Perusion Payment Server

Mike Heins
Perusion
<mike@perusion.com>

PCI and Interchange

- Problems with PCI
- Interchange and Credit Cards – History
- Lightening the compliance load
- Perusion Payment Server (PPS)
- Configuration and features
- Interface with Interchange
- Futures

The Problem of PCI

- PCI DSS established as consolidation of Visa/MC/Amex/Discover/JCB programs in 2004

The Problem of PCI

- PCI DSS established as consolidation of Visa/MC/Amex/Discover/JCB programs in 2004
- Designed to give issuers, banks, merchants and customers more confidence in credit card security

The Problem of PCI

- PCI DSS established as consolidation of Visa/MC/Amex/Discover/JCB programs in 2004
- Designed to give issuers, banks, merchants and customers more confidence in credit card security
- Compliance is extremely complicated and implementations are open to interpretation

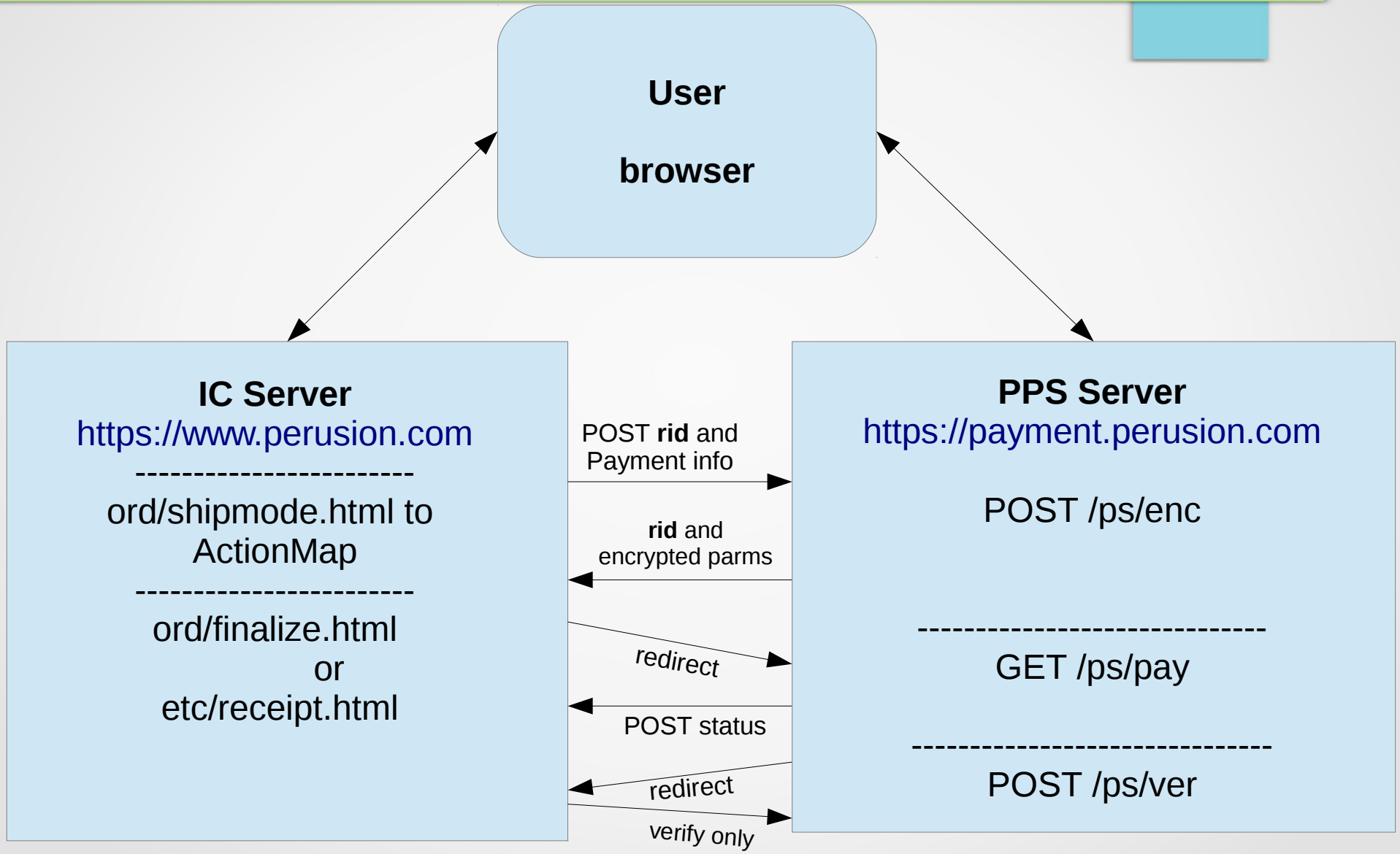
Questionnaires

- Current Interchange needs Self-Assessment Questionnaire (SAQ) C, difficult to impossible for small merchants to comply with letter of “law”
- Scans are mandatory – and expensive and time-consuming
- Handling card data is worrisome
- But Paypal and external payment servers only go so far
- Risk, risk, and more risk when card data goes through your own server
- SAQ A is easy to answer, but not possible with your server handling card data

Perusion Payment Server

- No data or session storage, could theoretically be implemented on read-only file system
- Based on Dancer
- Card interfaces from Business OnlinePayment or modfied Vend::Payment modules
- Direct interface to any web server via Plack or PSGI
- Uses any template engine
- We provide ActionMap routines for Interchange interface
- Easy to have your own **https://pay.yourstore.com/**
- Provided as a service, no release yet

Architecture



Sequence

- Generate reference ID (**rid**)
- IC POSTs **rid** and payment parms to PPS
- PPS returns encrypted parms and **rid** to IC
- IC redirects to PPS with **rid** and parms
- PPS collects and submits card data
- PPS POSTs status to IC
- IC either returns receipt or asks for finalize order

Advantages

- No cardholder data on IC server
- SSL server probably should be used, but not totally necessary
- No risk of trojan intermediary
- Complete sequestration of payment data
- Cardholder data not stored anywhere, only sent to user and processor

Dancer routes

- POST /ps/enc
  - Receives **rid** and name, address, zip, total, nitems, whatever needed
  - Encrypts parms with Crypt::CBC (Blowfish) and secret based on vendor, returns to IC
- GET /ps/pay
  - Receives **rid** and encrypted parms
  - Presents payment page
- POST /ps/pay
  - Receives **rid**, parms, and card data
  - Processes card

Dancer routes

- GET /ps/ver
  - Receives **rid** and card MD5
  - Presents verify page
- POST /ps/ver
  - Receives **rid**, card MD5, card number
  - POSTs status with RID back to IC Server

The accounts.yml file

```
---
perusion:
  config_name: perusion
  enc_params:
    - rid
    - fname
    - lname
    - company
    - address1
    - address2
    - city
    - state
    - zip
    - nitems
    - subtotal
    - shipping
    - amount
    - total_cost
    - salestax
  pay_opt:
    id: 4xxxxxxxxx
    secret: XAWXAWaaaaaaaaW
    payment_module: Vend::Payment::AuthorizeNet
    payment_routine: authorizenet
```

The accounts.yml file

```
---
perusion:
  .....
  mv_pays: Live_Perusion_Key
  pay_view: perusionnew.tt
  postback_jump: https://www.perusion.com/c/public/ord/final.html
  postback_url: https://www.perusion.com/c/public/postback
  cvv_info_url: https://www.perusion.com/c/public/ord/ccsecure.html
  timeout_url: https://www.perusion.com/c/public/ord/shipmode
  cancel_url: https://www.perusion.com/c/public/ord/error
  page_title: Perusion Payment Page
  company: Perusion
  logo: /pci/images/header1k.jpg
  menu: /home/perusion/www/pci/include/menu
  content: /home/perusion/www/pci/include/content
  footer: /home/perusion/www/pci/include/footer
  css_dir: /pci/css
  js_dir: /pci/js
  jquery_latest: //ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js
```

ActionMap routines

- payment.am
  - Assembles parms for sending to PPS
  - creates payment table record (**rid**)
  - POSTs to PPS
  - Redirects to PPS
- postback.pm
  - Accepts payment status, **refid**, card markers
  - Preps or submits order

Payment table

```
CREATE TABLE payment (
  refid int(11) NOT NULL auto_increment,
  session varchar(255),
  username varchar(128),
  tid varchar(128),
  date_created timestamp NOT NULL default CURRENT_TIMESTAMP,
  postdata text,
  postback char(1) default '0',
  token varchar(32),
  card_ref varchar(16),
  card_type varchar(16),
  card_exp varchar(16),
  error_code varchar(32),
  error_message varchar(255),
  card_sum varchar(48),
  csc_status varchar(255),
  avs_status varchar(255),
  authorized_amount varchar(20),
  PRIMARY KEY  (refid),
  KEY payment_refid (refid),
  KEY payment_tid (tid)) ;
```

Summary

- PPS allows merchant to answer Self-Assessment Questionnaire A, no cardholder data present
- Allows interface with existing IC-ported gateways
- Flexible template engines via Dancer
- Built for Interchange, could be used for any cart
- Software as service, no maintenance or server required, PCI SAQ C work pushed to Perusion
- No scanning service necessary
- Security risk devolves to Perusion