Detecting Ponzi and Pyramid Business Schemes in Choreographed Web Services

Murat Gunestas, Murad Mehmet, Duminda Wijesekera

Chapter 10

DETECTING PONZI AND PYRAMID BUSINESS SCHEMES IN CHOREOGRAPHED WEB SERVICES

Murat Gunestas, Murad Mehmet and Duminda Wijesekera

Abstract Businesses are increasingly using web service choreographies to implement dynamic service invocations and content specific operations. These web service choreographies can be misused at multiple levels – by exploiting their technical capabilities and using them to design complex illegal business schemes such as Ponzi, pyramid and money laundering schemes. One of the main problems with the illegal schemes is that they are similar to legal multistage business schemes; their illegality is apparent only to a macroscopic observer. This paper describes some of these schemes and demonstrates how to obtain evidence pertaining to the schemes using cryptographically-secure local message repositories. The evidence gathered is of considerable value to financial fraud investigators, business arbiters, potential investors and judicial actors.

Keywords: Web services, choreographies, Ponzi schemes, pyramid schemes

1. Introduction

Businesses are increasingly invoking dynamic services and generating content-specific operations among choreographed web services, thereby creating service interdependencies between web services. These dynamic service interdependencies can be exploited to create new misuse activities. Some exploit the infrastructural dependencies of the services themselves while others use the infrastructural dependencies to create illegal business schemes. This paper focuses on detecting Ponzi and pyramid investment schemes created to defraud unsuspecting investors.

Illegal business schemes are difficult to detect because they are similar to legal business schemes at a microscopic level and are apparent only at the macroscopic level. Thus, they can elude local monitoring of web

Table 1. Pyramid scheme.

Level	**Payments of \$400 (#)**		
1 (\$100 × 3 = \$300)	1 × #	1 × #	1 × #
2 (\$30 × 9 = \$270)	3 × #	3 × #	3 × #
3 (\$30 × 27 = \$810)	9 × #	9 × #	9 × #
4 (\$30 × 81 = \$2,430)	27 × #	27 × #	27 × #
...	...	...	...
21	10,460,353,203 × #		

transactions. Ponzi and pyramid schemes [13, 14] are difficult to differentiate from multilevel marketing schemes that either run their own businesses or invest in others. The basic dynamic of these two schemes is to rob Peter to pay Paul [16].

The Ponzi scheme is named after Charles Ponzi [16]. For many years, he collected money and promised returns within 90 days to investors who enrolled other investors in the scheme. Ponzi paid out the early investors using funds invested by late joiners. Other than running this scheme, Ponzi neither ran a business nor invested in other businesses and, consequently, neither incurred a profit nor a loss.

Numerous Internet-based Ponzi schemes are currently being investigated [9, 10]. In 2006 alone, 25,000 web sites suspected of running Ponzi schemes were shut down by the U.S. Securities and Exchange Commission [7].

A classic pyramid scheme, shown in Table 1, also uses the same principle. The orchestrator who originates the scheme promises top-level investors large returns on their investments, as do the recruited investors to their potential recruits. The example in Table 1 is organized as a four-level payment scheme with a span of three, i.e., only up to four levels of ancestral recruiters profit from investments, and each recruit at every level recruits three others. A Level 1 investor recruits three others and receives \$100 per recruit. Each recruit is expected to recruit three others, thereby building a recruit tree. The Level 1 investor is paid \$30 (not \$100) per recruit at Levels 2, 3 and 4; and does not profit from investors beyond Level 4.

The orchestrator considers the recruiting activity to be complete when he receives \$400 from an investor. Therefore, the orchestrator pays his recruiters \$100 for the first parent or \$30 for the three immediate ancestral recruiters, paying at most \$190 on an investment of \$400. Consequently, an investor makes \$3,810 (\$2,430 + \$810 + \$270 + \$300) on his investment of \$400 to his promoter if he successfully recruits three other investors and they successfully recruit three other investors all the

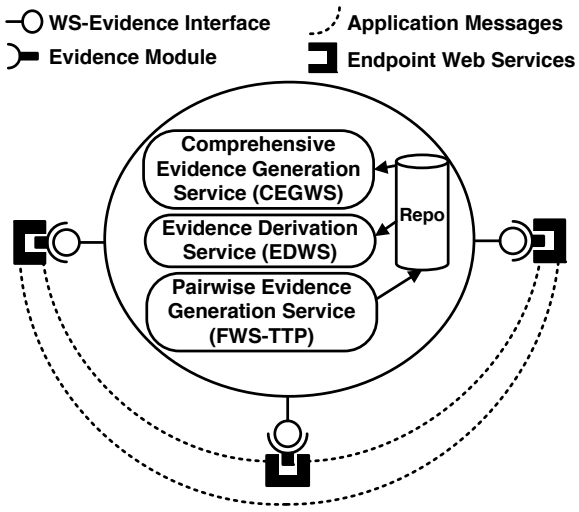


Figure 1. Evidence generation framework.

way down to Level 4. The scheme grows quickly, but is unsustainable in the long term and many promoters lose their investments because of the difficulty of attracting new recruits [14].

Financial institutions and their business partners are increasingly engaging service-oriented architectures such as dynamic brokering, creating dynamic service interdependencies that can be exploited to orchestrate illegal business practices. In order to detect such illegal schemes, it is necessary to have a comprehensive non-repudiable perspective of complex multi-party transaction models. A multi-party communication can arise in two possible ways among dynamic web services. In the first, a static communication pattern is specified and all participants follow this previously-known pattern. In the second, web services discover and transact with other services, dynamically creating choreographies that were unknown *a priori*. Consequently, in order to discover illegal activity, it is necessary to identify illegal business transactions using both types of choreographies.

2. Evidence Generation Framework

This section briefly reviews the evidence generation framework (EGF) of Gunestas, *et al.* [2], which is used to track business-level choreographic misuse. The EGF has three layers (Figure 1). The pairwise evidence generation service (Level 1) generates evidence for pairwise interactions between web services. The evidence derivation service (Layer 2) derives

facts from available pairwise evidence in order to refute or justify claims of agreement violations between communicating partner services. The comprehensive evidence generation service (Layer 3) generates instances of requested choreographies from Layer 2 and Layer 3 data.

The EGF provides online evidence generation and management capabilities to other web services as a web service itself. In order to use the EGF, other web services (called member services) should integrate the EGF with themselves using a centralized service access point. Thereafter, the EGF acts as a trusted third party. As a service, the EGF receives and retains service requests and responds in a cryptographically-secure manner, retains the correspondence in secure repositories, and provides the correspondence for dispute resolution and forensic investigations. The EGF provides "evidence adapters" for all requests.

Gunestas, *et al.* [2] have constructed a prototype implementation of Layer 1 and several protocols based on one-way and request-response message exchange patterns. Currently, the EGF provides evidence for non-repudiation, fairness and timeliness using digital signatures to provide proofs of receipt and delivery, to link a message to its creator/sender and to provide message integrity. For accountability, the EGF uses fair non-repudiation mechanisms that utilize trusted third parties (TTPs). Some fair exchange protocols (e.g., [6]) do not use TTPs because they assume that the participants have prior knowledge of the message contents; but they are not used in the EGF because web services may not always know the expected message content. Timeliness is required because of the time-sensitive nature of most business transactions. Evidence records are based on the time observed at TTPs. EGF servers gather pairwise transactional evidence that flows between sender and receiver web services, employing inline TTPs that use the Simple Evidence Layer Protocol (SELP) or offline TTPs that use the Optimistic Evidence Layer Protocol (OELP) [4]. SELP and OELP are used by end-points to obtain non-repudiable evidence by engaging a specific message format and digital signatures.

3. Evidence of Observed Interactions

Web services use many kinds of messages (e.g., one-way messages and request-response messages) in order to choreograph business processes between themselves that correspond to the four proposed WSDL operation types (in-only, out-only, in-out and out-in). An external observer who is ignorant of the business processes can observe only the one-way and request-response message exchange patterns that are formalized in Definition 1.

Definition 1 (Message): A web services message consists of three components: (i) mandatory fields corresponding to the sender, receiver and time, where the first two are URLs and the last is chosen from a set T; (ii) optional fields corresponding to a finite set of attributes from a set A; and (iii) message content consisting of strings from an alphanumeric set C.

In this paper, the symbol | denotes string concatenation and $encA(r)$ denotes the string obtained by encrypting r using A's key. Furthermore, if m is a message, then $m.a$ denotes the value of its attribute a. For example, $m.time$ is the value of the timestamp of m.

Definition 1 establishes the notation for describing the messages used to extract knowledge about externally-observable facts pertaining to choreographies. Because different choreographic specifications may select different labels for their identifier fields, in order to address naming convention problems, we use XPATH expressions to specify ID values. Furthermore, because any fabricator can produce messages, we rely on cryptographically-secure messages to ascertain reliable evidence. The messages are collected to derive "evidence objectives" – claims that are to be substantiated or refuted using the collected evidence, e.g., message origin, message properties or the intended recipient. This evidence is generated from cryptographically-secure messages. However, certain objectives such as evidence of delivery and evidence of non-availability may require messages to be signed by a TTP.

Definition 2 (Primitive Evidence Objectives): The three primitive evidence objectives are: (i) Evidence of Origin: message m with origin A and content $r|sigA(r)$ from A to B is said to provide evidence of origin; (ii) Evidence of Delivery: message m with content $ack|sigTTP(ack|m)$, where TTP is a trusted third party or content $ack|sigB(ack|m)$ and B is $m.recipient$ is said to provide evidence of delivery; and (iii) Message Evidence: evidence of a message m is a pair (m_1, m_2), where m_1 is the evidence of origin and m_2 is the evidence of delivery of m.

According to Definition 2, cryptographic evidence is required from a web service of a trusted third party for claims of origin and delivery. Interested readers are referred to [3, 4] for details about how the evidence of origin and evidence of delivery can be collected at TTPs using WS-Evidence messages that are generated via non-repudiation protocols.

The message evidence (ME) is stored in the form of log records in the EGF as described in [3]. Because log records may contain large volumes of data, message evidence indices (MEIs) are used to refer to messages. Table 2 shows a sample index table, where the first column is an index for stored packets that have the attributes of time, sender, message string

Table 2. Sample MEI table.

ID	Time	Sender	Receiver	Msg	Content
63..	21	A	B	r	<..invID..>
67..	22	B	C	m	".."
68..	23	C	B	k	<..payID..>

and content. The first reference in the table is to a message sent by A to B with content <invID>.

4. Evidence of Global Misuse

Mining choreographies that are created due to message content from external observations require linkage parameters, which can be derived from externally-invisible message content; this renders them not very helpful to external monitors and auditors. One opportunity to obtain this information is when a victim makes a complaint. Thus, the following method can be used to detect a Ponzi-like scheme:

- Accept a victim complaint.
- Examine the content of specimen records involving promotion or investment messages provided by the victim.
- Determine the parameters in the evidence that can be linked.
- Detect choreographies and design them in the dynamics of the algorithms.
- Create the algorithms.
- Run the algorithms in the appropriate order, possibly running more than one algorithm when required.
- Collect a comprehensive set of evidence to determine if the scheme is illegal and its effect on the network.
- Broadcast an alert to current and potential victims.

The method described above works for hierarchical schemes. Different methods would be applied to other schemes. We apply our heuristic method to specify several algorithms for Ponzi-like schemes.

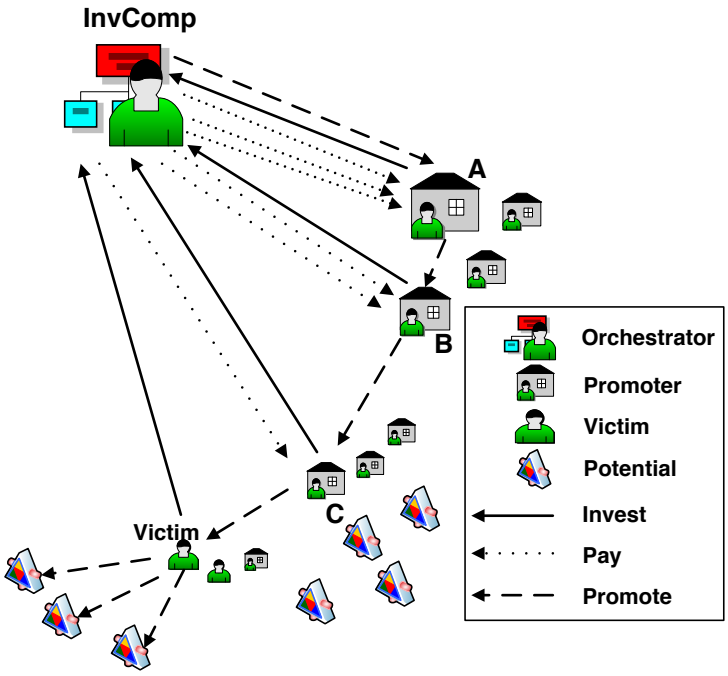


Figure 2. Ponzi-like recruits over web services.

4.1 Ponzi Schemes over Web Services

Typical Ponzi-like schemes have three types of actors: a malicious investment service acting as the orchestrator and investor services acting as promoters or victims (depending upon their investments and return rates). Figure 2 illustrates how these actors collaborate to spread the financial scheme over a network of web services. The investment company, InvComp (Orchestrator), promotes A (Promoter) to recruit B into its scheme by promising a quick return on investment and encourages B to promote the scheme to other potential investors. This promoting activity (using promote messages) may not necessarily be observed in the records because promoters may choose other means to convince investors. If A invests (using an invest message) in InvComp, then we say that A has been recruited. A now starts promoting InvComp to other investors in order to get a quick return on his investment and in the process recruits B. We recognize that B has been promoted by A because of the reference value in the content field of the invest message sent by B to InvComp. In accordance with the return policy of the scheme, InvComp makes a payment to A (pay message).

Table 3. MEI tuples featuring a misuse scheme.

ID	Time	Sender	Receiver	Msg	Content
...					
	45	B	InvComp	invest	Promoter=A
	55	InvComp	A	pay	150
...					
	67	C	InvComp	invest	Promoter=B
	76	InvComp	B	pay	150
	78	InvComp	A	pay	30
...					
	87	Victim	InvComp	invest	Promoter=C
	89	InvComp	C	pay	150
	92	InvComp	B	pay	30
	104	InvComp	A	pay	30

The choreographies between the investment company, recruiters and recruits spread in an investor web service network. When a recruit cannot attract enough investors, he loses his money (Victim). Figure 2 shows the complete and incomplete recruit choreographies.

We do not assume that we know the global scheme when mining. Instead, we assume that either a promoter web service identity or an invest message is submitted to a law enforcement agency by a victim. Following the heuristic method presented above, it is possible to find invest, pay and promote messages that contain attributes that refer to each other, demonstrating the collaboration in a pervasive manner.

4.2 Pattern Discovery

This section shows how to discover the patterns that help create comprehensive evidence of illegal business schemes. Following the heuristic method described above, we assume that a victim brings an invest message that contains a promoter web service. An algorithm executed on the evidence repository could show that the promoter web service has received pay messages from other web services and has sent an invest message to the same web service of the alleged investment company. Table 3 shows sample records corresponding to the misuse scheme.

The records in Table 3 show the pattern that keeps the fraudulent activity alive – invest messages are linked by sender header fields and promoter content fields. In other words, every promoter web service in an invest message is paid right after the invest message. The message

Table 4. Ponzi scheme with fanout 1 and depth 1.

I	invest;p pay where invest.sender=A and invest.reciever=B and pay.sender=B and pay.reciever=C and invest.prometer=pay.reciever
II	invest1;p invest2 where invest1.sender=C and invest1.receiver=B and invest2.sender=A and invest2.reciever=B and invest2.promoter= invest1.sender

evidence can be correlated to conclude that the promoter, victim and orchestrator may be involved in a hidden recruit choreography.

In cases where the promoting activity does not involve a web service message, the message or choreography patterns can be included as part of the recruiting activity. We call this content-based choreography "recruit." Pattern I in Table 4 is the signature for the "rob Peter to pay Paul" activity. Pattern II is used as the link between the recruiter and recruit, enabling the mining of recruit paths to create recruit trees from MEI records. For simplicity, we define patterns succinctly; however, more complex patterns may include pay messages, which increases the complexity of queries.

Table 3 shows that investors assume a promoter role within a subsequent recruit choreography pattern, thus creating a recursive investment scheme. For example, note that Investor B sends an invest message to InvComp at Time 45. At Time 67, Investor C makes a reference to B through his investment message and B receives a payment from InvComp some time later (pay message at Time 76). The same choreography can be observed between C and other subsequent investors, revealing that they recruited other investors shown in later records. Consequently, in order to detect such a scheme, it is necessary to recognize the recursive investment and payback schemes. The recursive scheme creates a recruit tree that joins a recruiter to all his recruits. By traversing a path in a recruit tree from a chosen (victim) node to the root of the tree (say recruit paths), it is possible to identify the orchestrator. The path Victim-C-B-A in Figure 2 is such a recruit path.

In this paper, we use the following notation to specify recruit trees formally. Let k be an integer that denotes the fanout of the recruit tree. Then, all finite sequences of $\{0, \ldots, k-1\}$ are used as identifiers for web service nodes. We use the notation $2^{k<w}$ to denote the set of finite subsequences of $\{0, \ldots, k-1\}$. For example, all binary sequences can be used to index trees with fanout 2, where the left child of node x_p is x_{p0} and the right child is x_{p1}, where p is a finite sequence of integers $\{0,1\}$, i.e. $2^{<w}$. The notation p<q denotes that p is a subsequence of

q where $p, q \in k^{<w}$. The length of $p \in k^{<w}$ is denoted by |p|. Now suppose $p \in k^{<w}$ where $|p| = m$ and $p = <p_o, \ldots, p_{m-1}>$. Then, the i^{th} ancestors of p for $i \geq 1$ are given by ancestor(i)=$<p_o, \ldots, p_{m-i}>$. Finally, ϕ denotes the empty string in $k^{<w}$.

Definition 3 (Recruit trees of fanout k and depth m): Suppose I is an investment company web service. Inductively define active(n) for every integer n as follows:
(i) Active(0)={m} where m is defined as a message where [m.sender=p_o, m.receiver=I].
(ii) Suppose active(n) has been defined and $p \in k^{<w}$ with $|p| = n$, then for each $i \in \{0, \ldots, k-1\}$ define [active($p\hat{}i$)=$msg_{,p}$PayBack($p\hat{}i$)] where msg satisfies [msg.sender=$P_{p\hat{}i}$, msg.reciever=I and msg.content= invest] and PayBack($p\hat{}i$) satisfies PayBack($p\hat{}i$)=$msg_1 \cap_p \ldots \cap_p msg_m$ where every msg_i is of the form [msg_i.content=pay, msg_i.sender=I, msg_i.receiver=ancestor(p,i)] for $i = l$.
(iii) Let active($n+1$)=active($p\hat{}0$) $\cup_p \ldots \cup_p$ active($p\hat{}(n-1)$).
(iv) Define a recruit tree to be active* = LFP(f, m, E) where function f is defined in items (i) and (ii); the message m is defined in item (i); and the set of message equations E is defined in items (i) and (ii).

We denote the class of Ponzi schemes of fanout k and depth l and attribute equations E as Ponzi(k, l, E), where E is the collection of equations in Definition 3. Definition 3 provides a generic definition for Ponzi-like schemes where the number of recruits employed by any recruiter is limited to an integer k and the number of ancestors deriving a payback from the recruitment is at most an integer l.

The web service nodes are numbered by strings from $\{0, \ldots, k-1\}$, resulting in trees where every node has at most n levels. Thus, the parameter p in Definition 3 denotes a path with $|p|$ elements in such a tree. Step 0 with the empty string ϕ represents the recruiter in item (i) in Definition 3. Item (ii) assumes that the tree is defined up to a path p of length n and finds its next level. This step is the sequential composition of two steps. First, P_p sends messages to each of its children to invest. Next, each of these children invest in I, followed by I paying the ancestors of these children. The ancestors who are paid back are limited to at most l generations. Item (iii) in Definition 3 collects all possible paths that extend the tree to the next level $n+1$. Item (iv) collects all the sub-trees with depth $n+1$.

5. Detecting Global Misuse

This section discusses how to mine global misuse instances of given patterns from log records of observed web transactions using Stream-

Algorithm 1 Detecting Recruits of Ponzi Schemes

INPUT: MEI tuples
OUTPUT: Ponzi-like recruit MEI pairs
DESCRIPTION: Glides over MEIs using a window size of 3 to detect Pattern I along with the predicates specified in the WHERE phrase

```
 1 CreateInputStream MEI ($MEI schema) ;
 2 CreateOutputStream PonziDetectOut ;
 3 CreateStream InvestFilterOut ;
 4 CreateStream PayFilterOut ;
 5 SELECT * FROM MEI
 6 WHERE msg==“invest” INTO InvestFilterOut
 7 WHERE msg==“pay” INTO PayFilterOut
 8 SELECT “Ponzi-like recruit” AS detected, invest.time AS investTime,
   pay.time AS payTime, pay.receiver AS recruiter, invest.sender AS recruit
 9 FROM PATTERN (InvestFilterOut AS invest THEN PayFilterOut AS pay)
10 WITHIN 3 (days) ON time
11 WHERE invest.receiver==pay.sender AND regexmatch
   (“.*”+“promoter=”+pay.receiver+“.*”, invest.content)
12 INTO PonziDetectOut ;
```

SQL [12] and a StreamBase platform [11]. StreamSQL is an event pattern language that can be used to define queries over streams of data. StreamBase is an event processing platform that can run StreamSQL queries over input source file or database and produce outputs.

StreamSQL has several commands. CREATE INPUT STREAM creates data streams from a named file that is pre-configured according to a known schema. CREATE OUTPUT STREAM creates an output stream that is pre-configured according to a schema. A PATTERN phrase defines the search criteria from multiple input streams. A WITHIN phrase creates the maximum size of a window that moves along a collection of aligned streams when searching for a pattern.

Algorithm 1, which is based on StreamSQL, discovers Pattern I defined in Table 4. The algorithm accepts MEI records in ascending order of timestamps. The algorithm processes the pattern by filtering the records into two groups, invest and pay, using the predicates defined in Lines 6 and 7. This enables the pattern to employ the appropriate template (THEN phrase) in Line 9, i.e., invest messages are expected before pay messages. The predicates defined in Line 11 stipulate that the receiver of the invest message should be the sender of the following pay message; and the promoter value in the content of the invest message should be the receiver of the following pay message. The window size is set to 3 in Line 10. The SELECT phrase gathers the re-

Algorithm 2 Enhancing Ponzi Detection

INPUT: PonziDetectOut from detectRecruits
OUTPUT: Ponzi alerts
DESCRIPTION: Counts detected Ponzi-like recruits using a window size of 6 as the minimum support. Emits Ponzi alerts when the minimum support is reached

```
1 SELECT "Ponzi Alerts," count() AS minSup
2 FROM PonziDetectOut [SIZE 6 TUPLES]
3 INTO PonziAlerts ;
```

quired information about the detected pattern and emits the result to the PonziDetectOut table.

Detecting a few Ponzi-like recruits may not be sufficient to declare that a Ponzi scheme exists. To increase the confidence, a minimum support value is defined as a threshold; and an alert is sent only when the threshold is exceeded. Algorithm 2 can be employed to strengthen Algorithm 1 by incorporating a predefined minimum support value.

Algorithm 2 sends an alert when at least six Ponzi-like recruits are detected in the output of Algorithm 1. Note that the addition of Algorithm 2 to Algorithm 1 decreases the number of false positives.

6. Generating Comprehensive Evidence

This section discusses how to detect the orchestrator or the earliest known recruiter of a Ponzi scheme by tracing a potential recruit path to its root. Next, all possible paths that originate at the detected recruiter are examined to identify others who have invested in the Ponzi scheme. We begin by defining a choreography.

Suppose that the complaint includes an invest message msg and that the investment company I is used by the participants in the scheme. We define an ancestorChain(n) as:

(i) ancestorChain(0) = msg

(ii) ancestorChain($n+1$) = [$msg_{l;p}$ ($msg_{1;p}$ k_1) $\cup_p$ $\ldots$ $\cup_p$ ($msg_{l-1;p}$ k_{l-1}) $_{;p}$ ancestor(ancestorChain(n),n)]

satisfying the equations E:

msg_1.content=pay and k_1.content=invest, $\ldots$,

msg_{l-1}.content=pay and k_{l-1}.content=invest

(msg_l.time < msg_l.time < k_1.time), $\ldots$,

(msg_1.time < msg_{l-1}.time > k_{l-1}.time) and msg_l.reciever=I.

Also, we define Earliest(msg) as LFP(f, msg, E).

As a special case, we show how to compute the ancestor chain corresponding to Pattern II in Table 4 using StreamSQL in Algorithm 3. Given a recruiter, the algorithm traces ancestor recruiters to find the

Algorithm 3 Computing the Orchestrator

INPUT: Promoter $P, MEI tuples
OUTPUT: Ancestor chain of promoters as RecruitPathOut
DESCRIPTION: Given the promoter, traces back the MEI records and finds the path and the distance to/from the orchestrator using Pattern II

```
 1 CreateInputStream MEI ($MEI schema) ;
 2 CreateOutputStream RecruitPathOut(
   $MEI schema, newPromoter String) ;
 3 CreateStream LocalStream ;
 4 DECLARE pointerPromoter String DEFAULT $ PUPDATE FROM
 5 SELECT newPromoter AS pointerPromoter FROM RecruitPathOut ;
 6 SELECT * FROM MEI
 7 WHERE msg==“invest” AND receiver==“O”
   AND sender==pointerPromoter
 8 INTO LocalStream
 9 SELECT time, sender, receiver, msg, content,
    GetXPATHValue(content,“../promoter/”) AS newPromoter
10 FROM LocalStream
11 INTO RecruitPathOut ;
```

orchestrator. It identifies when the scheme began by traversing records in descending order of timestamps looking for the senders of invest messages. This is done by declaring the dynamic variable pointerPromoter in Line 4 to which the suspected promoter is passed by default (see DEFAULT) as the orchestrator. Each time the output emits a hop (that meets the criterion in Line 7: invest message sent to the orchestrator) by the node to the receiver, the promoter value is extracted from the content of the invest message using a XPATH function (SELECT phrase in Line 9). It is then written to the output stream (Line 2) and assigned to the dynamic variable pointerPromoter in Line 5. The newly-assigned value is used as the predicate in Line 7 for locating the next message if it matches the sender value.

After the orchestrator or earliest recruits have been identified, a trace-forward algorithm is used to generate the evidence. Since the algorithm above locates one of the oldest message records for the trace-forward algorithm, it helps gather the most comprehensive evidence of the scheme.

Algorithm 4 compiles the evidence using Pattern II in Table 4. The algorithm accepts MEI tuples. The first SELECT phrase in Line 4 is a filter with predicates dealing with invest messages that are sent to the suspected orchestrator “O.” Pattern II is defined after the PATTERN phrase. The PATTERN phrase duplicates the invest MEIs so that it can apply the appropriate template (THEN phrase) and predicates (WHERE phrase in Line 8) between messages. The algorithm uses

Algorithm 4 Generate Recruit Tree

INPUT: MEI tuples
OUTPUT: RecruitsOut table leading to recruiter->recruit tree structure
DESCRIPTION: Tracing forward the MEIs, outputs an appropriate table representing a tree-view of the scheme using Pattern II

```
1 CreateInputStream MEI ($MEI schema) ;
2 CreateOutputStream RecruitsOut ;
3 CreateStream InvestFilterOut ;
4 SELECT * FROM MEI
5 WHERE msg==“invest” AND receiver==“O” INTO InvestFilterOut ;
6 SELECT recruitee.time AS recruitTime, recruiter.sender
  AS recruiter, recruitee.sender AS recruit
7 FROM PATTERN (InvestFilterOut AS recruiter THEN
  InvestFilterOut AS recruitee) WITHIN 6 (days) ON time
8 WHERE recruiter.sender==
  GetXPATHValue(recruitee.content,“../promoter/”)
9 INTO RecruitsOut ;
```

a window of size 6, limiting it to finding patterns within the specified period. Queries with narrower windows may cause the algorithm to miss more correlations than queries with wider windows.

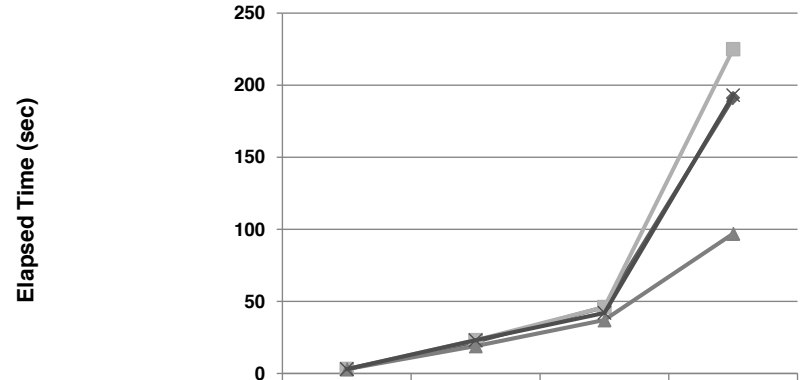
7. Experimental Validation

The three-layered evidence generation framework provides evidence of web service misuse. The services in the bottom and middle layers have been studied by others [1, 5, 8]. Therefore, we validate the top layer of the framework for which have introduced novel queries.

Table 5. Query-pattern mapping.

Query Name	Pattern	Pattern Name
DetectRecruits	invest;p pay	invest-pay
ClimbRecruitPath	invest1;p invest2	invest-invest
GenerateRecruitTree	invest1;p invest2	invest-invest

Table 5 summarizes the queries and their associated patterns along with the names we have used throughout this paper. In order to test their accuracy and performance rates, we generated synthetic data and used a special simulation platform described below.



	MEI-I	MEI-X	MEI-XX	MEI-L
GenerateCHOR-Investing	3	22	46	191
DetectRecruits	3	23	46	225
ClimbRecruitPath	3	19	37	97
GenerateRecruitTree	3	23	42	193

Figure 3. Capacity values of test data.

7.1 Data Characteristics

To our knowledge, our illegal business scheme is novel and we were unable to obtain real data featuring the scheme. Consequently, we generated synthetic data in the MEI format called MEI-I that conforms with the Ponzi/pyramid schemes described above. MEI-I contains a total of 193,827 records (one record/second) over a period of three days (January 19, 2006 to January 21, 2006). The MEI-I file has a comma separated values (CSV) format. We focused on the capacity rate of data that embodies the illegal business scheme in its records.

To evaluate the performance, we created three more data sets using our seed data. The MEI-X data set contains the same Ponzi malicious activity, but is ten times larger than MEI-I in terms of the number of records. MEI-XX and MEI-L are two other sets that are 20 and 50 times larger than MEI-I, respectively (Figure 3).

7.2 Test Environment

As mentioned above, we defined use/misuse patterns using StreamSQL and employed the StreamBase platform to detect the patterns. We used the specially-generated data described above in the MEI structure. The feed simulation platform of StreamBase was adjusted to accept this data in the form of CSV files. The platform empowers users to run StreamSQLs over any user-defined file satisfying the data schema ex-

Table 6. Test environment

Hardware	Software
CPU: Intel Core2 T7400 2.16 GHz,	OS: Windows XP SP2
4 MB L2 Cache, 667 MHz FSB	JVM : SUN JDK 1.5.0.15
Physical Memory: 2 GB, 995 MHz	StreamBase Studio: Version 6.4
Hard Disk: 250 GB, 7200 rpm	Max Heap Size: 1024 MB

pected by the algorithm. The platform also provides observable outputs of algorithm execution. Using the StreamBase Manager, it is also possible to observe CPU and memory usage during algorithm execution. Although StreamBase encourages the use of enterprise servers for benchmarking and improved performance, we observed that the feed simulation platform was adequate to test our queries over vast amounts of data with a reasonable resource allocation rate. Table 6 lists the hardware and software components of the test environment.

7.3 Test Results

The performance of the algorithms was tested by executing three major queries (Table 5) over data sets of different sizes (Figure 4). As mentioned earlier, we observed that maximum proximity rates give the best decisions in the tests. Therefore, we built our performance tests using these rates in the algorithms.

Figure 8 presents the test results. Note that queries for detecting recruits and generating recruit trees take more time than queries for locating the orchestrator. The algorithm execution times for detecting Ponzi-like recruits are greater than the execution times for the other algorithms over the largest data set MEI-L. The recruit path climbing algorithm, which traverses records in the backward direction, exhibits the best performance and there is no need to use a larger window size.

We used the StreamBase Studio Feed Simulation platform in our validation tests. Other higher performance platforms are available, but we obtained reasonable performance despite using an integrated development environment instead of an enterprise environment. We also observed that the queries yield results with reasonable accuracy for the synthetic data sets. These results were obtained because we determined a reasonable window size for window-based queries for each data set. Also, we converged the evidence outcomes by tuning the time, property and key-based patterns associated with the queries.

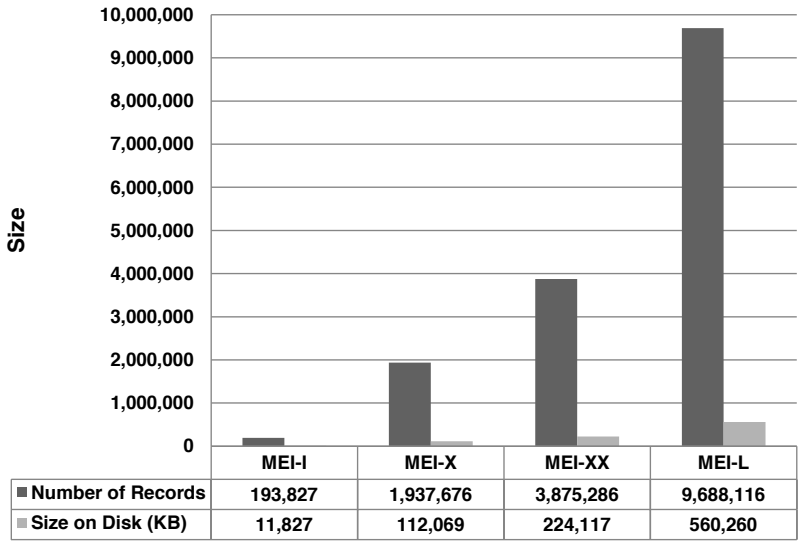


Figure 4. Performance test results.

8. Conclusions

Web service choreographies can be used as the foundation for detecting illegal business schemes. Our implemented system engages StreamSQL to define the associated use/misuse patterns and the StreamBase platform to detect the patterns in large streams of data. Although our choreographies only specify Ponzi-like schemes, the method can be used to specify (and detect) other illegal business activities [15] that can be mined from financial transaction repositories. Our future work will attempt to develop an online warning system that detects business schemes that appear legal from a microscopic view, but are illegal from a macroscopic perspective.

References

[1] M. Bilal, J. Thomas, M. Thomas and S. Abraham, Fair BPEL processes transaction using non-repudiation protocols, *Proceedings of the IEEE International Conference on Services Computing*, vol. 1, pp. 337–342, 2005.

[2] M. Gunestas, D. Wijesekera and A. Elkhodary, An evidence generation model for web services, *Proceedings of the IEEE International Conference on System of Systems Engineering*, pp. 1–6, 2009.

[3] M. Gunestas, D. Wijesekera and A. Singhal, Forensic web services, in *Advances in Digital Forensics IV*, I. Ray and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 163–176, 2008.

[4] A. Herzberg and I. Yoffe, The Delivery and Evidence Layer, Cryptology ePrint Archive Report 2007/139 (eprint.iacr.org/2007/139.pdf), 2007.

[5] A. Keller and H. Ludwig, The WSLA framework: Specifying and monitoring service level agreements for web services, *Journal of Network and Systems Management*, vol. 11(1), pp. 57–81, 2003.

[6] S. Kremer, O. Markowitch and J. Zhou, An intensive survey of non-repudiation protocols, *Computer Communications*, vol. 25(17), pp. 1606–1621, 2002.

[7] Los Angeles Times, Internet pyramid scheme alleged, September 28, 2006.

[8] A. Sahai, V. Machiraju, M. Sayal, A. van Moorsel and F. Casati, Automated SLA monitoring for web services, *Proceedings of the Thirteenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pp. 28–41, 2002.

[9] SavvySugar, Help! My friend is part of a pyramid scheme, *MainStreet*, June 23, 2009.

[10] R. Stewart, South Africa investigates an alleged Ponzi scheme, *The Wall Street Journal*, June 17, 2009.

[11] StreamBase Systems, StreamBase Products, Lexington, Massachusetts (www.streambase.com/products-home.htm).

[12] StreamBase Systems, StreamSQL, Lexington, Massachusetts (www.streambase.com/products-streamsql.htm).

[13] U.S. Security and Exchange Commission, Ponzi schemes, Washington, DC (www.sec.gov/answers/ponzi.htm).

[14] D. Valentine, Pyramid schemes, presented at the *International Monetary Fund's Seminar on Current Legal Issues Affecting Central Banks* (www.ftc.gov/speeches/other/dvimf16.shtm), 1998.

[15] C. Westphal, *Data Mining for Intelligence, Fraud and Criminal Detection: Advanced Analytics and Information Sharing Technologies*, CRC Press, Boca Raton, Florida, 2009.

[16] M. Zuckoff, *Ponzi's Scheme: The True Story of a Financial Legend*, Random House, New York, 2005.