BONDI

# W3C SECURITY WORKSHOP POSITION PAPER: A WEB BASED SECURITY MODEL FIT FOR PURPOSE

Nick Allott, David Rogers, Geoff Preston OMTP Office

## 1 THE PROBLEM

More and more web frameworks are being extended with access to native device capability. This breaks the traditional sandboxed security model of the web, and puts web culture and native applications on a collision course, which could result in a best of both worlds - a rich interoperable platform for application development - or a worst of both worlds a ubiquitous platform for unrestricted viruses and malware propagation.

Good enough security is the key enabler that can bring this best of both worlds vision to fruition.

There are, however, significant challenges to the creation of such a security model

- It must be usable – the configuration and utilisation of the security model must be appropriate to the man in the street.
- It must be applicable to its environment (it should accomadate the differences between the mobile and the fixed domains)
- It must engender confidence – the security framework should instil trust not fear in user of the system
- It must reflect market need – the model must represent the needs of the whole value chain, and not explicitly prefer any actor in the value chain, or uptake will be inhibited
- It must reflect user need – users must want it or it will not be utilised
- It must be robust - it must genuinely stand up to current security threats and offer options for the unforeseen.
- It must be future proof – it must anticipate future security threats.

This is a considerable challenge, but we have the advantage of learning from the many successful and unsuccessful security models that have preceded us.

## 2 TYPICAL SECURITY MODELS AND THEIR WEAKNESSES

Most current application security models fall into one or more of the following classifications

- Domain based security models: most frequently used in browser based scenarios, an application is identified by the domain on which it is hosted and the browser maintains a list (possibly populated from user prompting) of which domain is granted rights to which actions. Examples: Google Gears and ActiveX based security.

- Certificate based security: an application is signed with a certificate by some authority. Such models may use either a perimeter or least privilege security model,or in fact a combination of the two.
    - Perimiter Security: The application is granted rights to install based on having been signed against a valid root. Once installed the application has reasonably unrestricted access to APIs.
    - Least Privilage: The application is granted rights to access different API subsets depending on the authority that signed the application or the level of permissioning provided within the certificate. The rights to API access may be further restricted based upon the applications authors explicit request of specific capability; the principle of least privilege.
- Prompting based security: when an application requests access to a certain capability the application framework will explicitly prompt the user for permission for access. This permission may be granted in perpetuity from that point or may be requested on each subsequent access.
- No security: some application platform have no inherent security, there are no explicit limits on access. In these scenarios, it is typical for the burden of security to be born by a 3rd party agents, such as a virus checker.

The following list covers some typical security weaknesses inherent in the above security models seen in the market.

## 2.1 PROMPTING

Many security models make heavy use of user prompts in their implementation. Sometimes these prompts are at run-time (e.g. the prompt that asks you if you want to use the internet every time you open a web browser) or at install time (e.g. the prompt that informs you that this application does not have a valid certificate and asks if you really want to install it). The rationale behind end user prompting is that it defers critical security decisions to the user, but it also acts as a disclaimer abdicating technology and service suppliers from responsibility; convenient, but perhaps not in the best interests of the user.

Specifically, we can levy four distinct criticisms against prompting as the basis of a security model

- Sufficiently informed: is the end user sufficiently informed (or bothered enough) to make complex security choices that impact security?
- Social engineering attacks – spoofing and habituation: prompting regimes, especially the more aggressive ones that present users with multiple dialogs, run the risk of habituating the user to respond in a particular way, meaning of course they are no longer paying attention and making the informed choice, which was the original intent. This can be an inadvertent result of a prompting implementation, or there are a number of examples where this phenomena has been deliberately exploited by malware.
- Trust: a security prompt is intended to engender trust – I will trust this application more because it has asked me for permission and I have given it. However, too often this approach has the opposite effect, explicitly asking the question can raise the end user's consciousness of an issue, which would otherwise not be a problem. Trust means not having to ask permission! We will have succeeded in creating a strong framework, with robust chains of trust, when we have no more prompting

- Usability: all people with an internet background are aware of the impact of bad user interaction model (or a bad usability experience), with unnecessary clicks impacting on user behaviour and adoption. A prompting regime introduces unnecessary clicks.

As part of OMTPs work in the security area, we previously commissioned,extensive independent, cross cultural end user studies on security and prompting, the simple conclusion was: avoid it. Interested parties can contact us to get more details.

## 2.2 DNS AND IDENTITY SPOOFING

One of the critical inputs to a security policy is the "identity" of the application which is requesting access to a sensitive resource. Many native mobile application security models base this on some form of certificate, which has proven to be a strong form of application identity. There a number of implementations of web based security that make a security decision based upon the IP domain of the hosted application; this is a weak form of application identity. Simply put: it is very easy to spoof DNS responses, both theoretically and proven by exploits in the wild. We cannot base a robust security model on a weak domain model.

## 2.3 TROJANS

What if an application pretends to be one thing, then further down the line reveals itself to be something else. Prompting methods, domain based security and certificate based security in their simplest form cannot adequately address this issue. If I gave an application right to access my messaging APIs at a point of install, (by prompt) and this application is later revealed to be malicious, how do I find out that this is the case, and how is the situation resolved. Kill switches, blacklists and certificate revocation lists can be used theoretically, but the technical, infrastructure and contractual issues involved have proven this to be non trivial in practice

## 2.4 COST

Many end to end security frameworks can be expensive to set up and run. Certificate based security models require highly scalable certificate authorities. Historically these costs have been passed down to developers in terms of high costs for certificate issue and/or testing. This can be a significant barrier to innovation in the industry.

## 2.5 EXPLOITS AND UPDATES

Even assuming a perfect theoretical design, a secure framework is only as good as its implementation. Such implementations are inevitably flawed and a cursory examination of the data will reveal that a large number of malicious code attacks are based on exploits of in imperfect implementations. A robust implementation of a secure framework must start with the assumption that its implementation will be flawed and implement the mechanisms to allow this to be corrected on an almost continuous basis.

## 2.6 LACK OF FLEXIBILITY AND FRAGMENTATION

Where market forces have encouraged companies to adopt different security policies and the underlying security mechanisms prove hard to update either pre or post point of sale, considerable fragmentation, can result – leading to developer and end user frustrations. Certificate based models best embody this issue. It is common in these scenarios to only grant access (change policy) by certifying the application by a privileged body. For this to work root certificates from this privileged body need to be installed on every handset to which this policy must apply. This means that to change a policy, not only do root certificates need to be changed on every phone, but signing infrastructure needs to be deployed in the outside world for every privileged body that wished to make policy determinations.

For widgets the last issue is perhaps the most serious. It is the promise of web based widgets to be truly interoperable; interoperable across handsets and interoperable across service providers. But it easy to show this interoperability may break using a pure certificate based security model, where privilege is implicit in the signing authority. If an application is signed by handset manufacturer A how will it work when I try to run on a phone from manufacturer B, or if my widgets is signed by operator C how will it work when i try to run it when I am using a SIM from operator D.

# 3 PRINCIPLES OF THE BONDI SECURITY MODEL

The BONDI security model attempts to address each of these issues.

### 3.1.1 SEPARATION OF POLICY FRAMEWORK AND POLICY

BONDI will clearly separate the specification of a policy framework, which can be used to describe many different policies in a declarative fashion, and any discussions relating to the definition of default or mandatory policies.

### 3.1.2 AUTHORISATION

A declarative policy as embodied in BONDI, will clearly separate application authentication (the mechanism to describe the identity of the application) from authorisation (the mechanisms for describing the rights that are granted to this application).

The policy framework defined in BONDI will enable authorisation decisions to be based on a number of defined attributes associated to a web application, including but not limited to: the identity of the web application, the identity of the author of the web application, the identity of the issuer of the root certificate associated to a web application or implicit identity methods like runtime hashes. The identity mechanism used to make an authorisation decision will be determined by the security policy of the device.

### 3.1.3 DELEGATED AUTHORITY AND TRUSTED ADVISORIES

BONDI acknowledges that end users may not be interested in or capable of manually configuring fine grained access controls. BONDI will support the notion of an external entity that can be used to aid or set the access control decisions made by a user's device. There are many parties that could fulfil this role including: operators, handset manufacturers, application download portals, anti-virus companies, internet service providers and peer informed information exchanges. BONDIs technical specification will not preclude any of the above scenarios

### 3.1.4 END USER CONTROL

BONDI enshrines the notion of end user choice. An end user should be given either the explicit or implicit ability to select which authorities or advisories are used to inform the permissioning. An end user should have the ability to restrict permissioning related to their own private data, or any other service which may impact their bill.

### 3.1.5 DEVICE RANGE

BONDI is not just intended for high end handsets, and should consider the limitations of resource constrained devices.

### 3.1.6 EXTENSIBILITY

BONDI recognises that there is a commercial need to differentiate on service and capability and this creates a technical pressure to create new APIs. BONDI will ensure that an API extensibility mechanism is in scope and that this extensibility mechanism will run within the constraints of the BONDI security framework.

### 3.1.7 DEVELOPER ECOSYSTEM

BONDI specifications should be created sympathetic to developer needs: security infrastructure and specifications should be created to allow developers to freely innovate and the specifications should not preclude commercialisation routes with low barriers of entry.

### 3.1.8 REAL IMPLEMENTATIONS - REAL USE CASES

The final principle and possibly the most important is that the BONDI approach will be grounded in the real world, looking at real existing security models for reference and addressing the everyday use cases. What does this mean? It means that a measure of success of BONDI is that it can:

a) Be capable of modelling, within its flexibly policy description both existing security models in the market and more innovative varieties

b) Be able to support real user scenarios – such as swapping phones or operators.

# 4 IMPLEMENTATION OF THE BONDI SECURITY MODEL

These are all worthy goals but how in reality can these be satisfied? BONDI is implementing a candidate solution to these issues as an open source project. For security issues more than any other it is imperative that we can empirically prove and “destruction test” the proposed solutions.

- XML specified policy: the security policy is explicitly specified in an XML representation, which enables both flexibility and manageability. The policy description framework must be capable of modelling existing security policies that exist today, as well as more innovative solutions.
- Management protocols: secure management protocols will be defined which will enable delegated authority and trusted advisory models.
- Multiple identity models will be supported: an application can be identified by domain certificate or hash
- It is the intention to derive candidate specifications, from a proven open source implementation and submit these to W3C for standardisation.

# 5 NEED FOR CONSISTENCY

Assuming that the issues raised above are understood and that the principles of a solution accepted, there is still a final question to be tackled: how much of the security model should be standardised and how much left to proprietary mechanisms. This is of course is an open question and will be subject to healthy discussion within the standardisation process but we should be driven by the healthy desire to keep widget level interoperability as high as possible. This means the same widget working on: different widget frameworks/browsers, different operating systems, different handset manufactures and different operators.

The experience of different application technologies has clearly shown us that even though we may achieve API consistency, security fragmentation can inhibit the interoperability scenarios described above, and therefore standardising the critical elements of the security framework will be essential to success.