Proton: Enabling ITOM Visibility into The Browser

Web browsers are a unique blend of application and platform. Unlike infrastructure components of the past, the browser mixes aspects of different technologies into one. This merging of tech-enabled the browser to be the dominant market-changing tool it has become. In combining these aspects, the browser has presented a set of IT challenges in operations management. End users and business units rapidly embraced the browser due to its flexibility, and IT enjoyed the benefits of rapid deployment and commoditization.

As a result of this ubiquitous popularity, the role of the browser in Enterprise IT has exploded. More applications are accessed through the browser than conventional installed apps. Browser usage and adoption occurred so quickly that IT has been playing catch up. IT had to stay on top of the potential security, performance, compatibility, and operational issues created by rapid adoption. With frequent iterations of browser releases and new web technologies, IT management tools remain behind the curve. Initial efforts to manage the browser as a desktop application failed, not due to IT staffing limitations, but rather the fundamental mismatch of ITOM tools with the browser.

Consider the functionality of a browser:

- Web browsers don't have a specific purpose – they are a platform tool
- Web browsers must take unknown, loosely defined content and 'assemble it'
- Web browsers have models for data and visual layout
- Web browsers have their own Application Programming Interface (API) model
- Web browsers have particular scripting languages
- Web browsers offer the ability to load extensions for features or capabilities not yet included
- Web browsers have greater surface area for attack (as a result of the design)

The net of these features ends up making the web browser behave like a cross between an Operating System (OS) and an application. Given these features, does it make sense to try and manage a web browser like you would a traditional app? Traditional ITOM tools provide visibility *to* this critical "application," but not *within* it.

Foundations in Client/Server Computing

After breaking down how web browsers are different from other executables, it becomes clearer how existing IT management tools don't translate effectively into managing a web browser-based workspace. The alignment mismatch isn't just around the application and platform-tools themselves – there's often

a disconnect in how organizations broadly approach the topic of managing browser-based applications overall.

Early technology management thinking on browser-based applications 'ignored' the browser and focused on the endpoint. Security for web applications was mostly conceived of around the server. This approach makes sense when things are viewed as a natural evolution. In reality, the browser is quite far from that comparison.

In the analysis of managing and securing client/server applications, IT naturally focused on 'where' they saw the application running. The end-user workstation ran a client application that was in most senses, a 'dumb' terminal leveraging the host OS. The end-user workstation did little to nothing from a computational standpoint and was useless without a server connection. Some form of application packaging, delivery, and management was required, but none of the workload connected to the end-user workspace.

Focus naturally turned to the server, the traditional emphasis for application management. Security then moved to the server, with credentials stored and processed there. The server itself needed to be patched and maintained. Data entered into the client was processed and analyzed on the server. The client performed no processing or computations on the data passed back from the server. What mattered was understanding the health, security, and configuration of the server to ensure proper service delivery levels.

With a client/server data model as the template and an entirely passive client application, it's logical that the client application endpoint management was virtually ignored. It would make no sense to consume resources and expenses on measurement and control of components that yield no actionable value. Unfortunately, as traditional client/server models have been replaced by web-based applications, the thinking on endpoint management didn't evolve in concert. Given the overall speed at which web applications began to dominate, most in IT hardly had time to consider the data and design model changes brought about by web-apps.

The Web Browser as 'Killer App'

The web browser offers more than just the ability to offload some computational aspects of the client/server application. A web browser can run an entire application itself, only needing a 'dumb' server to provide base code. Most web applications don't run entirely on the browser side, so instead, let's focus on the more conventional approach where the web browser does most of the processing.

Having 'smarts' in the web browser unlocks a fantastic amount of potential. As noted previously, web browsers deliver an array of client-side computational feature aspects that make it more OS than an application. With both a scripting language and an API model, the web browser can deliver almost limitless experiences. Developers could now extend the capabilities of their applications without having to create infrastructure themselves. With the proliferation of readily available scripting libraries, they

could easily consume baseline or pre-built solutions. Suddenly the browser made delivering end-user applications a breeze.

This new functionality wasn't limited just to developers porting applications. IT administrators could now start to see a reduction in the number of system components and managed packages. Previously IT administrators would need to create, package, and update end-user applications for installation and distribution. Changes to core line of business applications required long lead times and extensive testing procedures before a new version could deploy.

Having applications run in the web browser removed a massive part of that management picture. Apps could update by replacing server-side code, and the next time the user accessed the web application, a new version would be available. At least updates appeared that way to the end-user. The web browser model inadvertently created a false sense of ease and security for IT and security admins. What seemed like a win/win situation with easier deployment and centralized management wasn't the whole story.

The browser gets deployed without being deployed

More than a decade after they have become the core of the 'modern desktop' and the most used application in the end-user environment, the same issues remain. These issues are further complicated because IT never really deployed a browser in the first place. Internet Explorer came with Windows and therefore was used by default. It might seem simple, but the mere fact that IT didn't manage the initial rollout of the browser has arguably had a lasting impact on the browser's perception in the business environment.

Leaving the legal battles aside, the notion that the browser was a component of the operating system allowed it to get an install base larger and faster than any other single enterprise-level application. Since IT didn't go through the standard process to package and deploy the browser, teams never went through their routine exercise to address application lifecycle management. After all, how many IT administrators even think about needing tools to manage notepad or calculator? Those tools just exist, and unless the end-user reports a problem, no one ever thinks about them.

So, the browser ended up on all end-user machines, and unlike notepad or calculator, the browser was a critical line of business tool. End users relied on it regularly, even in the earliest days, as the vehicle to access core applications. Over time the list of demands grew, but without IT following their standard software process. Web applications were 'deployed' by publishing code to websites. Users were driven to those new tools by links communicated via emails or through connections in other web applications.

IT intervention was required for sites that required third-party tools like Java or ActiveX controls. ActiveX represented a unique approach in that many of the controls were built into Windows, so IT wasn't involved. Specialized vendor applications did require IT involvement for deployment. Still, ActiveX controls themselves were viewed as a 'useless' piece of binary code rather than a standalone application. Meaning, there were no appropriate testing or validation processes, and IT would only deploy the controls because it was required.

Looking back now, we can see just how problematic this process truly is. We allowed the most powerful and broadly used application to be rooted in the environment without utilizing our time-tested methods and validation techniques. This process also helps explain the challenges of calling it ‘just a browser’ as well as describing why most IT administrators never thought about how to manage the browser itself. Instead, IT looked to existing ITOM tools they owned as a way to manage the browser.

Limitations of existing ITOM tools

There are no enterprise organizations that operate an infrastructure without some level of ITOM tools. The vendors and specifics will vary, but the software will always have a core set of functionalities based on supporting and controlling the end-user operating system and application set. These tools provide critical visibility into the end-user environment to address and resolve performance, reliability, and security issues - among other uses.

Having established that the browser is unlike other applications, it then stands to reason that existing tools would not necessarily translate for the browser's needs and design. Existing ITOM tools were designed for a world where the browser didn’t exist. Those tools do their prescribed functions very well, but they don’t manage or control browser activity.

The ITOM industry had addressed similar problems before – new technologies arose, and they adapted their offerings to deliver functionality. So, it’s no surprise that IT thought ITOM tools would inevitably add browser management functionality. Unfortunately, in response to the rise of the browser, ITOM vendors simply focused on the conventional installed characteristics. They looked at the browser executable no differently than a standard software package. The problem is that the browser isn't like these other applications, and traditional ITOM tools lack the information, insight, and details critical required for proper management.

ITOM tools entirely miss the ‘chewy center’ of the browser by using the standard application management model. They can report on high-level functions of the browser process – did it launch, how much memory is it using, etc. That is a fraction of the information needed to manage the browser environment. Without insight into the user's page-level activity, organizations are operating in the dark. Traditional ITOM tools are unable to provide answers when questions arise about web application usage, performance, security, add-ons, Java, dependencies, or any range of data regarding end-user browser activity.

Delivering Web Browser focused insights

That's where Browsium Proton comes in - Proton fills in the gaps left by other ITOM tools. It's a complementary tool, not a replacement, to an organization's existing toolset. It can be thought of as a module enhancement. Proton is built to look from inside the browser and can deliver critical insights that existing ITOM tools lack. Proton's data can be used in conjunction with the organization's existing management tools to manage the entire environment properly. When used with other Browsium apps, Proton data can be turned into actionable insights within the browser. Proton gives IT the ability to granularly manage settings, browser functionality, application compatibility, and security from within the browser.

Proton sheds light on the dark areas left behind by other ITOM tools. Proton takes a fundamentally different approach to gather and monitor data from inside the browser itself. This provides the most comprehensive data for browsers, web applications, and web application dependencies better than any tool on the market. Providing this data and the resultant insights in one place enables IT to regain control of the missing pieces of their IT dashboard.

For example, measuring performance data on the webserver provides a view from the outside, which doesn't reflect when loaded objects are usable. Proton performance data from within the browser ensures complete insight into the whole end-user experience. The data shows the entire range of the process from query to information calls to object load and when the page is user ready. To help IT administrators go deeper, Proton data can yield insights into what content and types of objects were loaded on that page. These insights produce actionable information to identify and resolve issues. Proton provides multiple layers of data appropriate to the browser and web application model. It delivers all the answers in one place, which is critical to saving time and financial resources.

Proton also fills the "ITOM visibility gap" when looking at 3rd party browser components. The most common example of that type is Java, but a range of other tools needed to augment browser functionality are included as 3rd party components. In the case of Java, existing ITOM tools are great at revealing Java runtime environment installations. They can deliver information about the machines with Java installed and a complete list of which versions are on each machine. Then the visibility gap comes in, and they are unable to tell anything more than that for any web or browser-based Java usage.

Proton goes beyond just data on where Java installations exist – it includes the version, requested version, what systems used which releases, and other critical data for managing Java. From a security and application rationalization perspective, Proton can deliver a report on which versions (if any) use which website's web applications to aide in quickly removing unnecessary installs. Proton can even provide data on non-web-based Java application usage. This usage further provides critical operations data on how, when, and where Java is used. Without this added level of detail, it is impossible to manage Java effectively and safely in the enterprise.

Additionally, existing ITOM tools are limited and entirely blind to browser extensions. With all modern browsers using JavaScript-based extensions, the components are not actually 'installed' in the conventional sense. Traditional ITOM tools look for any footprints in the system registry or application

related data stores. Meaning, they lack ways to detect non-installed browser extensions. Proton is designed for the browser environment, so it can see precisely show what extensions are added to the browser.

Proton can not only detect their presence but also provides the usage data that IT needs to evaluate and quantify risk. Extensions don't require administrator-level permissions to install so that any end-user can access and install them on their systems. Extensions can do a wide range of things and present a unique security threat. Some common examples are password managers that monitor all browser traffic and can read as well as inject content, and unwanted extension screenshots in regulated industries.

This type of data and detail is critical to ensure that IT and security administrators know what is truly going on at that endpoint. Traditional ITOM tools aren't designed around the browser model, so organizations end up with blind spots and visibility gaps around these types of critical data. Proton ensures and enables enterprises to control and manage their systems at the granularity levels they have come to expect in their environment.

Manage the entire desktop

The browser is a fixture of the modern end-user computing environment. The browser radically changed how IT delivers services, and in turn, it requires a change in how IT manages the situation. It is not possible to get a complete ITOM picture without the proper tools in place. Continuing to manage by relying on existing ITOM systems will keep IT from controlling the environment and achieving the mission. Adding Browsium Proton to your ITOM toolset will instantly augment existing ITOM solutions and deliver critical insights.

Proton provides visibility and sheds light on the dark corners current ITOM tools leave unchecked. Advancements in web browser capabilities demand that IT has the tools and strategy needed to manage browsers today and into the future. The start of this strategy is to gain real visibility into the browser environment. Proton turns on the lights to provide the most in-depth view across all browser platforms and delivers the tools to fill the gaps of existing ITOM environments required for the modern desktop.