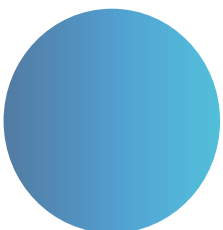


White paper:

THE VARNISH PAYWALL

– From general content to paid premium content, control access to your content the way you define it.



From general content to paid premium content, control access to your content the way you define it.

Introduction

This white paper outlines the different restrictions that can be placed on content with the Varnish Paywall™, how integration with the various other systems can be implemented, how the Varnish Paywall™ ensures that subscribed and regular users will be able to retrieve your website's content equally fast and what pitfalls should be avoided when implementing it.

Over the last couple of years content heavy websites, digital media in particular, have increasingly been moving from a business model purely driven by advertising to a model based on a combination of both advertising and a digital pay-for-access method, a so called paywall. A paywall allows a website owner to set and use arbitrary rules to restrict access to their online content. The paywall requires authentication and authorization before it delivers the content to the user and traditionally website owners have felt that the logic behind this process, given that these are often a complex set of rules, must be placed in the application server layer. But as the application server layer is slow and hard to scale, placing the paywall logic there can significantly slow down a website's performance. With the Varnish Paywall™ this challenge is no longer true. And since performance is a very important element of any web management strategy the Varnish Paywall introduces a solution that will help solve a challenge many are struggling with: a way to control access to online content without sacrificing web performance.

What you will learn:

This white paper outlines the different restrictions that can be placed on content with the Varnish Paywall™, how integration with the various other systems can be implemented, how the Varnish Paywall ensures that subscribed and regular users will be able to retrieve your website's content equally fast and what pitfalls should be avoided when implementing it. **From general content to paid premium content, control access to your content the way you define it.**

What is Varnish Cache?

Varnish Cache™ is a web application accelerator. The software speeds up a website by storing a copy of the page served by the web server the first time a user visits that page. This is referred to as caching. The next time a user requests the same page, Varnish® will serve the copy, or the cached content, instead of requesting the page from the web server. The end result? Your application servers need to handle less traffic and your website's performance and scalability go through the roof.

The Varnish Paywall is installed in Varnish Cache and its functionality can be implemented on any kind of content across all platforms.

Sources

- [1] The MD5 Message-Digest Algorithm is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value.
- [2] The Varnish Configuration Language (VCL)™ is a small domain-specific language used to define request handling and document caching policies for Varnish Cache. When a new configuration is loaded, the varnished management process translates the VCL code to C and compiles it to a shared object which is then dynamically linked into the server process.

Different access control schemes

When deploying a paywall one of the more time consuming parts is for the business organization to decide what sort of restrictions are to be placed on the web content. For most web editorial staff this is new territory and it can be difficult to weigh the balance, to get optimal distribution of the content and to monetize it.

These are some of the most common approaches to digital content restrictions within the Varnish Paywall™:

Subscriptions

The most common and best understood limitation on content distribution is to require the user to have an active subscription. When a subscription is verified the user gains access to all restricted content.

Metered

With the metered approach the amount of content each user can access is based on certain restrictions governed by arbitrary rules set by the content owner. For example, a baseline limit of five articles per week can be made available for the user which could then be doubled to 10 in instances where the user is willing to sign in and leave behind some information, such as a name and telephone number or email address. The meter could then be reset for each new week or month and Varnish would keep track of articles already viewed by the user so that they are not charged if the viewed articles are revisited.

Product plans

With the product plan approach multiple subscription types are made available to the user. Subscriptions may be based on certain content, e.g. all sports content across all publications. Each piece of content then carries information on what subscription type/product plan allows access to the content the user is trying to access.

Handling non authorized users

Once you've set up a Varnish Paywall, Varnish will only grant access to a user after verifying their identity. Varnish however, doesn't necessarily have access to the central user database so it relies on a Single Sign On (SSO) service to do the actual authentication. After successful authentication the SSO service must set a cookie to relay the users authentication to Varnish. The cookie can then be made tamper-proof with a cryptographic signature.

The cryptographic signature works by calculating a hash string for the content of the cookie and a shared secret ("password") that has been set earlier and is recognized by both the SSO service and Varnish Cache. The signature is then appended to the cookie. Once set up the cookie might look something like this:

```
userid=perbu,expiry=1334569718,signature=e19799f8f1b6bbf6c42c730b792f0312
```

If the shared secret is for example "rabbits_are_cool" and we decided to use MD51 as the hash function the signature would be generated like this:

```
$ echo userid=perbu,expiry=1334569718,
rabbits_are_cool | md5sum
e19799f8f1b6bbf6c42c730b792f0312 -
```


In order to verify the cookie in Varnish you would need to extract the cookie into a variable or a header and then have some VCL2 that looks something like this:

```
# We've parsed the cookie here into:
# x-auth-cookie (the whole auth cookie)
# cookie-expiry (the expiry time, in epoch format)
# The cookie format needs to be verified beforehand

if (now - std.integer(http.req.cookie-expiry, 0) > 0) {
    # cookie is OK - not expired
} else {
    # cookie is expired - show login/signup-page
}

if (http.req.x-auth-cookie == digest.md5("rabbits_are_cool",
    "userid=perbu,expiry=1334569718") )
{
    # the user is OK. We can serve content...
} else {
    # something fishy going on.
    error(406, "Subscription required");
}
```

If needed, Varnish can also verify the user's identity through a lookup in a web service, a database or other data source. That would be somewhat more complicated and could slow down the Varnish Paywall™ as it would have to retrieve information from another service for every single request. Varnish usually serves content straight from memory so accessing a network service will result in a significant slowdown.



User authorization within the different access control schemes

Varnish now recognizes the user attempting to access your content and can move on to check whether the user is actually authorized to do so. The implementation of this authorization process would vary depending on the access control scheme

Choosing the appropriate access control scheme

It is possible to choose what form of authorization schemes should be enabled on a per object basis. The content management system sets the HTTP header X-Authorization to signal to Varnish what access control scheme should be used on the content requested.

It is possible to use several different access control schemes on a website according to your needs. Hence, certain content may be made freely available while other content requires a subscription and yet another class of content would be metered.

The Varnish Paywall™ is dynamic and nearly all configuration is derived from the content management system behind it. Content can therefore be moved behind the paywall at any given time without any reconfiguration of Varnish.

Authorization revalidation

If there is an occasional need to revalidate the authorization, for instance to check whether the account has been revoked, Varnish can be setup to handle a situation where the authorization has expired without needing to re-authenticate the user. There are two ways of doing this:

1. Silent reauthentication

Using client side asynchronous revalidation a client side script can be used to examine the session cookie and resubmit it to the SSO service. If the SSO service finds that the user is eligible it can reissue the session cookie. The website can use this method to extend the session without the user noticing.

2. Redirection to SSO

Another way is for Varnish to issue a redirection back to the SSO service when the session cookie has expired. The SSO service would then examine the user, reissue a new cookie and redirect the user back to the article. The user would feel a negligible impact as the page would take a little longer to load than usual but there should be no need to prompt them for username and password.

Subscription authorization

The authorization process for the product plan approach is slightly more complicated than the others since an advanced subscription authorization entails a many-to-many relationship between users and subscriptions as it is possible for a user to have a multitude of subscriptions (product plans) with this method for access control.

The easiest way to implement this subscription method would be to have a web service answer simple queries containing a user/product pair with a simple 'allowed' or 'denied'. The result of this can be cached in a cookie so that for instance the last 10 products each user is allowed to access are stored in the cookie. This cookie would also carry an expiry time so it would eventually time-out triggering a reauthorization of the user.

A clever way to speed up this authorizing web service would be to route it through Varnish Cache.

Sources

- [3] Memcached is an in-memory key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.
- [4] Redis is an open source, advanced key-value store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets.
- [5] A central processing unit (CPU) is the hardware within a computer system which carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.

Metered authorization

When running the authorization process for the metered method for access control, Varnish needs to keep track of which articles have been read within the current specified time period. The simplest way to do this is to maintain a list of article IDs. This ensures that even if the user reloads the page no article is counted twice. One would also need to maintain a timestamp to know when the user is given a new quota of articles. This string could either be pushed to the client and stored in a cookie or stored in a data service such as memcached³ or redis⁴.

The advantages of the Varnish Paywall

As explained above the commonly used solution for implementing paywalls is to place the complex rules governing access control within the application server layer. This typically results in the content being served from this layer, significantly slowing down the website's performance. However, placing the rules for access control within the Varnish caching layer, the website's performance is no longer a challenge. In fact one of the major advantages of the Varnish Paywall is its positive effect on a website's performance.

Here are some of the other key advantages of the Varnish Paywall:

Scalability.


Varnish will retain its scalability with the paywall deployed. The Varnish Paywall is compiled to native code which on a modern CPU would run so fast that the time taken to execute the logic will be insignificant.

Speed.

Varnish Cache is built for speed. It executes its policy code more or less a thousand times faster than your typical Java or PHP based application servers, mostly due to the fact that the configuration is compiled into machine code free of system calls. System calls require expensive context switches, stall the CPU⁵ and wreck havoc in the CPU cache. Avoiding system calls hence speeds up the execution speed.

Flexibility.

What makes the paywall solution from Varnish unique among pay-for-access solutions is its ability to integrate with any type of publication and payment system. All web content management systems (CMS) communicate through the HTTP protocol. The authorization and authentication schemes outlined in this white paper are all



Varnish now recognizes the user attempting to access your content and can move on to check whether the user is actually authorized to do so.

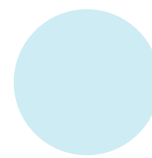
done outside the CMS. For convenience we recommend that the CMS is altered to send specific HTTP headers along with the content to indicate what schemes should be used but this could also be a static list of URLs.

If Varnish needs to communicate with a network service in order to authenticate or authorize users this could happen through a multitude of protocols. Varnish supports ReST services. It is possible to connect Varnish to any open source database as long as a C or C++ API is available.

Key takeaways

As a response to less than encouraging ad sales most content heavy (e.g. digital newspapers and magazines) websites are currently looking into ways to get paid for content without sacrificing scalability, flexibility and speed. The Varnish Paywall is a great alternative for these organizations because by placing access-control rules within the Varnish layer you ensure that your website remains as fast and scalable as it was before you implemented a paywall (even faster if you weren't using Varnish for caching). Hence, subscribers will not experience any delay when they request content from websites that have implemented a Varnish Paywall.

The Varnish Paywalls allows content owners to monetize their content in a multitude of ways. The different restrictions can be mixed and matched freely and work across all devices. Another compelling quality of the Varnish Paywall is its ability to integrate with any publication and authentication system





About Varnish Software

Varnish Software is a global pioneer in high-performance digital content delivery. Powered by a uniquely flexible caching technology, Varnish Software's solutions are indispensable common denominators among some of the world's most popular brands, such as Eurosport, Twitch and Tesla. The company's solutions enable organizations worldwide to provide a superior user experience with fast digital content delivery at any scale, while giving them the flexibility to maintain control over their content and make the technology their own.



www.varnish-software.com

Los Angeles - Paris - London
Stockholm - Singapore - Karlstad
Dusseldorf - Oslo - Tokyo



www.varnish-software.com