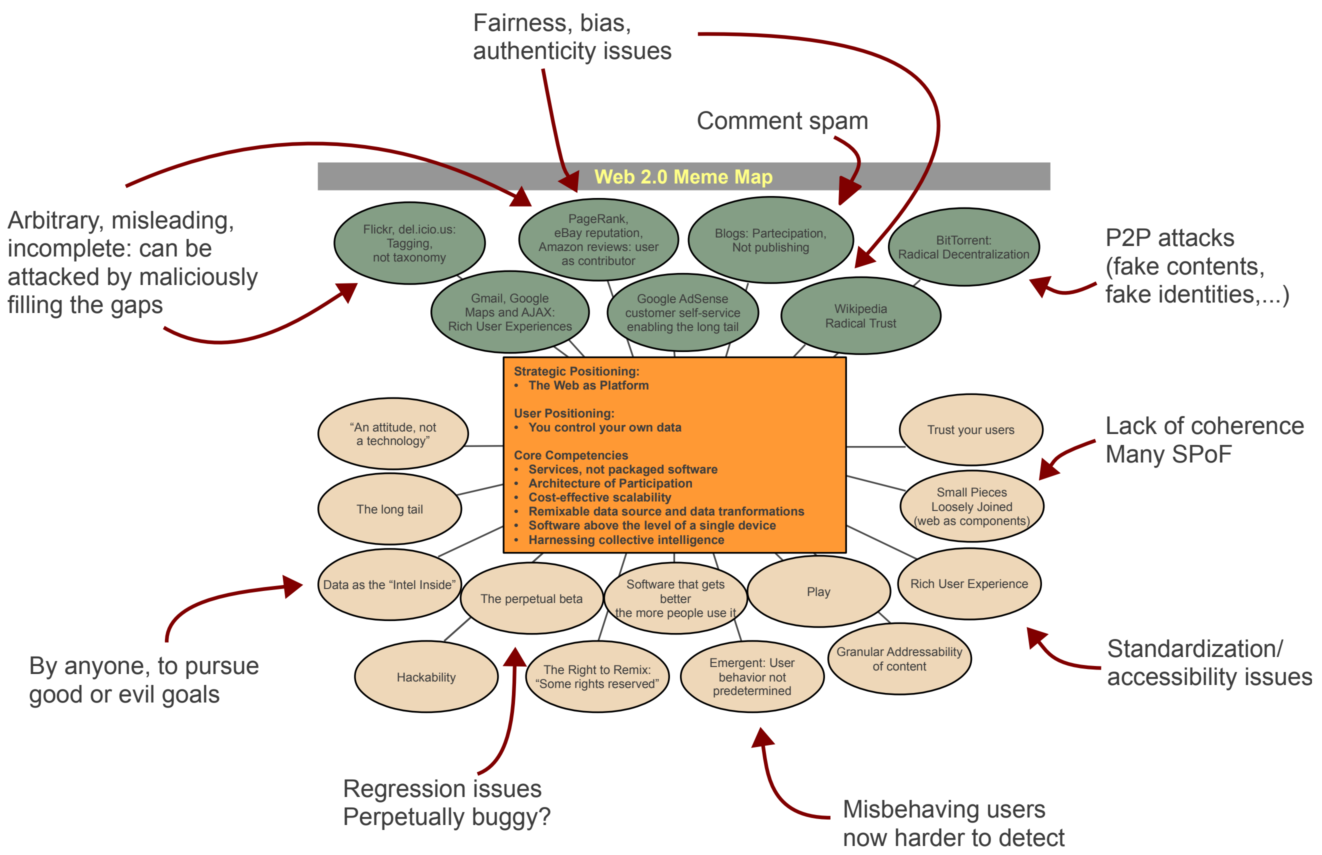
In the words of Cooper [10], "because Web 2.0 is an emergent phenomenon rather than a specific technology or set of technologies, there is no single designer or responsible authority that ensures that the technologies and practices support accessibility". Many of the consolidated technologies from the previous generation of the Web are used to transport pieces of information in a way that is often dictated by convenience, even if doing so means attributing new and ambiguous semantics to existing syntactical constructs. The solution Cooper suggests, namely adopting the Semantic Web approach to give unambiguous meaning to the "Microformats", still appears to be far from mainstream.

Reliability becomes an issue when complex services depend, possibly in a critical way, on a plurality of data sources and computation services. In a worst-case scenario, each of the components is a single point of failure. The unavailability of a piece of data or the impossibility to access a method of elaboration could jeopardize the whole application, even if all of the remaining components correctly work. Equally worrisome is the possibility of unexpected and undocumented modifications in the interfaces offered by one component. Other components could suddenly find it impossible to correctly communicate with the "updated" one, with effects ranging from visible crashes to subtle alterations of the displayed results. Literature exists [11,12] that addresses the issue of reliability, or robustness, of the kind of distributed systems powering Web 2.0 applications. The proposed solutions appear to be still at the academic level rather than widely deployed.

Finally, the Web 2.0 technology also introduces specific security issues, related to the much wider range of opportunities that are offered to attackers wishing to inject fabricated information into the data flow of an application. Inadequate knowledge of the possible vulnerabilities, in our opinion, would have particularly harmful consequences when the technology is exploited to drive decision- or policy-making processes.

We can show the aforementioned issues in the context of the original "meme map" (Figure 1) by Tim O'Reilly, synthesizing the cultural and technological elements that characterize Web 2.0. In our view, the innovations bring several vulnerabilities along with revolutionary possibilities.

Figure 1. The original "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software" (by Tim O'Reilly, © 2005 O'Reilly Media, Inc., [2]) surrounded by the vulnerabilities introduced by the new paradigm.


Fairness, bias, authenticity issues
Comment spam
Web 2.0 Meme Map
Arbitrary, misleading, incomplete: can be attacked by maliciously filling the gaps
Flickr, del.icio.us: Tagging, not taxonomy
PageRank, eBay reputation, Amazon reviews: user as contributor
Blogs: Partecipation, Not publishing
BitTorrent: Radical Decentralization
P2P attacks (fake contents, fake identities,...)
Gmail, Google Maps and AJAX: Rich User Experiences
Google AdSense customer self-service enabling the long tail
Wikipedia Radical Trust
Strategic Positioning:
• The Web as Platform
User Positioning:
• You control your own data
Core Competencies
• Services, not packaged software
• Architecture of Participation
• Cost-effective scalability
• Remixable data source and data tranformations
• Software above the level of a single device
• Harnessing collective intelligence
"An attitude, not a technology"
The long tail
Trust your users
Lack of coherence Many SPoF
Small Pieces Loosely Joined (web as components)
Data as the "Intel Inside"
The perpetual beta
Software that gets better the more people use it
Play
Rich User Experience
By anyone, to pursue good or evil goals
Hackability
The Right to Remix: "Some rights reserved"
Emergent: User behavior not predetermined
Granular Addressability of content
Standardization/ accessibility issues
Regression issues Perpetually buggy?
Misbehaving users now harder to detect

The goal of this paper is to describe these vulnerabilities in the mentioned context, and eventually to show why politicians, companies and agencies should take a more careful approach in embracing Web 2.0 technologies as a means of gathering feedback about their decisions, and even more when collecting inputs that can influence them.

The paper is structured as follows. We start providing the two necessary pieces of background. First, we introduce the decision making process at a very high level, in order to highlight some possible vulnerabilities allowing an attacker to influence the process outcome. Then, we describe the architectural and implementation aspects of Web 2.0 relevant to the field of decision making. On this foundations, we proceed highlighting the Web 2.0-specific attack vectors that can be used against the previously cited vulnerabilities, and how they substantially differ from those available in the Web 1.0 environment. The abstract considerations about these threats are exemplified in the last section before drawing conclusions.

2. Decision Making

2.1. Background

We make decisions everyday, from simple ones like where to have lunch, to critically important ones like choosing a home or a job. The decision making process brings the differences between the available options out, to sort them and find the best match with the decision maker's expectations; it is a cognitive process that is influenced by the environment as well as the intrinsic characteristics of the alternatives [13–15].

The literature of decision making was first organized by Ward Edwards in 1954 with his "The theory of decision making" [16], followed a few years later by his seminal work [17] starting the discipline of Behavioral Decision Theory. The bases for his work came from the disciplines of psychology, economics, and mathematics. The theories he considered and then the one he proposed assume rational behavior: individuals have transitive preferences, and choose among alternatives in order to maximize utility or expected utility. From then onwards especially in the 1970s, other fundamental contributions appeared to form the broader field of rational decision making, like the statistics-oriented Dempster–Shafer Theory [18], or the application of rational decision making to business organizations proposed by Herbert A. Simon [19]. Refer to "Judgment and decision-making theory" [20] for an overview of the theories and methodologies used to analyze judgment and decision-making processes.

When the decision involves a political aspect, effects unforeseen by "pure" rational theorists can arise. As Redlawsk [21] summarizes, unconscious biases can lead people in the process of making a choice to favor facts that support prior preferences, rather than to rationally update them. In this scenario, the term "motivated reasoning" is adopted [22–24] as a more appropriate description for a process that is also affective rather than purely rational.

2.2. A Vulnerability in the Process

To give a concrete example of the vulnerabilities that are the subject of this paper, the details of the different kinds of decision making processes are not relevant. Rational processes, either devoid of political significance or fully influenced by the affective biases that these introduce, share a very basic and common ground: they start from a set of possible choices and proceed to pick one (or more) of them as the result. We deem useful to model the key aspects of the process in a very simplified fashion as follows.

We define two sets:

- The initial set (IS) is the input of the process: the set of options initially available to the decision maker.
- The chosen set (CS) is the output of the process: the set of options representing what the decision maker has decided. It could be a stand choice or a set of choices.

The relationship in between IS and CS is the following one:

$$CS \leq IS$$

This relationship is always true in a correctly modeled scenario, since the goal of a decision making process is reducing the IS. Starting with a IS of cardinality greater than one, the two limit conditions where the outcome is either

$$IS \equiv CS$$

or

$$CS \equiv \emptyset$$

represent a failure of the process. In both scenarios the decision maker cannot reach a decision. The former scenario leaves the decision maker with the same set of choices present in the initial set. The process failed by being unable to reduce uncertainty between the available options, consequently not allowing to highlight the most desirable ones. The latter scenario represents a situation in which the decision maker is left with no viable options. Since some viable options, by definition, were present in the initial set, the process failed by ruling them all out, for example by applying exceedingly restrictive constraints.

Let us illustrate with an example how a malicious attacker can influence the decision making process to cause its failure, with a simple example that is not tied to any specific technology.

A man survives a flight crash and finds himself stranded on a uninhabited island. He needs to eat, and for that reason he needs to choose between hunting, fishing and collecting vegetables. A simple decision process can proceed by sequentially applying constraints that allow to prune the choice tree. Hard constraints are applied early: our subject is unable to build a fishing rod because he has no idea of how to realize the fishing hook, so fishing is ruled out. Personal preference and rational considerations follow: he does not like vegetables, and he knows that they do not provide enough calories to survive. Only hunting is left as an option, but our unlucky (and quite picky) man discovers he does not bear the sight of blood. He ends up with no choice.

The decision process failed, and missed the goal of indicating a way to satisfy the man's need for food.

Could this failure be the outcome of a malicious attempt at thwarting the process? The answer is clearly positive. An attacker might have added a spurious constraint such as: "vegetables do not provide enough calories for survival", that reinforce the decision maker's natural inclination against the vegetable-gathering option and cause him to exclude it. An attacker might have removed other suitable options from IS, like "fishing with a net" as a sub-alternative to "fishing with rod and hook". In these examples the attacker aims at driving the decision process towards the failure mode represented by $CS \equiv 0$. A subtler way the attacker could trick the decision maker into failure is by injecting unrealistic but favorable options into the IS such as: "you need to decide between hunting, fishing, collecting vegetables and going to the grocery store". Of course the decision making process will pursue the maximum benefit with the minimum associated inconvenience, and consequently even slightly more favorable and/or less constrained options have good chances to be selected. Unfortunately, they may prove impossible to turn into reality, possibly when it is too late or extremely inconvenient to reconsider the decision.

In summary, the vulnerability stems very clearly from the opportunity to interfere with the set of facts and constraints on which the decision-making process is based.

2.3. Public Administration and Decision Making in the Web 2.0 Era

Politicians, policy-makers, and administrators often wanted to collect the people's voice, in the keen interest of making better decisions or rather to share the responsibility for the consequences. Before information and communication technologies were involved, face-to-face meetings were the preferred means of discussing issues and sharing views. The decision-making process then proceeded rather straightforwardly, with the appointed public servants taking into account the collected ideas to a certain extent, and finally converting every contribution into a policy or plan.

Nowadays, Web 2.0 technologies and platforms are gaining popularity to gather decision-making inputs, to listen to people feedbacks, comments and feelings. For example U.S. president Barack Obama said (http://www.cbsnews.com/8301-503543_162-5666788-503543.html), speaking about Twitter:

"It makes me a better leader because it forces me to hear opinions that I don't want to hear."

In the Web 2.0 era, public input can reach the decision-makers through both unstructured channels (like Twitter in the previous example) and structured platforms (see for example, among many works, an analysis [25] of European eParticipation initiatives by Panopoulou *et al.*, or a survey [26] of participatory budgeting (PB) experiences in Central and South America by Cabannes–the survey itself predates Web 2.0 but many PB projects have been subsequently updated to exploit the new technology).

This possibility for wide inclusion of stakeholders does not change the model of the decision-making process. In the end, there is a single person or a small group who takes on the responsibility for the final choice: a politician or other appointed public servant. What makes a difference is the increase in the amount of opinions, suggestions, complaints, requests, *etc.* gathered, and the corresponding reduction in the actual possibility for verifying the authenticity of them all.

It is not widely understood that these technologies and platforms can facilitate an attacker wishing to inject malicious data, with the aim of forcing preconditions (such as "the vegetables do not provide enough energy for survive" in the preceding example), providing attractive choices (such as the "grocery store"), and eventually influencing the decision maker. The following chapters will clarify this claim by illustrating the peculiar aspects of Web 2.0 technologies that make these attacks possible, highlighting the increased risks with respect to the Web 1.0, and providing an example of a realistic attack.

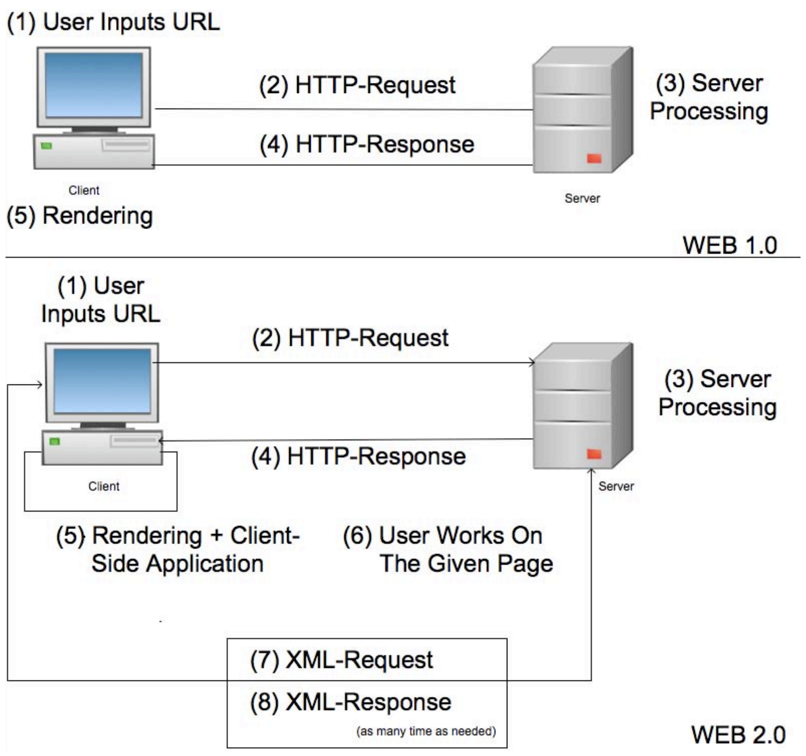
3. Web 2.0 Structure

Web 2.0 is a general name adopted to emphasize a new way to collaborate through the Internet. From a practical point of view, as it often happens, it is just a sort of "collective" name to keep together a number of different technologies, the most important ones being Ajax [27], Adobe® Flash® [28], Adobe® Flex® [29] and JavaScript [30]. These were already singularly available during the past years, but their coordinated usage allows to achieve the functionalities that represent the innovation over the old system, thereinafter known as Web 1.0.

The fundamental difference between Web 1.0 and Web 2.0 is caused by the shift of computational capabilities from the server to the client side. Web 1.0 interfaces had to delegate the processing of almost every user action to programs executed on the server, thus severely limiting interactivity. Web 2.0 interfaces, instead, are based on technologies that allow to perform significant computational activities on the client, greatly enhancing the user experience. The active components of a Web 2.0 page can render the consequence of some user action (e.g., a mouse click on a multi-level menu) by redrawing the involved area only, thus saving the client most of the rendering computational load. The needed contents can be pre-loaded (if they are predictable, as it is the case for the menu example), or dynamically gathered from a server, but without the need for transferring the unchanged parts of the page, thus saving most of the network-related lag.

Figure 2 illustrates these mechanisms. Let us suppose that some users want to interact with each other by exchanging information through a server. The upper section shows a Web 1.0 interaction: the browser of each user performs an HTTP request to fetch a page, the server provides it with an HTTP response, and the browser renders it on the screen. With a limited ability to perform computations on the user input and, more importantly, with no possibility to modify the page to reflect the results of the computation, the browser must repeat the whole process every time the user initiates an interaction. New information provided by other users is available only upon explicitly checking for updates, a process whose frequency is limited by the slowness of the request/response/render cycle. The lower section shows the same interaction, implemented with Web 2.0 tools. In this case, after loading the first complete page by means of a standard HTTP transaction, the browser does not leave it any more. In response to user input, the browser computes a different kind of request (XMLHTTP [31,32]) and sends it to the server. The server computes the data needed to satisfy the request and, besides replying, it can notify other users participating in the same interacting session. The browser then selectively updates the rendered page to show the new contents. The main idea behind this technology is to delegate the control flow to the browsers. Since performance and computational abilities are not an issue anymore, this new paradigm exploits the client's power, implementing a distributed calculator that amplifies the whole set of possible actions between users.

Figure 2. WEB 1.0 *versus* Web 2.0, one way communication paradigm.



The efficiency of the illustrated process allows to consider the browsers as logically connected with each other through the server (In some services the connection between clients is real and not logical as shown in Figure 2: the clients leverage the same technologies to exchange data without passing through a server). Moreover, each client has a high degree of control over the data, both when it is collecting it from the user (for example the browser can validate input parameters, showing graphical cues in case of errors or dynamically adapting the layout of an input form to the collected data), with or without interaction with the server (This is the classical example of RSS news where the client does not interact with the server side beside the first request), and when it is displaying it to the user (for example by notifying the user that some data is getting stale, and waiting for the user to decide how to react).

Another important property of the Web 2.0 is the link dynamism also called “permalink” (permanent link)[33,34]. Due to the enormous data traffic between clients (thanks to the Web 2.0 participation flow) and the substitution of dynamic databases for static files behind the service, the ability to reach each piece of information became an important issue. Permalinks are dynamically generated links which point out to specific data (or specific information) and not to a static page which will change during time.

4. Security Issues

Interaction comes with complexity. As Figure 2 may suggest, the second generation of Web is far more complicated than the first generation. Currently there are many more applications running on the world wide web, supported by browser plug-ins and mashing-up the highly varied contents provided by web sites. Complexity is always the best hiding place for bugs and vulnerabilities which might exploit the system in an unpredictable way. In this section we want to discuss about the new vulnerability frontiers that Web 2.0 introduced, providing basic and non-technical knowledge to everybody who uses it as a source of information for a decision making process. The following paragraphs describe the difference between the attack sets of the old and the new Web, quickly categorized in Table 1. For a glossary of the various vulnerabilities and threats we suggest to take OWASP [35] as a reference and starting point.

Table 1. A summary of Web 1.0 and Web 2.0 vulnerabilities.

Security property	**Violation**	**Web 1.0 attacks**	**Additional Web 2.0 attacks**
Availability	Denying service	DoS against the server	DoS against one server of the many composing the application
Confidentiality	Stealing data	Exploit server vulnerability (code injection, SQL injection, Cross-site scripting)	Exploit client vulnerability (hijack browsing by injecting malware on client-based execution environments)
Integrity	Altering data	Defacement by server vulnerability exploitation	Request forgeries and session hijacking
			Client-to-client propagation of malware

4.1. WEB 1.0 Attacks

Since there were no actual capabilities for interaction between clients, the main actor of this era was the web server, which was consequently the target of most attacks. For sure the most widespread attack of this era was the *denial of service* (DoS), where the attacker using botnets [36] or protocol vulnerabilities [37,38] was able to divert traffic or to saturate the server resources.

Saturating the server resources is commonly achieved by either consuming all of the available network resources or by forcing the allocation of all of the available memory. The first technique is, in turn, implemented in at least two variants. In the first one, the attacker can leverage traffic amplification effects: small requests, which do not saturate the attacker's bandwidth, are crafted to cause larger replies that saturate the victim's bandwidth. Proxy systems placed for performance optimization can turn against their users and exacerbate the effect [39]. In the second variant, the attacker can flood the server with connection opening (TCP SYN) requests that are never followed by the packets needed to complete the protocol, leading it to use all of the possible concurrent connections (there is an upper bound which is OS-specific and cannot be overcome) until a timeout expires. During this period, the victim cannot accept any useful connection, and as soon as timeouts expire and some slots are freed, the attacker can quickly fill them up again. The memory saturation technique can leverage both application-level vulnerabilities and network-stack-level ones. In both cases, the attacker opens a connection, and obviously the server creates a process (or a connection handler) to serve the incoming request. The attacker crafts the input stream so that the created process or handler keeps growing instead of performing some computation, sending a reply, and terminating as it usually happens. Soon all the available memory is taken up by these rogue processes and none is left for the allocation of useful ones.

Also the *Defacement* attack was popular during the past decade. The attacker exploited web server vulnerabilities to obtain enough rights to change the web site home page. The goal of this attack was to discredit the image of a company or to deliver rogue contents by leveraging a popular site. The

Defacement attack became a *Massive Defacement* attack in the case where the web server rights were not properly setup. For example if the same web server was holding up multiple websites and it uses a single user/group to manage them, an attacker able to compromise one website was potentially able to compromise all of them. This vulnerability, often due to misconfiguration but sometimes due to outright lack of sufficiently expressive and granular access control mechanisms, implied that each website owner had to trust every single website owner on the same web server. Many past attacks have been performed by attackers who simply acquired a legal space into the desired web server. Once the attacker owned a spot into the attacked web server he was obviously able to attack his own web application and perform a Massive Defacement attack on the entire web server exploiting the misconfiguration of access rights.

Step by step, web applications became more sophisticated by starting to use login credentials and becoming able to hold up images, documents, and executables. This intermediate era opened up doors to *Authentication*, *Authorization*, *Upload*, *Code Injections*, and *Database* attacks. Those attacks addressed the web applications' vulnerability rather than web server's vulnerability, shifting a little bit the target of the attack.

The most widespread attacks targeted against web applications all stem from a single kind of vulnerability: improper validation and sanitization of input collected through web forms before processing it. Examples include:

- SQL Injection [40], where the attacker is able to execute arbitrary queries on the target machine, by injecting them through web forms.
- File inclusion, where the attacker exploit forms to upload hidden command shells and backdoors onto websites to take full control of them.
- Cross-site scripting, where the attacker injects code through a form that is reflected back on the page by the server, and then executed with privileges associated to the vulnerable site (usually higher than the privileges associated to the attacker's site).

Since the web became more and more accessible, many programmers started their own websites. Due to a lack of communication many web programmers were not aware about application side vulnerabilities and built frameworks and web tools vulnerable to such attacks which spread bugs and vulnerabilities over years.

During those years numerous underground communities began their life by selling "0days" (*i.e.*, exploits for vulnerabilities for which no remedy was known), user data stolen by exploiting vulnerable web apps, and web server accesses. There are still many other attack techniques that through web applications want to address the hosting web servers (See for example [41]) rather than clients: all of them sit into the Web 1.0 attacks ecosystem.

4.2. Web 2.0 Attacks

As already discussed in previous sections the introduction of Web 2.0 brought a new communication layer: the "client-to-client" communication, meaning that a client interacts with other clients in an autonomous and automatic way. A user updating his pages with new contents, such as links, comments, pictures and videos, triggers the ability to see changes in real time on every connected client. The publishing process passes through the web server usually without its intervention, ending on the listening client platforms. For example let us consider the FaceBook platform. When a user uploads his status, each connected friend sees what the user has just uploaded. The uploaded content is stored on the FaceBook web servers for immediate as well as delayed delivery. For the sake of efficiency, the servers perform little or no inspection of the content. Thus, the servers can be healthy carriers for attacks.

The more complex infrastructure exposes a larger area on which to perform attacks. In a sense, this observation is not so obvious; in fact day after day technology is improving the security of different engineering disciplines such as home building (for example against earthquakes), car design (for example introducing the airbag technology), electronic system protection (for example introducing new isolation systems), society organization (for example adopting new surveillance systems), *etc*., but the security of the Web, which is often used as a building block for these and many other applications, is still more an art than an exact science.

In short, Web 2.0 attacks are the same as Web 1.0 attacks plus client side attacks. Of course all the previous-era attacks stay unchanged in the present, but now we have new entries such as: *browsers and plugin* attacks, *linking* attacks [42,43], *infrastructure* attacks such as browser buffer overflows and middleware (SOAP) protocol attacks [44,45], and finally *application* attacks. Those attacks aims to exploit vulnerabilities in the corresponding components of the "enriched browser" by injecting malicious code through the visualized pages or by spreading out malicious versions of the same components. The attacker can exploit rich execution environments such as JavaScript, CSS or general HTML Objects (Flex, Flash, Microsoft SilverLight) both to act locally on the victim system and to propagate his malware as a worm, potentially to each client which comes into contact with an infected one. The malicious injected code could use attacks such as: Cross-Site Request Forgery (CSRF), Broken Session Management, and Incorrect Authentication Management to steal personal information (credit card numbers, website credentials and so on) or to force specific browsing behaviors. One of the most dangerous attack performed by malicious code is the ability to force a client behavior. Through CSRF the attacker includes a link or a script in a page that accesses to a specific web site in which the attacked user is known (or is supposed) to have been authenticated. The malicious code uses the authentication token belonging to the attacked client to perform specific actions. For example the attacker could buy items on a e-commerce site, or bid an auction or read sensitive data from a restricted area.

Web servers, web browsers, and network infrastructure are nowadays commodities that from the viewpoint of the end user are blended into an indistinct middleware, enabling the delivery of distributed, social applications. Processes and software products which were designed for the traditional client-server model (or even for off-line usage!) have been quickly adapted to become Web 2.0 services without much thought for the much wider spectrum of possible threats against their vulnerabilities, both intrinsic and newly introduced along with the added technological layer.

5. Attack Example

In order to get a better grasp of the possible consequences of the mentioned security problems, this section provides an example on how to exploit a Web 2.0 infrastructure to compromise a decision making process.

Even in its simplicity, this example lends itself to show different opportunities for an attacker to interfere with the process. It is applicable in the case of a political actor wishing to follow the "Obama's suggestion" regarding the social media, and deciding to use a Web 2.0 platform (WP) as an input of his decision process. It is applicable as well in a voting-like scenario where the public is actually the decision maker, basing its choice upon pieces of information and interactive discussions taking place on social media.

5.1. Attack Scenario

In a critical web service with low load, the correct way to handle interaction and data gathering would be through a *double input filtering* scheme. As the word "double" suggests, input filtering happens into two different stages in the system. The first stage operates on the client and is often implemented as obfuscated Javascript libraries, in order to make them harder to manipulate for an attacker, but there are no really effective techniques to protect their integrity. Those libraries thus usually control the compliance of user-provided data to basic format assumptions, mainly for the sake of quickness in reporting possible errors. For example let us assume we need to read an email field. Each email field is formed as following: <string> @ <string> . <com, org, info, gov, edu, it ... ... >. The client-side stage verifies the congruence of what the user types with such a pattern, so for instance if the user inserts *xxxxxxx* it will be flagged as an incorrect input value without involving server-side processing. The second stage operates on the server side. On Web 1.0 platforms, it performed a wealth of checks that were feasible only with the inclusion of server-stored information. Today, most of its work would be repeating the checks already performed by the first stage, with the addition of some binary correctness checks regarding, for example: encoding, boundaries, signedness, *etc*.

Since double input filtering results too resource-consuming for an high-traffic participation platform, an overly enthusiastic Web 2.0 approach suggests to simply eliminate the second stage, strengthening the client-side checking procedures, to distribute the computation among end-users.

In the end, this creates a commonplace scenario, in which the sacrificed redundancy paves the way for attacks.

Let us assume the WP is structured as a central service, as many current services, to delivery short messages, images and multimedia contents among subscribers in the following way:

- A user writes messages and his followers read the message.
- WP uses client input filter libraries written in a client side language such as Ajax of Flash.
- WP trusts the input since it has already been checked and filtered from client input filter libraries.

The attack on this kind of system works as follows, as illustrated in Figure 3.

Step (1) represents the classic request phase from user's client to service's server.

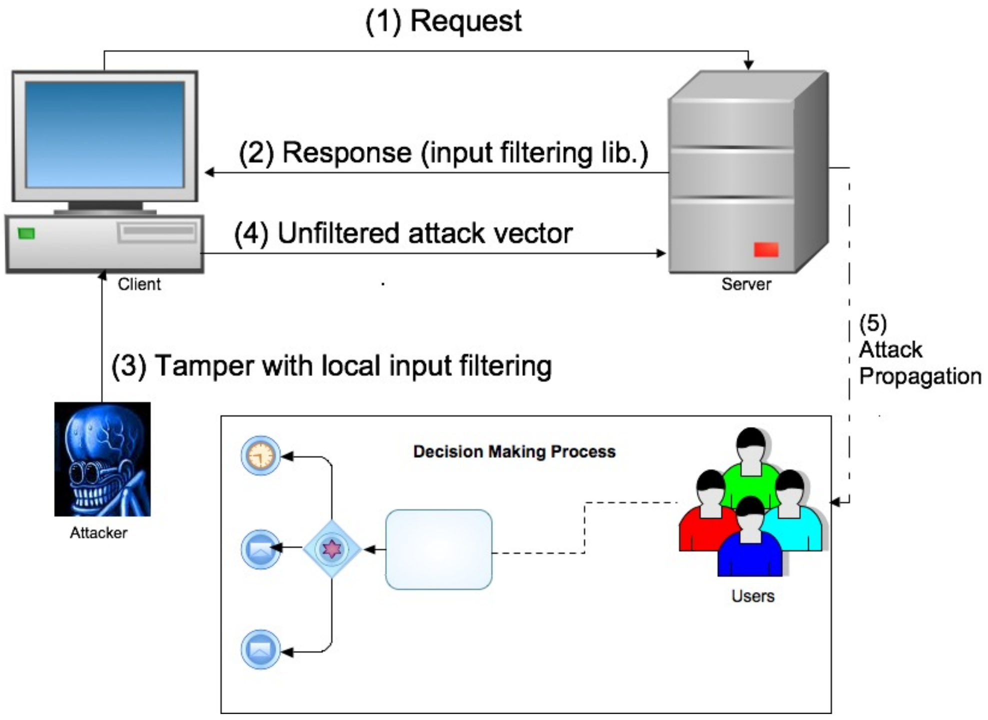
Step (2) describes the server's response within the embedded input filtering procedure.

Step (3) is the core of the attack. The attacker, manipulates the local input control by corrupting local memory or even simply by forging a local HTML page with absolute and remote URIs but without control procedures. The corrupted client interface sends back to the server the attack vector.

Step (4) Since the server does not perform double input control, it trusts the stream coming from the client and records it in its database.

Step (5) By querying the service, unaware users receive in their web browser the attack vector forged by the attacker. The attack vector is a now a legitimate browser-based application.

Figure 3. Example of Web 2.0 Attack on decision making process structure.


(1) Request
(2) Response (input filtering lib.)
(4) Unfiltered attack vector
Client
Server
(5) Attack Propagation
(3) Tamper with local input filtering
Attacker
Decision Making Process
Users

The decision maker cannot tell the altered application from the correct version, and thus bases the process on inputs that the attacker can arbitrarily set.

In terms of the illustrated decision making process, the attacker can operate on the IS by injecting new contributions which appear to come from a variety of legitimate users, as well as preventing the propagation of legitimate contributions. By doing the latter operation in a targeted fashion (for example by preventing the decision maker alone from seeing some specific pieces of information) the attacker keeps specific control over the perceived IS. By doing the same in a more manifest way (for example by simulating frequent failures during the comment publication attempts) the attacker can disenfranchise participants and lead them to abandon the process altogether.

Basically, the most important thing to notice here is: it is not necessary anymore to attack a professionally-managed web server to deliver malicious contents to clients, a flaw in one client can open the door to many others. The more hype is placed in shifting control to the user's side, for the sake of building attractive applications, the higher the risks for an attack to succeed (possibly on a large scale).

The following section illustrates a specific example showing how the Web 2.0 characteristic of relying on rich client-side-executed applications gives an attacker the opportunity to bypass centralized checks—typical of the previous generation—and gain access to a variety of possible malicious actions, ranging from the hijacking of the intended navigation path to the injection of doctored contents and the execution of de-facto malware.

5.2. Attack Vector Example

In this section we illustrate a brief description of the MySpace QT attack, a simple example of attack vector as introduced in Section 5.1.

MySpace was one of the first Web 2.0 online services to introduce the possibility for users to embed movies, audio, pictures and comments on their own profile. MySpace users can watch other users' profiles and interact with them. One of the most appreciated MySpace features is the capability to upload movies; in fact many movie stars and music stars are using their MySpace profile to show up new backstage videos, inedited live acts and unpublished pictures. Garage bands and music lovers are also using MySpace as a launching platform where they can show to the world what they do, for free on both sides, gaining popularity and receiving feedbacks from people.

Apple's QuickTime [46] is one of the most used tools to play multimedia contents such as movies, picture galleries and audio files, by MySpace users as well as all over the Internet. The QuickTime movie file structure incorporates a feature called HREF tracks which allows users to embed URLs into an interactive movie. This feature is widely used to dynamically link contents into advertisements or political races. For example a soda brand can show a movie trailer featuring its product. On the bottom of the screen or directly on the bottle an ad with a "buy me now" message can be displayed. The user can click on it and be redirected directly on the company e-Commerce page.

The attacker can exploit this feature by adding malicious JavaScript code through the HREF tag, so when the movie is played by unaware users the evil code is executed. The code below represents a working implementation of this kind of attack: QTWorm.

```
 1  javascript:
 2
 3  void((
 4   function() {
 5
 6     var e=window.document.createElement('script');
 7
 8     var ll=new Array();
 9     ll[0]='http://www.daviddraftsystem.com/images/';
10     ll[1]='http://www.tm-group.co.uk/images/';
11
12
13     var lll=ll[Math.floor(2*(Math.random()\%1))];
14
15     e.setAttribute('src',lll+'js.js');
16
17     window.document.body.appendChild(e);
18  })
```

Line 6 contains the critical vulnerability. QuickTime filters potentially dangerous HTML elements like `script`, `img`, *etc.* but lets the programmer create whole new elements through the API `createElement`. The worm uses this API to create a new `script` element, circumventing QuickTime filters. Each script element has got the ability to perform actions, even system actions like binding a new listening port or connecting to a remote address. It is a kind of self standing program inside the web browser. From line 8 through line 10 the worm initializes an array containing "n" (in this case 2) malware repositories from which it will download the malicious code and execute it. Line 13 implements the oracle, which decides from which site to load the malicious code. Finally line 15 appends to the current page the freshly-downloaded script. The last instruction, on line 17, starts the execution of the malicious code by adding the script to the body of the page.

As a self standing program the script is able to show, to hide and generally speaking to modify the user interface behavior, altering the user's perspective. If the user, as discussed in Section 2, has to take decisions through this interactive channel, he might be influenced by the script. For example, assume a record producer is looking for new talents on MySpace. Once he finds some candidates, he wants to figure out how they work by clicking on their video trailers. If the video trailers are compromised by an attacker the producer might see fake videos, he might listen to counterfeited music or be redirected to unrelated (without he realizing it) sites containing questionable contents to alienate whichever favor he might have about the candidate.

This example explains the indirect and implicit channel between one client and another one (the attacker, the producer and the talents are clients of MySpace), and how easy it could be to propagate malware into a Web 2.0 environment.

The very same example applies to political contexts as well, in two different directions. When the people is the decision maker, for example in an election, compromised electoral videos can misinform, influence or confound voters. When a politician is the decision maker, for example offering multimedia description of possible projects to stimulate comments by the public, an infected social media platform can report him altered or fabricated comments, while suppressing the visualization of original ones.

6. Conclusions

The Web 2.0 provides countless opportunities for the improvement of communication between citizens and governments. Ultimately, the gathered opinions, feedbacks, suggestions and sentiments provide a significant input to steer the decision making processes. It is important to raise the proper awareness on the risks associated with this trend.

In this work, we discussed a way to subvert the correctness of a decision making process based on Web 2.0 tools. The success of Web 2.0 is the main enabling factor for this kind of attacks, because it offers the attackers thousands of end-user, not professionally guarded systems as entry points. We provided a general description of possible means to attack the abstract decision making process itself, and a working example of how to taint any Web 2.0 application that may be involved.

As it often happens in computing, technologies developed with user convenience in mind do not take security in proper consideration.

Examples of this approach cover the full spectrum of contexts, from operating systems targeted at home users [47] to critical processes like e-voting—both on the side of standard procedures [48] and on the side of deeply technical testing campaigns (See section 4.1.7 of [49]) [50].

Governments and public administrations can attain undeniable benefits from leveraging the vast potential of Web 2.0, but they must carefully deploy and monitor their applications to ascertain the integrity of the collected data.

References and Notes

1. DiNucci, D. Fragmented future. *Print* **1999**, *32*, 221.
2. O'Reilly, T. What Is Web 2.0. Available online: http://oreilly.com/web2/archive/what-is-web-20.html (accessed on 1 August 2012).
3. Decrem, B. Introducing Flock Beta 1. Available online: http://www.flock.com/node/4500 (accessed on 13 January 2007).
4. Lawson, M. (Interview with) Berners-Lee on the read/write web. Available online: http://www.readwriteweb.com/archives/interview_with_tim_berners-lee_part_1.php (accessed on 1 August 2012).
5. Drogkaris, P.; Gritzalis, S.; Lambrinoudakis, C. Transforming the Greek E-Government Environment Towards the E-Gov 2.0 Era. In *Proceedings of the First International Conference on Electronic Government and the Information Systems Perspective*, Bilbao, Spain, 31 August–2 September 2010; Springer-Verlag: Berlin, Germany, 2010; pp. 142–149.
6. Koffa, A.; Kastania, A.N. Web Applications and Public Diplomacy. In *Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Cardiff, UK, 8–10 September 2010; Springer-Verlag: Berlin, Germany, 2010; pp. 53–62.
7. Traunmüller, R. Web 2.0 Creates a New Government. In *Proceedings of the First International Conference on Electronic Government and the Information Systems Perspective*, Bilbao, Spain, 31 August–2 September 2010; Springer-Verlag: Berlin, Germany, 2010; pp. 77–83.
8. Ostergaard, S.D.; Hvass, M. eGovernment 2.0–How can Government benefit from web 2.0? Available online: https://blog.itu.dk/MEGV-E2011/files/2011/11/scijournalsdohvassv2.pdf (accessed on 1 August 2012).
9. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web: Scientific American. *Sci. Am.* **2001**, *284*, 29–37.
10. Cooper, M. Accessibility of Emerging Rich Web Technologies: Web 2.0 and the Semantic Web. In *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*; ACM: New York, NY, USA, 2007; pp. 93–98.
11. Laranjeiro, N.; Vieira, M.; Madeira, H. Improving Web Services Robustness. In *Proceedings of IEEE International Conference on Web Services*, Los Angeles, CA , USA, 6–10 July 2009; pp. 397–404.
12. Randles, M.; Lamb, D.; Odat, E.; Taleb-Bendiab, A. Distributed redundancy and robustness in complex systems. *J. Comput. Syst. Sci.* **2011**, *77*, 293–304.
13. Tse, E. Planning and Decision Making Processes. In *Proceedings of the American Control Conference*, Arlington, VA , USA, 14–16 June 1982; pp. 923–924.
14. Eisenhardt, K.M.; Zbaracki, M.J. Strategic decision making. *Strateg. Manag. J.* **1992**, *13*, 17–37.
15. Tsebelis, G. Decision making in political systems: Veto players in presidentialism, parliamentarism, multicameralism and multipartyism. *Br. J. Polit. Sci.* **1995**, *25*, 289–325.

16. Edwards, W. The theory of decision making. *Psychol. Bull.* **1954**, *51*, 380–417.
17. Edwards, W. Behavioral decision theory. *Annu. Rev. Psychol.* **1961**, *12*, 473–498.
18. Shafer, G. *A Mathematical Theory of Evidence*; Princeton University Press: Princeton, NJ, USA, 1976.
19. Simon, H.A. Rational decision making in business organizations. *Am. Econ. Rev.* **1979**, *69*, 493–513.
20. Stevenson, M.K.; Busemeyer, J.R.; Naylor, J.C. Judgment and Decision-Making Theory. In *Handbook of Industrial and Organizational Psychology*, 2nd ed.; Consulting Psychologists Press: Washington, DC, USA, 1990; Volume 1, pp. 283–374.
21. Redlawsk, D.P. Hot cognition or cool consideration? Testing the effects of motivated reasoning on political decision making. *J. Polit.* **2002**, *64*, 1021–1044.
22. Kunda, Z. Motivated inference: Self-serving generation and evaluation of evidence. *J. Personal. Soc. Psychol.* **1987**, *53*, 636–647.
23. Kunda, Z. The case for motivated political reasoning. *Psychol. Bull.* **1990**, *108*, 480–498.
24. Lodge, M.; Taber, C. Three Steps toward a Theory of Motivated Political Reasoning. In *Elements of Reason: Cognition, Choice, and the Bounds of Rationality*; Cambridge University Press: Cambridge, UK, 2000.
25. Panopoulou, E.; Tambouris, E.; Tarabanis, K. eParticipation initiatives: How is Europe progressing? *Eur. J. ePract.* **2009**, *7*, 15–26.
26. Cabannes, Y. Participatory budgeting: A significant contribution to participatory democracy. *Environ. Urbanization* **2004**, *16*, 27–46.
27. Hazaël-Massieux, D. JavaScript Web Apis. Available online: http://www.w3.org/standards/webdesign/script.html (accessed on 3 August 2012).
28. Adobe Systems Incorporated. Adobe Flash Professional CS6. Available online: http://www.adobe.com/products/flash.html (accessed on 3 August 2012).
29. Adobe Systems Incorporated. Flex. Available online: http://www.adobe.com/products/flex.html (accessed on 3 August 2012).
30. ECMA International. Standard ECMA-262 ECMAScript Language Specification. Available online: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf (accessed on 3 August 2012).
31. W3Schools. *Learn JavaScript and Ajax with w3Schools*; Wiley Publishing: Weinheim, Germany, 2010.
32. Lerner, R.M. At the forge: Beginning Ajax. *Linux J.* **2006**, *151*, 10.
33. Blood, R. How blogging software reshapes the online community. *Commun. ACM* **2004**, *47*, 53–55.
34. Tung, W.F. Analytical Trackback Interconnections for SNS-Based Blog Services. In *Proceedings of the 4th International Conference on New Trends in Information Science and Service Science*, Gyeongju, Korea, 11–13 May 2010; pp. 615–620.
35. OWASP Foundation. The Open Web Application Security Project. Available online: https://www.owasp.org/ (accessed on 3 August 2012).

36. Liu, J.; Xiao, Y.; Ghaboosi, K.; Deng, H.; Zhang, J. Botnet: Classification, attacks, detection, tracing, and preventive measures. *EURASIP J. Wirel. Commun. Netw.* **2009**, *2009*, doi:10.1155/2009/692654.
37. Ballani, H.; Francis, P.; Zhang, X. A study of prefix hijacking and interception in the internet. *SIGCOMM Comput. Commun. Rev.* **2007**, *37*, 265–276.
38. Ballani, H.; Francis, P.; Zhang, X. A Study of Prefix Hijacking and Interception in the Internet. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Kyoto, Japan, 27–31 August 2007; ACM: New York, NY, USA, 2007; pp. 265–276.
39. Triukose, S.; Al-Qudah, Z.; Rabinovich, M. Content Delivery Networks: Protection or Threat? In *Proceedings of ESORICS 2009 : 14th European Symposium on Research in Computer Security*, Saint-Malo, France, 21–25 September 2009; Backes, M., Ning, P., Eds.; Springer: Berlin, Germany, 2009; Volume 5789, pp. 371–389.
40. Halfond, W.G.; Viegas, J.; Orso, A. A Classification of SQL-Injection Attacks and Countermeasures. In *Proceedings of the IEEE International Symposium on Secure Software Engineering*, Arlington, VA, USA, 13–15 March 2006.
41. OWASP Foundation. Top 10 2007-Malicious File Execution. Available online: http://www.owasp.org/index.php/Top_10_2007-A3 (accessed on 3 August 2012).
42. Fogie, S.; Grossman, J.; Hansen, R.; Rager, A.; Petkov, P.D. *XSS Attacks: Cross Site Scripting Exploits and Defense*; Syngress Publishing: Rockland, MA, USA, 2007.
43. Lin, X.; Zavarsky, P.; Ruhl, R.; Lindskog, D. Threat Modeling for CSRF Attacks. In *Proceedings of the 2009 International Conference on Computational Science and Engineering*, Washington, DC, USA, 2009; Volume 3, pp. 486–491.
44. Seo, J.; Kim, H.S.; Cho, S.; Cha, S. Web Server Attack Categorization Based on Root Causes and Their Locations. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*; IEEE Computer Society: Washington, DC, USA, 2004; Volume 2, p. 90.
45. Chong, K.; Song, H.Y.; Noh, S.H. Traffic Characterization of the Web Server Attacks of Worm Viruses. In *Proceedings of the 2003 International Conference on Computational Science*; Springer-Verlag: Berlin, Germany, 2003; PartII, pp. 703–712.
46. Apple Inc. QuickTime. Available online: http://www.apple.com/quicktime/ (accessed on 3 August 2012).
47. Remember Microsoft Bob, an alternative approach to the desktop environment, that offered users to choose a new password if they appeared to have forgotten it.
48. Felten, E. "Hotel Minibar" Keys Open Diebold Voting Machines. Available online: https://www.freedom-to-tinker.com/blog/felten/hotel-minibar-keys-open-diebold-voting-machines/ (accessed on 3 August 2012).
49. Calandrino, J.A.; Feldman, J.A.; Halderman, J.A.; Wagner, D.; Yu, H.; Zeller, W.P. Source Code Review of the Diebold Voting System. Available online: http://www.sos.ca.gov/voting-systems/oversight/ttbr/diebold-source-public-jul29.pdf (accessed on 3 August 2012).

50. During informal discussion between the authors of this paper and a TTBR auditor it emerged that Diebold told them that most of the issues were already solved, but the certified version was held back 16 revisions behind the latest one for convenience reasons.