安全性の画面転送か、利便性の無害化か…。
もう悩まない新Web分離

株式会社アシスト
仮想化事業推進室

セッションの目的

- Web分離製品の「画面転送」、「サニタイズ」の仕組みと違いを**スッキリ**理解いただく。
- 仕組みを理解した上で、画面転送とサニタイズのどちらが良いのか結論を出す！

Discramer:本セッションでご紹介するクリスタルレンダリングは、正式リリース前の機能です。来年初頭にリリース予定ですが、リリース時期の変更や、機能・仕様の変更が行われる可能性があります。

構成

- インターネット分離ソリューションが生まれた経緯
- いろいろあるWeb分離ソリューション
- Ericom Shieldの画面転送の仕組み
- Ericom Shieldのサニタイズの仕組み
- 多面的に比較
- 結論

株式会社アシスト　会社概要

設立	1972年3月
代表取締役会長	ビル・トッテン
代表取締役社長	大塚辰男
資本金	6,000万円
売上高	317億円（2018年度）
従業者数	1,160名（2019年4月現在）　※グループ会社含む
事業内容	コンピュータ用パッケージ・ソフトウェアの販売、技術サポート、 教育およびコンサルティング
本社所在地	東京都千代田区九段北4-2-1　市ヶ谷東急ビル
オフィス所在地	札幌、仙台、名古屋、金沢、大阪、広島、宇部、福岡、沖縄
取引会社数	6,300社（2018年度）
主要取扱製品数	60製品（2019年4月現在）
グループ会社	株式会社アシスト本舗 株式会社アシスト北海道 株式会社のれん

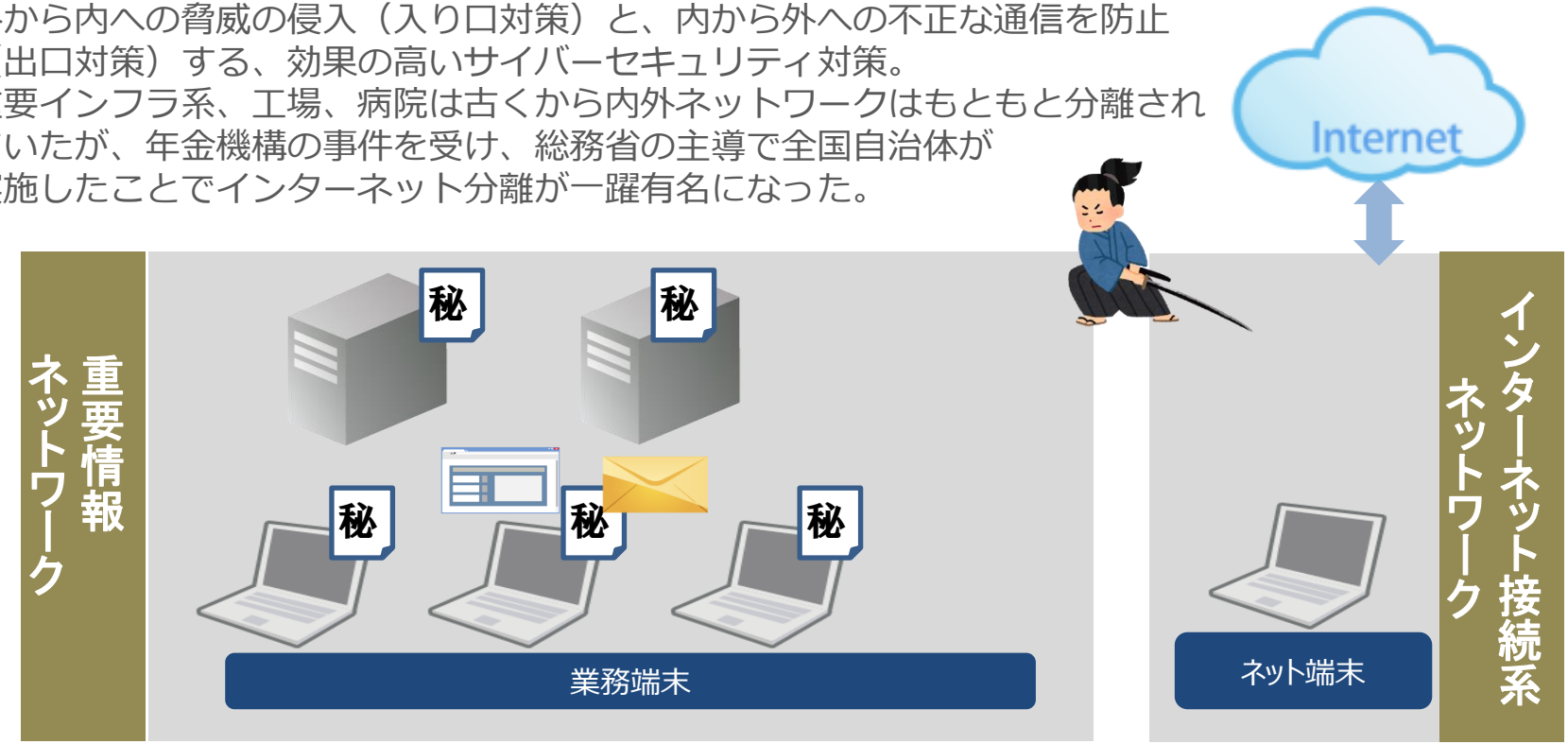
サイバー攻撃対策としての多層防御の現状

対策	抑止・防止効果
ユーザー教育	ポリシー・手順に沿って業務をしても、高度に偽装されたメールの添付ファイルの開封、本文のリンククリックによる感染を防げず、感染の認識も困難
アンチウィルス	パターンマッチされないポリモーフィックマルウェア、ファイルレス攻撃によって標的型攻撃対策としては効果が低い。誤検知や大量のアラートに埋もれてしまうことも
EDR/SIEM/ログ取得	アラートを上げることはできても防止・抑止はできない。対応には高度な技術者によるトリアージと対応の実行権限が必要
サンドボックス	高度なマルウェアはサンドボックスにいることを感知して振る舞い、すり抜ける
Webフィルター	攻撃サーバのドメインは短時間で切り替わるため、ドメインレピュテーションのデータベース登録が行われる前に被害にあう。善意のサイトも改ざんさている場合がある
SSLインスペクション	Webの通信内容をトレースできても不正通信の検知は検知エンジンの次第
メールゲートウェイ	わかりやすい攻撃メール、登録済みのドメイン、実行形式の添付ファイルであればフィルタリングできるが、高度なものはすり抜ける
IPS	高度な攻撃と正当な通信や振る舞いと高度な攻撃の区別が困難
脆弱性管理	既知の脆弱性に対する攻撃は対応可能だが、脆弱性を利用しない攻撃やゼロデイは防げない

上記は対策の一部ですが、すべて対策実施しても防ぎきれないのが標的型攻撃
最大の脆弱ポイントはWebサイト閲覧によるローカルPCでのコード実行

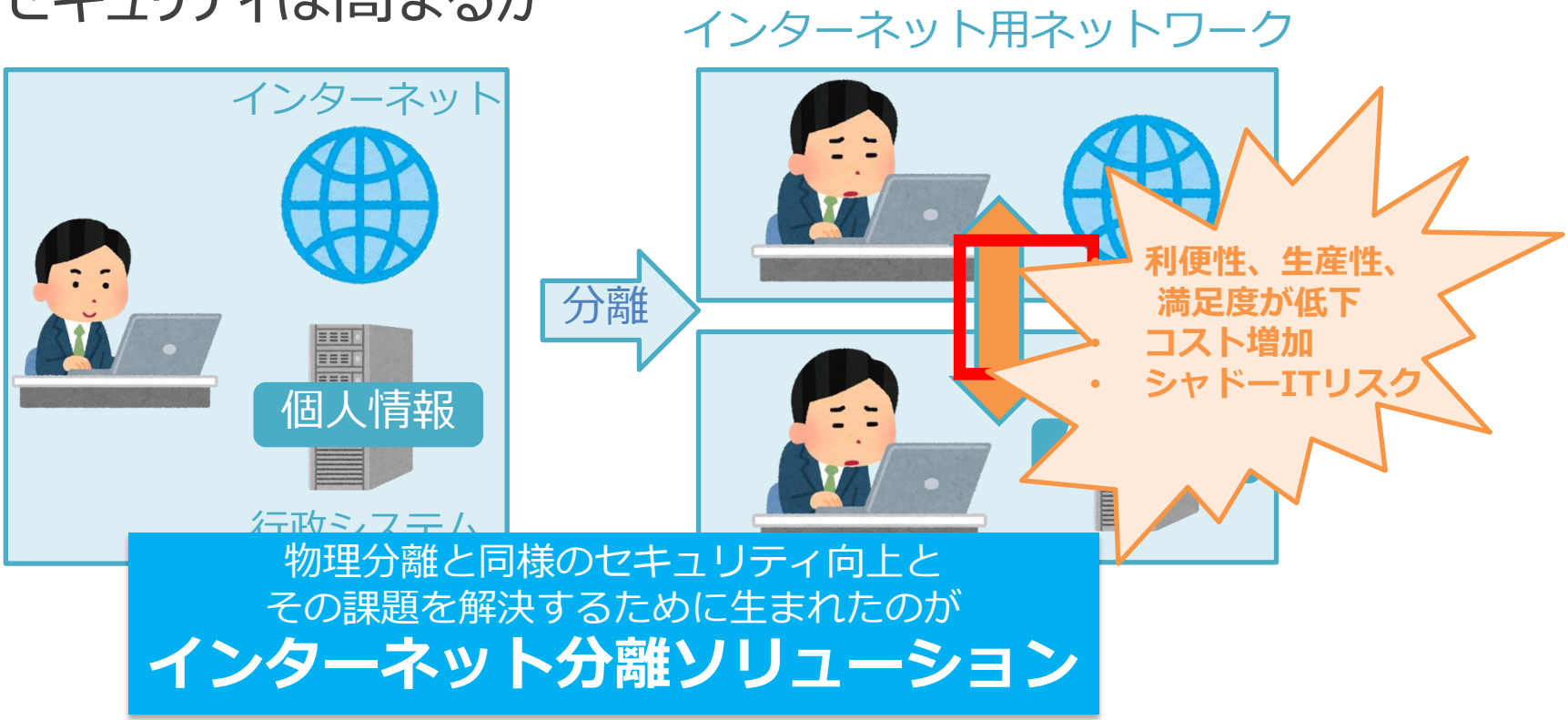
インターネット分離とは

攻撃の9割が入ってくるインターネットと内部ネットワークの**通信経路を分断**し、
外から内への脅威の侵入（入り口対策）と、内から外への不正な通信を防止
（出口対策）する、効果の高いサイバーセキュリティ対策。
重要インフラ系、工場、病院は古くから内外ネットワークはもともと分離され
ていたが、年金機構の事件を受け、総務省の主導で全国自治体が
実施したことでインターネット分離が一躍有名になった。



インターネット分離の最大の欠点

- セキュリティは高まるが…



物理分離と同様のセキュリティ向上と
その課題を解決するために生まれたのが
インターネット分離ソリューション

インターネット分離されると困るアプリたち

ブラウザ
/http(s)

・ネット検索
・SaaSアプリケーション
・Webメール
・CMS
・ファイル共有等

Webアプリ・
アプリ更新/http(s)

・Windowsアップデート
・パターンファイルアップデート
・Skype、Zoom、ハングアウト
等

メール/SMTP

ローカルメーラー

インターネット分離ソリューション

デスクトップをまるごと分離しなくても、
インターネットアプリを分離・無害化、制御すれば良い

インターネット分離ソリューションの守備範囲

Web分離ソリューション

分離環境でWebを実行、害の無い結果のみ取得
本セッションで深堀りします

ERICOM SHIELD

特定アプリと接続先ドメインを
ホワイトリスト化
単純な仕組みで安全性向上

ERICOM SHIELD

メール無害化

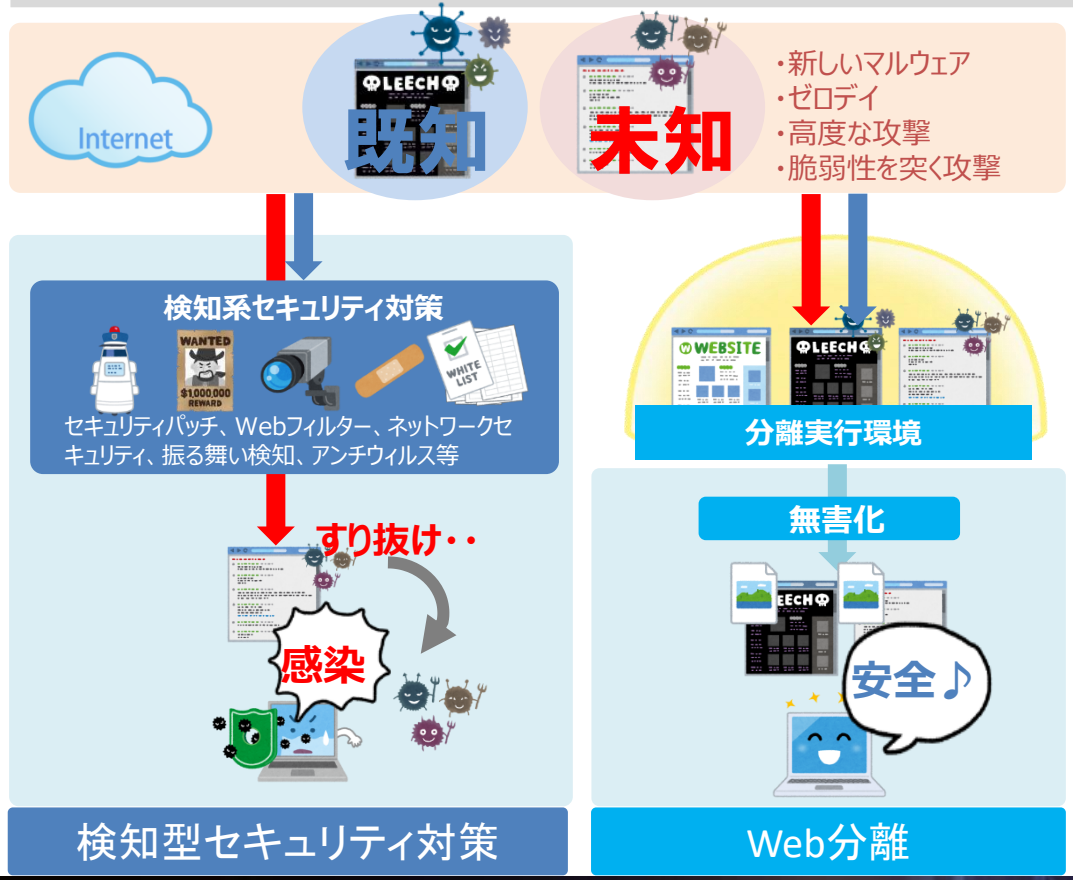
メールを消毒
常識を覆す
新たなメールセキュリティ
Mail Sanitizer®
クラウド
プロット

ブラウザ/http(s)

Webアプリ・
アプリ更新/http(s)

メール/SMTP

検知型アプローチとWeb分離の違い



<Web分離のメリット>

- ✓ **既知/未知の脅威を分離**
- ✓ ゲートウェイ、エンドポイント、不正通信検知、SOC等への**過剰なセキュリティ投資を抑制**
- ✓ **人の脆弱性による感染を防止**
- ✓ ユーザーが安心して業務ができる
- ✓ 端末のマルウェア感染が激減し、内部ネットワークのセキュリティインシデントが激減
- ✓ セキュリティ担当・IT担当が助かるセキュリティ向上施策

Ericom Shieldの特徴

画面転送方式による
Webの完全分離

ローカルブラウジングと
同等のユーザビリティ

Linuxブラウザベースのため
Microsoftライセンスを削減可

タブ毎分離による
感染範囲/リスクを最小化

あらゆるブラウザ、
デバイスに対応

ファイル無害化機能を
ビルトイン

セキュリティ

ユーザビリティ

コスト

Ericom Shieldの特徴

ローカルブラウザのお気に入りを利用可能

ローカルブラウザに既に登録済みのお気に入りを継続して利用可能です。もちろん新たにお気に入りを登録することも可能です。

ローカルプリンタで印刷

閲覧中のWebページを一度PDF化し、ローカルプリンタで印刷を行います。
見えている範囲でなく、Webページ全体の印刷が可能です。

ローカルデバイスの辞書登録を利用可能

ローカルデバイスに登録している辞書登録機能をそのまま利用可能です。

クリップボード対応

リモートブラウザ側でコピーしたテキストをローカルデバイス側のクリップボードにリダイレクトします。その逆も可能です。

動画/音声対応

動画再生のプラグイン等もインストールせずに、安全に動画再生。

ファイルのダウンロード/アップロード

ビルトインされたCDRとShieldの連携により、違和感無く無害化されたファイルをダウンロードすることが可能です。
インターネット側へのアップロードも対応しています。

✓ローカルブラウジングとほぼ変らないユーザビリティを提供

無害化における「画面転送」と「サニタイズ」の争い

画面転送派の意見

画面転送は全Webコンテンツを
分離して実行するから
完全分離だから100%安全です！

（実は仕組みが分かっていないが）
サニタイズは完全分離ではない！
自治体では分離方式として公認されていない！

サニタイズ派の意見

アクティブコンテンツはコンテナで分離実行し、
画面操作の命令だけを送り、感染リスクの
ないコンテンツは端末側でレンダリング
通常のブラウジングと変わらない
スムーズさを実現します！

画面転送は画面がカクカクして使いものに
なりませんよ！
ネットワークも広帯域が必要です！

画面転送もサニタイズも対応（2020年初頭）するEricom Shieldを使って
両方式の裏側の仕組みを解き明かし、スッキリさせます！

本題

Ericom Shieldの場合、
どっちを選べば良いの？

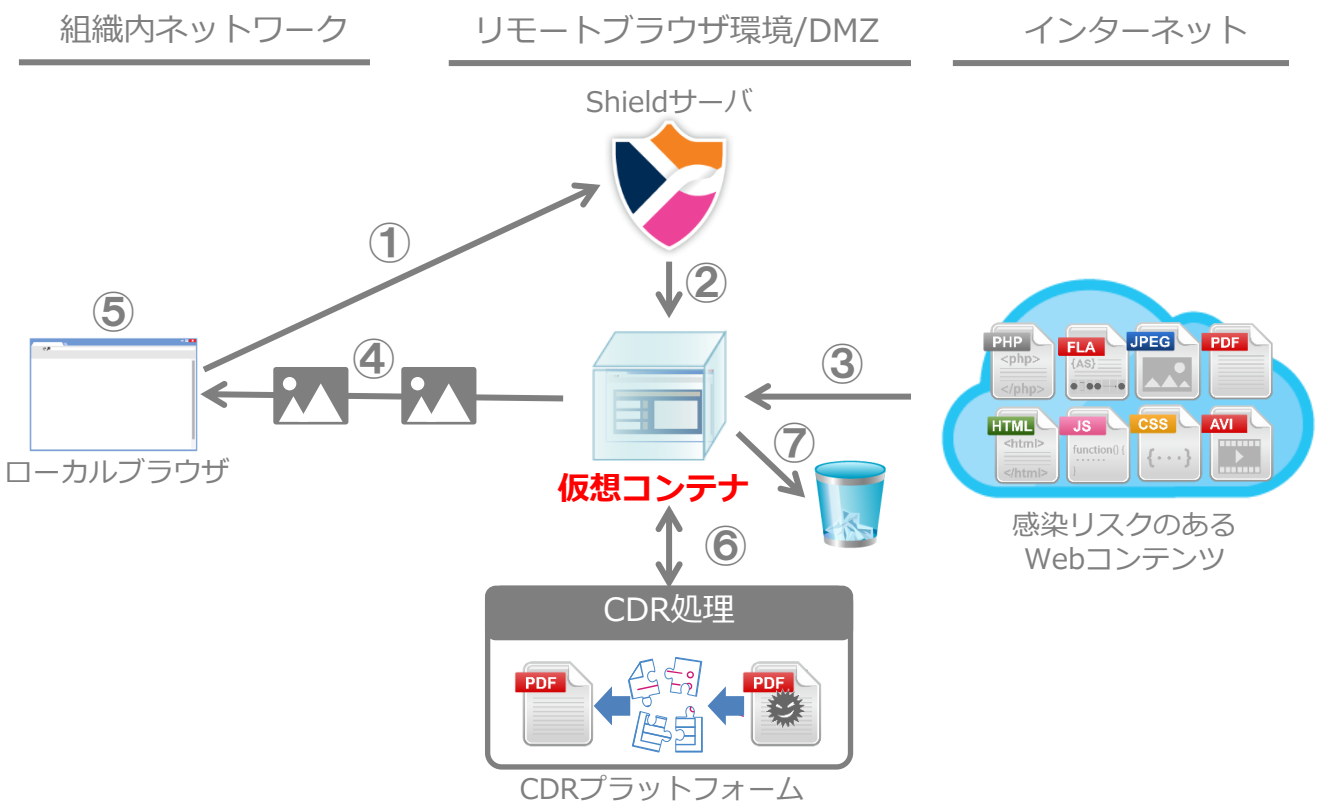
画面転送 **VS** サニタイズ

Ericom Shieldとは

メール、ドキュメント、ブラウザをきっかけにした、
以下のようなあらゆる*Web*経由の攻撃から端末と
内部のネットワークを守る*Web*分離製品です

- 未知の攻撃、未知のマルウェアによる攻撃
- メール、ドキュメント、Webサイト中の悪意のあるリンク
- ドライブバイダウンロード
- マクロ等のエクスプロイトが含まれるドキュメントのダウンロード
- Webアプリのアクセスポリシー制御

Ericom Shieldのシステム構成概要



＜構成上の特徴＞

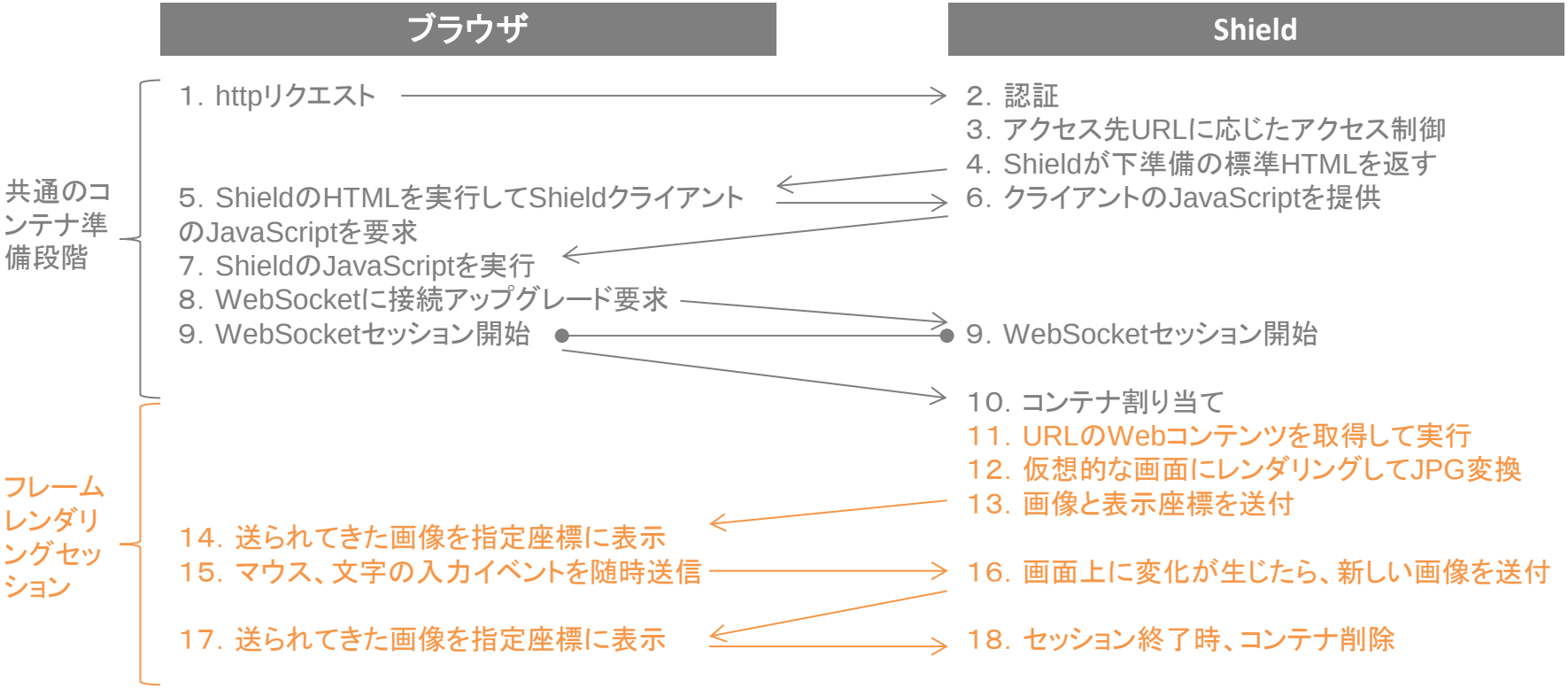
- 専用クライアントも、ブラウザプラグインも不要
- EricomのSSLルート証明書をインストールして、Proxyの向き先をShieldに変えるだけで利用可能
- Shieldの置き場所はオンプレミス、クラウドどちらでもOK

Ericom Shieldの無害化方式の呼び名

画面転送
＝
フレームレンダリング

サニタイズ
＝
クリスタルレンダリング
（2020年初頭リリース）

【フレームレンダリング】Shieldによる画面転送の流れ



【フレームレンダリング】仕組み解明

インターネットWebサイト

Ericom Shield

- Webプロキシ/認証
- アクセスポリシー制御
- ベースのHTMLを返す
- コンテナ管理と割当
- Webコンテンツの実行
- 結果の画面をJPG化して転送
- ブラウザからのイベントをストリーム実行
- 画面差分のJPGと配置座標をストリーム転送

クライアントPCのブラウザ

```
<html><body>
<script src=Cookie読み取り.js></script>
<script src=Shieldクライアント本体.js></script>
<script src=Shieldクライアント設定.js></script>
<canvas ローカルブラウザのウィンドウサイズ>
</canvas>
</body></html>
```

マウス、キーボードのイベント送信

ベースのHTML

ストリームされるJPGがはめ込まれるcanvas領域を定義。ウィンドウサイズを変えるとリアルタイムに変わる。

【フレームレンダリング】動的なトップ画面表示

- **この間約１秒。600KB程度のダウンロード**
- 企業や情報系Webサイトはトップページが動的なものが多いため、常に画面転送は発生する
- 画面はキャッシュされず再利用されないため、常に画像がストリームされ続ける
- **動的な画面を表示したまま操作しなければインアクティブセッションの切断イベント発生まで画面はストリームされ続ける。1分で36MB、5分で180MB**
- FPS値のデフォルトは8FPS/sec

【フレームレンダリング】マウス操作によるイベント発生時

ローカルブラウザ上でマウス移動

コンテナブラウザで実行し、
画面変化を検出

ローカルブラウザのcanvasに上塗り

- ローカルブラウザのマウスの座標情報、クリックといったイベントは常に送信され、リモートのコンテナ側で同じイベントが実行される。コンテナ側で画面が変化すれば、その画面変化がローカルに送信される
- **差分の画面転送は画面全部ではなく、差分が生じた部分とその周辺が送られる**
- 画面内の要素も見た目にも変更がない場合、画面転送は発生しない

【フレームレンダリング】スクロール

- スクロール時のFPSはデフォルトで5に設定され、jpgの圧縮率も高く設定されているためよく見るとにじみが確認できる
- スクロールが完了すると、静止時の圧縮率に戻り、精彩な画面になる
- **スクロール1秒あたり、800KB程度の転送量**
- スクロール時のFPSと、圧縮率は変えられるが、圧縮率は90%にしてもサイズが10分の1にならない

【フレームレンダリング】文字入力

「A」を打鍵

変換候補が表示

さらに変換候補が表示

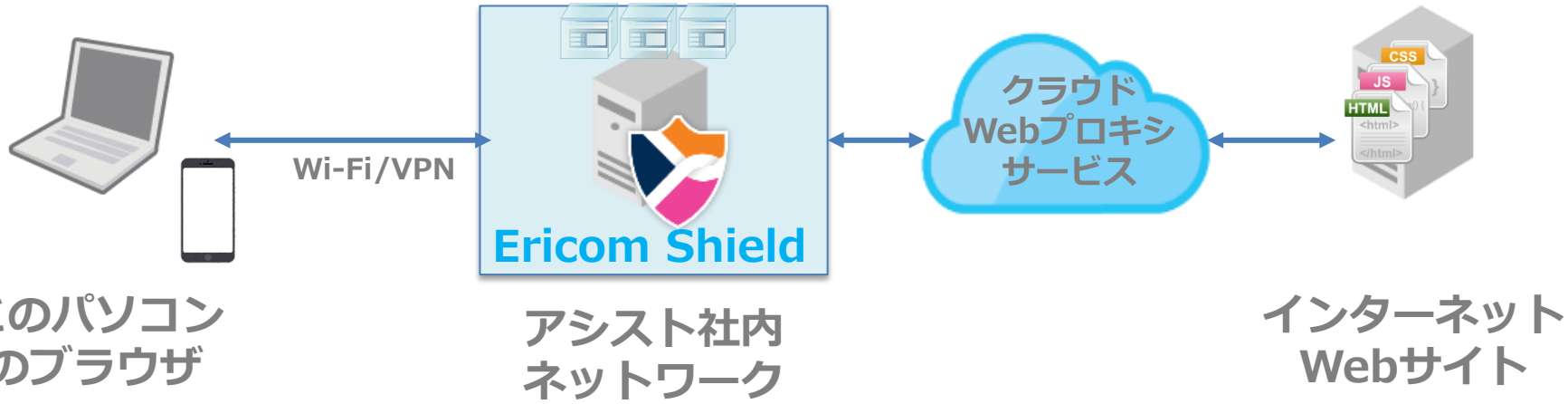
「S」を打鍵

- **アシストと入力して文字変換完了までに約30枚のjpgがダウンロードされた。トータルおよそ3MBのデータ転送量。その間およそ4秒**

【フレームレンダリング】カーソル点滅

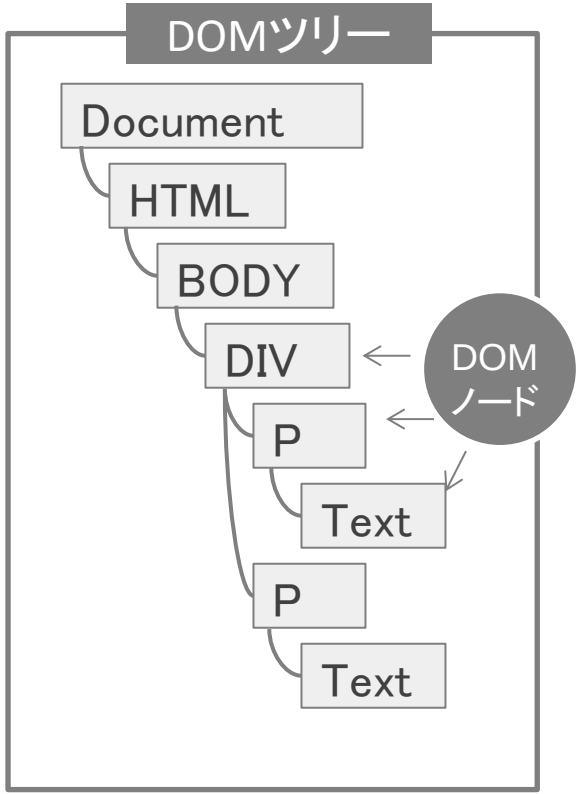
- カーソルの点滅も画面転送の対象になる変化の一つ
- カーソルの点滅を無効にすることはブラウザではできない

デモ環境

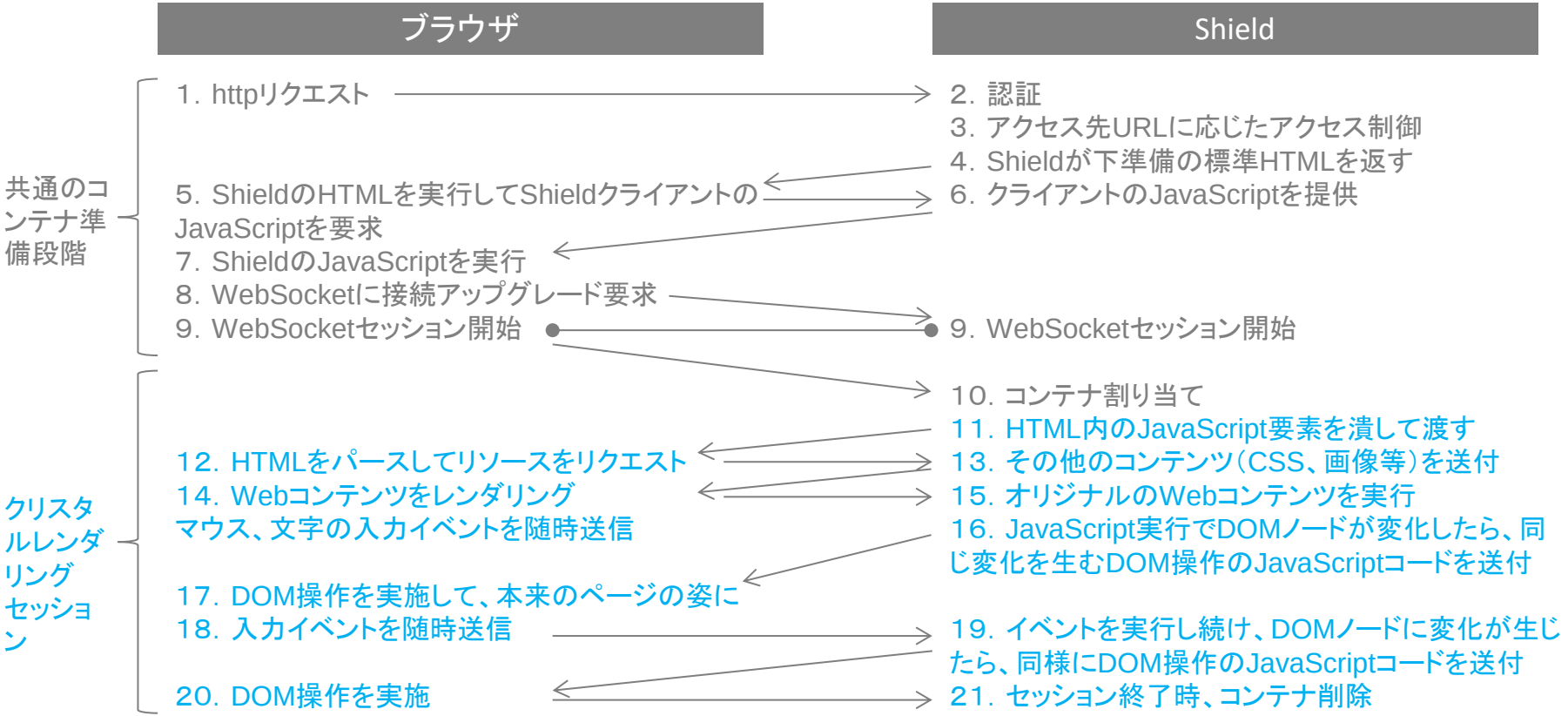


クリスタルレンダリングを理解するためのキーワード「DOM」

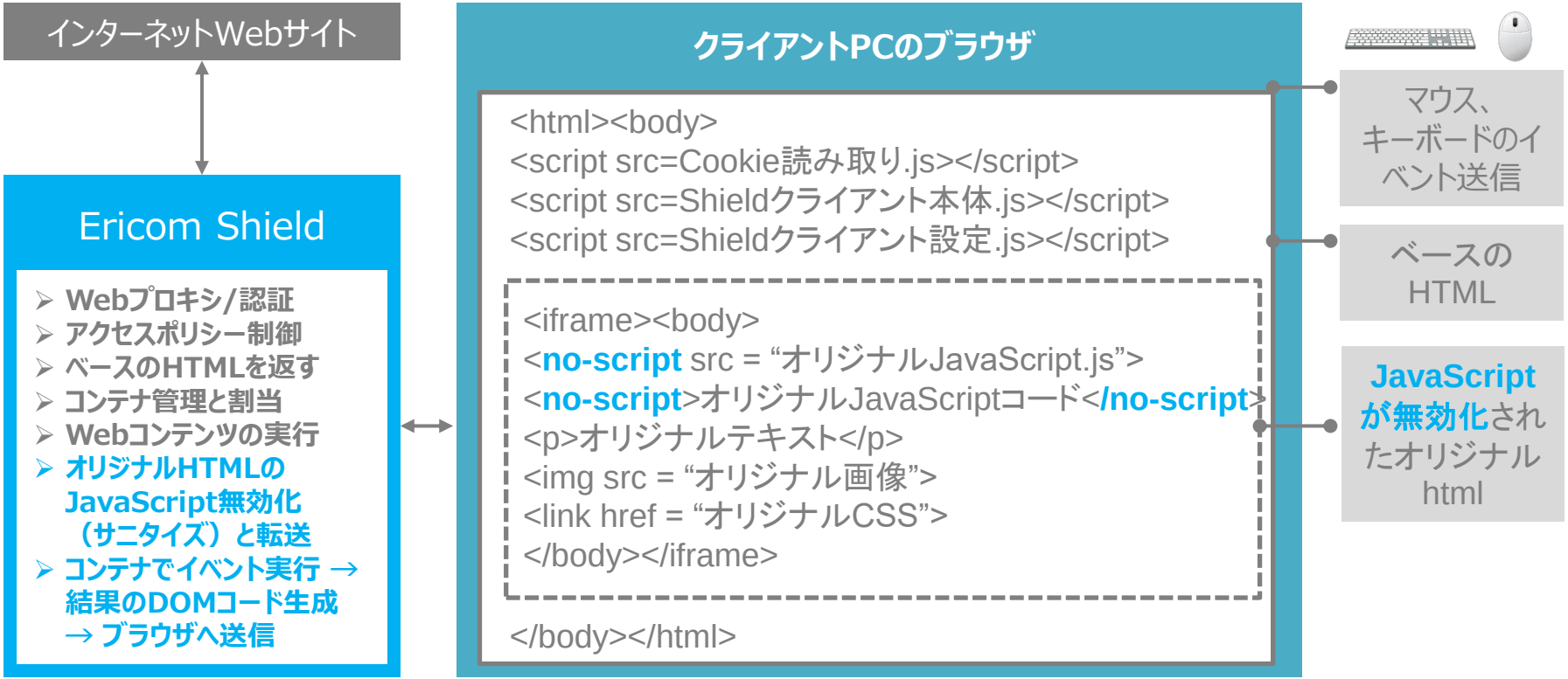
- HTMLをブラウザが読み込むと、ブラウザ内にDOM （Document Object Model）ツリーが作られる
- このツリーを構成するすべてのDOMノードを、JavaScriptでつついて好きなように操作して見た目を変えられる
- 文字や画像を変えたり、背景や文字色などのスタイルを変えたり、アニメーションのようにダイナミックに動かしたり、新しいDOMノードを追加することもできる



【クリスタル】Shieldによるクリスタルレンダリングの流れ



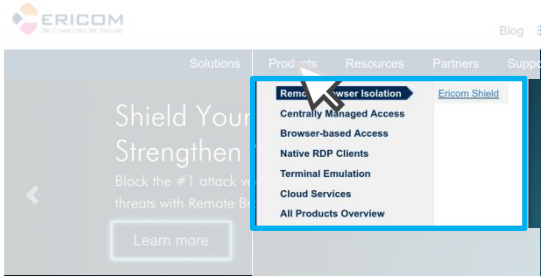
【クリスタル】仕組み解明



【クリスタル】オリジナルJavaScriptを使わずに結果をエミュレーション

ブラウザ	WebSocket通信	Shieldコンテナブラウザ

①「マウス位置は x:300px y:50だよ」

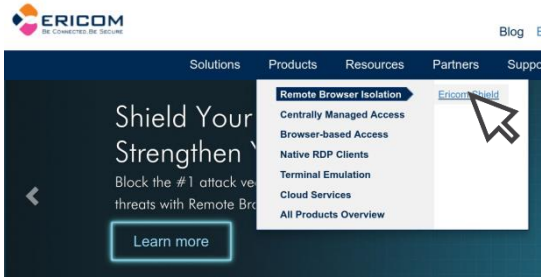


④「リンクメニューのDOMノードを作成して、座標 x:300px　y:80に表示せよ」

②コンテナ上でマウスイベントを実行
③DOMノードの変化を検出

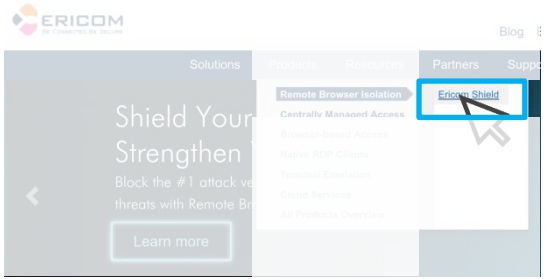
⑤指示通りにDOMノードを作成し、指定位置に表示

⑥「マウス位置は x:400px y:100だよ」



⑨「Ericom Shieldというテキストの DOMノードに「マウスオーバーされたときの文字スタイル」を適用せよ

⑪「マウス位置 x:300px y:100でクリックしたよ」

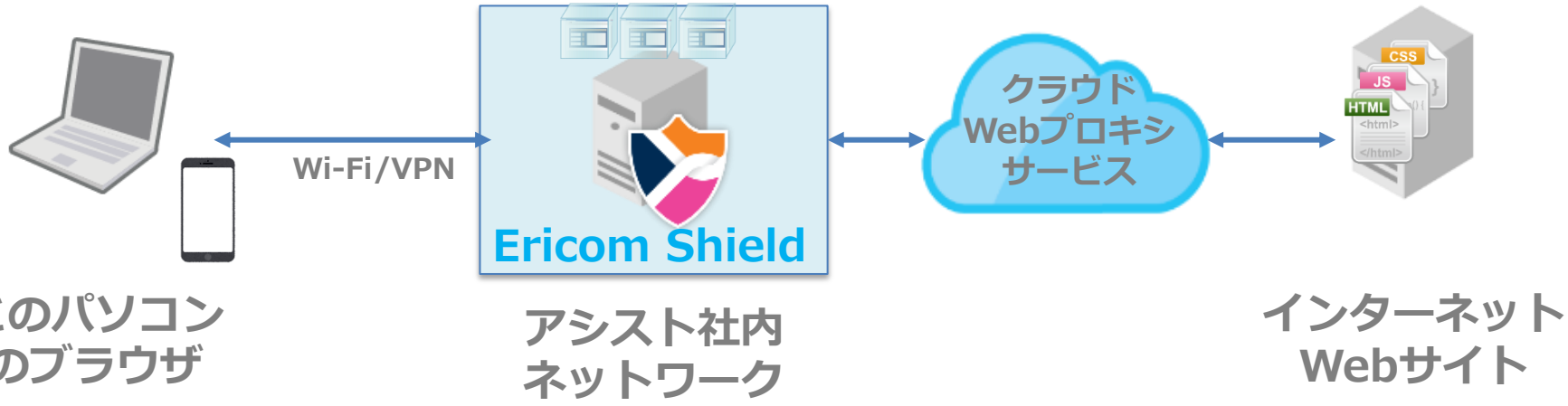


⑫このウィンドウのままhttp://xx/xx.htmlにアクセスせよ

⑦コンテナ上でマウスイベントを実行
⑧DOMノードの変化を検出

⑪DOMノードに「マウスオーバーされたときの文字スタイル」を適用

デモ環境



【セキュリティ強度比較】フレームレンダリング VS クリスタルレンダリング

	フレームレンダリング	クリスタルレンダリング
Web実行における端末のマルウェア感染	インターネット上の一切のコンテンツはコンテナで実行され、端末側には一切入ってこないため、ゼロデイ攻撃、高度な攻撃であったとしても端末側は100%安全といえる。	オリジナルのJavaScriptは無効化され、ページ内の静的要素のみを端末側でレンダリングするため、安全。将来的にブラウザを標的にJavaScript以外の攻撃手法が出現した際は、機能強化で対応。
ダウンロードファイルのサニタイズ	Votiroを利用可能。ただし、実行ファイル等サニタイズできないものあるため、全てが安全になるわけではない。	同左
フィッシング対策	詐欺サイト側に自ら情報を入力してしまえば、情報搾取は免れない。InterSafe WebFilterやEricomのオプションと一緒に利用することで、認識された詐欺サイトへのアクセスは制御可能。	同左

【使い勝手比較】フレームレンダリング VS クリスタルレンダリング

	フレームレンダリング	クリスタルレンダリング
ローカルライクなブラウジング・ナビゲーション	ブラウザから利用でき、ブラウザメニューもすべて有効なので、普段と使い勝手は変わらない。ただし、文字入力、画面スクロールのラグでストレスを感じやすい。	静的なページ表示に必要なコンテンツはすべてローカル側に落ちてきているため、ローカルブラウジングと同様にスムーズにナビゲーションできる。
ローカルブラウジングと同様の機能性	動画・音声再生、コピー＆ペースト、お気に入り、ユーザー辞書、印刷、ファイルのダウンロードとアップロードなど、基本的な機能にすべて対応。	同左
あらゆるWebサイトを閲覧可能か	Linuxのブラウザで閲覧できるサイトであればすべて閲覧可能。 ただし、クライアント証明書や、ローカルストレージを利用する特殊なWebサイトでは利用できない。	左記の制限にプラスして、現時点ではJavaScriptの使われ方によってはナビゲーションができない、表示がされない、といった場合がある。多機能なWebアプリ系（Salesforce等）はまだ苦手。

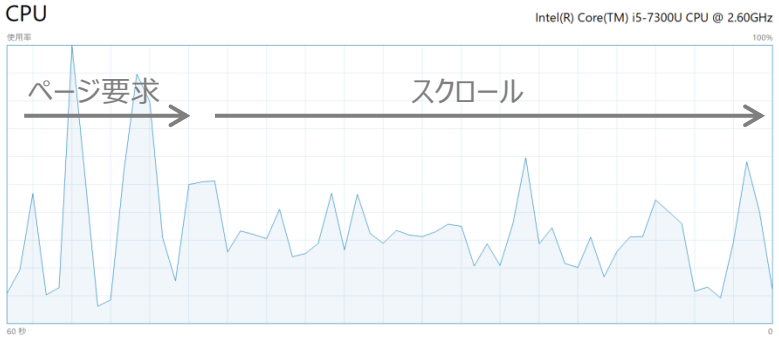
【端末CPU比較】フレームレンダリング VS クリスタルレンダリング

テストシナリオ：日経のWebサイトを表示して、ページの一番下まで1分かけてスクロールを実施

フレームレンダリング

・ページ要求してから表示完了までのタイミングは早い
・静止している時間は通信も処理も発生していないため、CPU使用率は低い
・スクロール時も、単純に画像のやり取りなので、CPU使用率は特別大きくない

クリスタルレンダリング



・ページの表示開始時にCPUを多く消費。ローカルブラウジングと同じリソースの処理傾向
・ページ内のコンテンツを表示仕切る前でもスクロール可能

【端末ネットワーク比較】フレームレンダリング VS クリスタルレンダリング

テストシナリオ：日経のWebサイトを表示して、ページの一番下まで1分かけてスクロールを実施

フレームレンダリング

- **スクロール時平均3Mbps**程度の通信が発生
- 別タブ表示中は、隠れたタブの画面転送は止まる
- モバイル環境で長時間利用するには向かない
- 利用者の数に応じて、アクセスネットワーク、サーバネットワークの増強も検討が必要

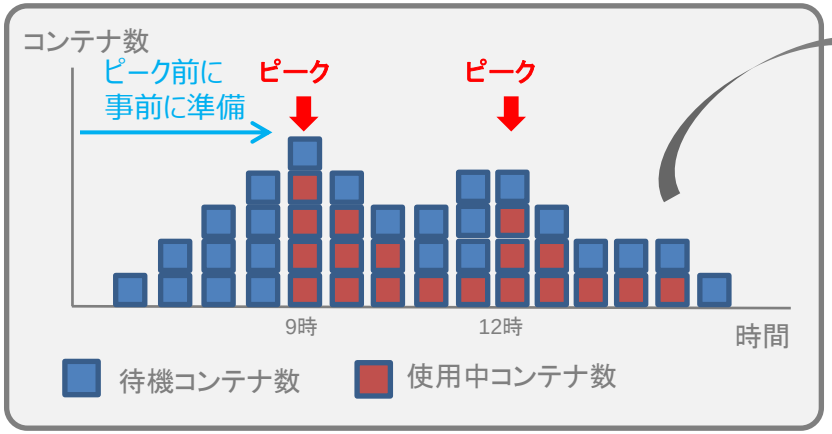
クリスタルレンダリング

- ページ要求時、**MAXで7Mbps程度まで上がった。普通のブラウジングと同じ挙動**
- **スクロール中の帯域使用量は平均150Kbps程度**
- HTMLをダウンロードした後はスクロールや、テキストは読めるため、低帯域・高遅延ネットワークで利用可

【ブラウザサーバのリソース】フレーム＆クリスタル共通

最もサーバCPU負荷がかかるのはコンテナ作成時。ブラウザ閲覧時の負荷は両方とも気にするほどではない。
朝のアクセスピークに合わせてブラウザコンテナ数を増やす機能で準備しておけばオフピークできる。

	作成	待機	利用	破棄
CPU負荷	**高**	低	中	



ピーク時前に待機コンテナ数を増やしてオフピーク
準備せずピーク時にコンテナを大量作成するとスローダウン
の懸念あり

httpリクエストに対して瞬
間的に割り当て

ユーザーがタブを閉じたり、
リロードしたり、ページ内リ
ンクをクリックしたり、
無操作から一定時間を経過す
ると自動的に破棄

【サーバリソース比較】フレームレンダリング VS クリスタルレンダリング

	フレームレンダリング	クリスタルレンダリング
CPU	既定値：0.2CPU～2CPU/コンテナ ページロード時：25～40%程度 スクロール時：15%～25%程度 一部動いているとき：10%程度 安静時：1%	既定値：0.2CPU～2CPU/コンテナ ページロード時：30%～60%程度 スクロール時：15%程度 JavaScriptが多いページ：15%程度 安静時：1%
メモリ	既定値：150～1,000MB/コンテナ ブラウジングに必要な機能だけで使うため、VDIと比べると極めて軽量	同左
ネットワーク	・操作時＆動的ページの閲覧時は平均的に1～3Mbps/コンテナ程度使用するため、ユーザーの数に合わせてアクセス回線とサーバー側ネットワークの増強の検討が必要 ・携帯キャリアのデータプランには向かない。	普通のブラウザと同様な帯域利用なので、今のネットワーク構成を変える必要がない

結果まとめ

大項目		フレーム レンダリング	クリスタル レンダリング
セキュリティ	Web実行におけるマルウェア感染	◎	○
	ファイルのサニタイズ	○	○
	フィッシング対策	△	△
使い勝手	ローカルライクなブラウジング・ナビゲーション	△	○
	ローカルブラウジングと同様の機能性	○	○
	あらゆるWebサイトを閲覧可能か	○	△
ハードウェア（コスト）	CPU	○	○
	メモリ	○	○
	ネットワーク	△	○

結果まとめ

大項目		フレーム レンダリング	クリスタル レンダリング
セキュリティ	Web実行におけるマルウェア感染	◎	○
	ファイルのサニタイズ	○	○
	フィッシング対策	△	△
使い勝手	ローカルライクなブラウジング	○	○
	ローカルブラウジング時の操作性	○	○
	古いWebサイトを閲覧したい	○	△
ハードウェア(コスト)	CPU	○	○
	メモリ	○	○
	ネットワーク	△	○

引き分け

個人的にはクリスタルレンダリングの今後の進化に期待

アシストからの推奨

こんな方にお勧め！ フレームレンダリング

- 自治体、教育委員会のように国から完全分離を義務化、推奨されている分野
- 国家機密、軍事技術開発、先端技術開発に関わっている
- 怪しいサイトを潜入調査するセキュリティリサーチャー

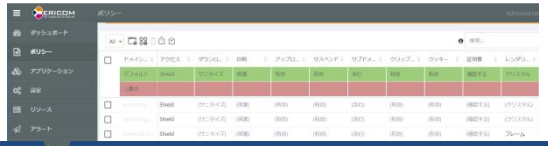
こんな方にお勧め！ クリスタルレンダリング

- Web分離はしたいが、使い勝手を維持したい
- 導入時にオンプレミスのネットワーク構成を変えたくない
- 文字入力するWebアプリの利用
- モバイル端末から携帯キャリアのデータ通信で利用する

一番お勧め！

実はEricom Shieldなら
同じライセンスで使い分けが可能！

WebセキュリティをEricom Shieldで一元管理



どのサイトを	誰に	どう接続させて	どう使わせるか
□ あらゆるサイト □ 特定ドメイン ※オプション機能 □ 特定のURL	□ 全員 □ 特定グループ	□ クリスタルレンダリング □ フレームレンダリング □ ローカルブラウジング □ アクセスブロック	□ ファイルのサニタイズ有無 □ アップロード可否 □ コピー＆ペースト可否 □ 印刷可否　etc

「うちの組織にピッタリなWebセキュリティ」を実現

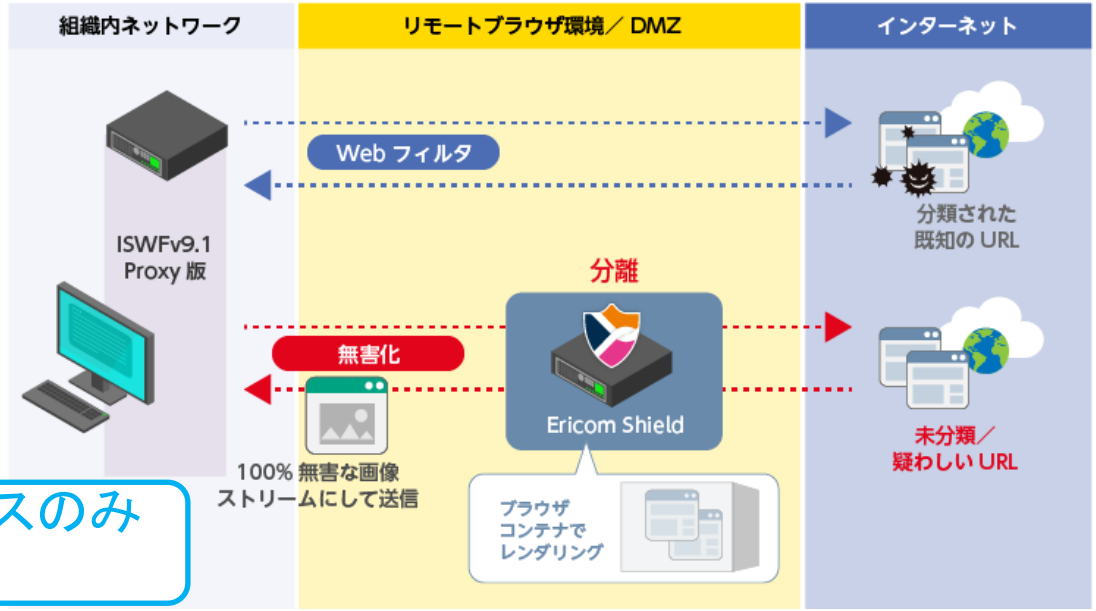
Ericom Shield×Webフィルター

Ericom Shield URL
カテゴリゼーション・オプション

Ericom Shieldのオプション機能。URLカテゴリごとのアクセス制御を**Shield内で定義＆管理**します。

InterSafe WebFilter連携

URLカテゴリごとのアクセス制御は**InterSafe側**で行います



「未分類」カテゴリのURLアクセスのみを分離する低コスト構成

EricomShieldとInterSafe WebFilterの連携イメージ

まとめ

- フレームレンダリング（画面転送）は端末側は100%安全だが、文字入の多いサイトには向かない。ネットワークにも考慮が必要
- クリスタルレンダリング（サニタイズ）の仕組みはオリジナルのJavaScriptをコンテナ側では実行するが、クライアント側では実行させず、DOMの操作命令で結果の見た目を同じにしている
- クリスタルレンダリングの操作感は普通のブラウジングと変わらず、理論上安全。ただし、まだ動かないサイトも多い
- どちらもWebセキュリティとしては十分だが、組織のリスク判断に応じて使い分ければセキュリティと使い勝手を最適化できる

超サポ
愉快カンパニー
アシスト

※本資料に記載している情報は、2019年10月17日現在のものです。
※本資料の内容は、今後予告なく変更されることがあります。
※OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
※文中の社名、商品名等は各社の商標または登録商標である場合があります。