Privacy-Preserving Payload-Based Correlation for Accurate Malicious Traffic Detection

Janak J Parekh, Ke Wang, Salvatore J. Stolfo
Department of Computer Science
Columbia University
{janak,kewang,sal}@cs.columbia.edu

ABSTRACT

With the increased use of botnets and other techniques to obfuscate attackers' command-and-control centers, Distributed Intrusion Detection Systems (DIDS) that focus on attack source IP addresses or other header information can only portray a limited view of distributed scans and attacks. Packet payload sharing techniques hold far more promise, as they can convey exploit vectors and/or malcode used upon successful exploit of a target system, irrespective of obfuscated source addresses. However, payload sharing has had minimal success due to regulatory or business-based privacy concerns of transmitting raw or even sanitized payloads. The currently accepted form of content exchange has been limited to the exchange of known-suspicious content, e.g., packets captured by honeypots; however, signature generation assumes that each site receives enough traffic in order to correlate a meaningful set of payloads from which common content can be derived, and places fundamental and computationally stressful requirements on signature generators that may miss particularly stealthy or carefully-crafted polymorphic malcode.

Instead, we propose a new approach to enable the sharing of suspicious payloads via *privacy-preserving* technologies. We detail the work we have done with two example payload anomaly detectors, PAYL and Anagram, to support generalized payload correlation and signature generation *without* releasing identifiable payload data and without relying on single-site signature generation. We present preliminary results of our approaches and suggest how such deployments may practically be used for not only cross-site, but also *cross-domain* alert sharing and its implications for profiling threats.

1. INTRODUCTION

A Distributed Intrusion Detection/Collaborative Intrusion Detection System (DIDS/CIDS) is one that employs *multiple* Network Intrusion Detection and/or Host Intrusion Detection sensors (NIDS/HIDS), often across multiple local area networks, and correlates resulting alerts to get a broader picture of Internet-based threats. Most existing approaches (see section 5) share header-based alert data, e.g., source IPs, destination ports, and aggregate statistics; the goal of correlating such features across multiple sites is to detect *common sources of attack*, especially as they are performing initial scans to build hitlists in a future attack. Not only can these lists be used in building fast-propagating worms [38], they can also be used for *targeted* attacks, e.g., an attacker looking to exploit a critical infrastructure industry that may use common services, such as the financial services industry.

However, the advent of botnets [7, 10, 35] and other forms of indirection have made it far more difficult to discover the *true* attack source, instead of bot machines which play a small role in the actual process. While firewalling can be employed against common IPs [50], there is no guarantee an attacker will not scan using one network and attack using another, thereby defeating proactive attempts.

Instead, one can approach the problem from the perspective of detecting the actual *exploit* used in the attack attempt: in an increasingly monocultured-software world, specific vulnerabilities are common to a large pool of applications [30]. We can leverage the *commonality* of such attack approaches to identify and protect against such attacks even if target machines are unpatched and remain vulnerable. This is particularly advantageous for zero-day worm detection, when common attack vectors may present themselves across many sites in short timeframes, i.e. correlation of common alerts across space, and for stealthy scanning for long time periods, i.e. correlation of common alerts across time. In either case, correlating alerts among collaborating sites requires careful design for accuracy and efficiency.

Of course, exploit-specific vulnerability detection has its challenges: in particular, a reliance on *payload* detection and correlation is necessary. It is impractical to assume that organizations can exchange raw traffic streams; there is far too much data of a potentially sensitive nature. Even if exchanged material is confined merely to suspicious payloads as classified by an anomaly detector, organizations may fear that some legitimate and/or sensitive traffic may be misclassified and exchanged to other, possibly competing institutions. Instead, techniques are required to exchange *privacy-preserving* alerts that make it impossible for other entities to determine the actual content of the underlying traffic, yet at the same time exchanging information that can effectively be correlated. We propose that this is not only possible, but practical and broadly applicable, and propose a collection of techniques to do so.

This paper is organized as follows. Section 2 briefly discusses the concepts of payload anomaly detection, and introduces two detectors developed by our group—PAYL and Anagram—as a representative class of local detectors. Section 3 then introduces the privacy-preserving correlation techniques at the heart of this paper. Section 4 shows some early results based on the techniques described in section 3. We discuss related work in section 5, briefly look at future possibilities in section 6 and conclude in section 7.

2. PAYLOAD ANOMALY DETECTION

In order to better motivate the correlation techniques described in this paper, we first describe two payload anomaly sensors developed at Columbia: PAYL, which implements anomaly detection based on frequency-based 1-gram modeling, and Anagram, which uses binary-based mixtures of higher order n-gram modeling ($n > 1$). Both sensors train on *normal* unencrypted content flows and employ

the models to test for suspicious traffic.[1] Alerts are generated on traffic sufficiently deviant from normal; it is these alerts that we wish to share with other sites to resolve false positives from true zero-day attacks. The reader is encouraged to refer to [47, 45, 46] for detailed descriptions of the aforementioned sensors.

It is also important to note that we do not intend to address all possible payload detection techniques here. Consequently, the techniques described in section 3 may be usable with other (possibly host-based and/or misuse) sensors.

2.1 PAYL: 1-gram frequency modeling

PAYL's models are 1-gram byte frequency distributions conditioned on packet length; tested traffic is classified as normal or malicious by computing the Mahalanobis distance between the distribution of the candidate packets and the frequency model. A larger distance means bigger deviation from the model and a more abnormal packet; thresholding differentiates normal from malicious traffic.

A *raw* PAYL alert typically contains metadata, including the source and target IP/port pair, payload length, and score (distance from model). Additionally, the suspicious packet may be included in its alert. While the payloads can be shared, they significantly increase alert sizes and run into privacy issues, especially for misclassified traffic, i.e. false positives. While PAYL's false positive rates have been determined to be very low [45], the notion of transmitting any raw payload inhibits collaboration among defensive sites.

2.2 Anagram: n-gram binary modeling

Anagram uses an alternative approach to anomaly detection via *binary-based high order n-gram modeling*. Compared to 1-gram, higher order n-grams are better at modeling sequential content information in packets, and thus it is capable of detecting significant anomalous byte sequences and their location within a packet. To avoid significant memory overhead associated with n-gram frequency distributions, only a binary (yes/no) statistic is kept for each possible gram. Scoring is accomplished by counting the percentage of not-seen-before (i.e. unusual) n-grams out of the total n-grams in the packet, and thresholding is again applied to differentiate traffic.

Surprisingly, analysis shows [46] that binary-based modeling produces extremely good results; it turns out the additional data representation of frequency-based modeling is less advantageous when the space of potential grams grows significantly (e.g., the likelihood of having significant frequency information for distinct 5-grams, or 256^5 grams, is significantly smaller than for the 256 distinct 1-gram), and the representational power of higher-order n-grams effectively offsets the loss of frequency information.

The structure of a raw Anagram alert is similar to that of a raw PAYL alert.

2.2.1 Bloom filters

Even though binary-based modeling significantly reduces space overhead, there is still a significant number of possible n-grams as n increases, and a typical hash set structure uses at least 4 bytes per entry. Since only the binary set property is needed, we can use a more efficient, bit-based representation to store the model, reducing data requirements by an order of magnitude. A *Bloom filter* [5] is one such structure; it is represented as a bit array of n bits, where any individual bit i is set if the hash of an input value, $\bmod n$, is i.

[1] Anagram utilizes other information and is semi-supervised.

A Bloom filter contains no false negatives, but may contain false positives if collisions occur; the false positive rate can be optimized by changing the size of the bit array to avoid saturation, as well as using multiple hash functions (and requiring all of them to be set for an item to be verified as present in the Bloom filter). Operations on a Bloom filter are also $O(1)$, keeping computational overhead low. Finally, a Bloom filter has interesting privacy-preserving properties; we explore these in the next section.

3. CORRELATION TECHNIQUES

In this section, we describe several techniques (both raw and privacy-preserving) to support content-based alert correlation. First, however, we develop several metrics as to how we can best compare these techniques.

3.1 Evaluating correlation techniques

The techniques described in this paper essentially trade off the amount of information contained versus the privacy maintained. On one extreme, we can consider the idea of transmitting the raw packets that generated alerts; while this enables any correlation technique, we consider it infeasible because of the sheer amount of data and the fact it is not privacy-preserving. On the other end of the spectrum, we can consider privately-encrypted packet content: unless the key is shared, it essentially appears as noise to peers—but this requires all or no trust. The techniques in this paper fall somewhere in between, and we characterize their relative merits from two perspectives: our ability to correlate data given a transformed version of packets and the amount of privacy that is gained using different privacy-preserving transformations of packet content.

Correlation ability. The fundamental question, given any technique, is whether it is possible to correlate alerts with low false positive and low false negative rates. Given raw packets that generate an alert, there are several well-defined algorithms that aim to accomplish this task. We consider the *longest common subsequence*, or LCSeq, as an appropriate baseline, as it is able to find any non-semantic commonality in the candidate packets, and discuss it below. Other approaches, including semantic matching, are discussed briefly in section 5, and are considered outside the scope of this paper, which focuses on correlation amongst pure network sensors, i.e. no host-specific information.

Given a technique, and a collection of alerts, we can then compute a *similarity score distribution* as each pair of alerts is tested (see section 4.1). This score distribution then becomes a useful metric for comparing correlation ability. If we consider LCSeq as a useful baseline, for instance, we can measure the deviation of other techniques from LCSeq as a comparative measure of how other techniques correlate alerts. Ideally, a network sensor would be able to use a privacy-enabled technique and get similar results, signifying an increase in the privacy preservation while maintaining the ability to determine common threats and exploits.

Privacy gain. We characterize the baseline as having *no privacy* as raw packets are exchanged, and having *total privacy* with encrypted content without the corresponding key (noise). To characterize intermediate approaches, we utilize a probabilistic model: given a representation of the encoded payload, what is the likelihood that a curious peer would be able to reconstruct the original, possibly sensitive data? For most of the approaches listed, we can estimate this probability by determining the number of original payloads that could be represented by the encoded alert; the resulting measurements are discussed in section 4.4.

Correlation speed. Finally, one remaining important characteristic is the ability to correlate *quickly*, especially if many sites are involved with many alerts being generated and exchanged. This "speed" metric is reflected in two aspects: the resulting alert size after a transformation is applied, and the computation overhead necessary to transform the original alert. As with the previous cases, we consider raw packets the baseline: it is the largest unencrypted alert encoding (up to 1500 bytes, i.e. bounded by packet size, per alert) and LCSeq is amongst the slowest correlation mechanisms (up to polynomial-time with respect to buffer size).

3.2 Alert correlation

Alert correlation is presented as three main approaches: raw packet alert correlation, frequency-based alert correlation, and n-gram alert correlation.

3.2.1 Baseline: Raw payload correlation

As previously discussed, we choose raw packet alert correlation as a baseline technique: it contains the most complete original information.

SE: String Equality. This is the simplest and most intuitive correlation approach. Two alerts are deemed similar to each other only if they have identical content. This metric is very strict and does minimize false positives, but has no tolerance for any variation—fragmentation, polymorphism, obfuscation, etc. Equality is memory and computationally efficient (linear time).

LCS: Longest Common Substring. LCS is one of the classic string comparison techniques; it is less deterministic than SE, and is not susceptible to fragmentation. The longer the string that LCS computes, the greater the confidence that the compared alerts are similar. While it allows minor payload manipulation, multiple changes often cause a short LCS, reducing confidence in its correlation ability. LCS is reasonably fast; a suffix-tree implementation is linear-time, but at the cost of having to store a suffix tree per alert (or $O(n^2)$ for a naive but memory-efficient algorithm).

LCSeq: Longest Common Subsequence. LCSeq can be considered a generalization of LCS; instead of finding a single contiguous matching block, LCSeq allows non-matching characters to be interposed. This enables detection despite a variety of payload manipulation operations, including insertion and reordering, and potentially polymorphism. Like LCS, the length of a LCSeq is an indication of similarity. Its main shortcoming is its computation overhead; at best, sparse dynamic programming can achieve, on average, $O(n \lg n)$ complexity (and can range to $O(n^2 \lg n)$ worst-case).

ED: Edit Distance. Edit distance, also known as Levenshtein distance, is another most commonly-used approach to compare string similarity. It computes the smallest number of insertions, deletions, and substitutions required to change one string into another. In general, it has similar properties as LCSeq.

3.2.2 Frequency-modeled 1-gram alert correlation

Having discussed different techniques for raw payload comparison and correlation, we now describe our first alert transformation: frequency modeling. As our work on PAYL demonstrates [47], 1-gram frequency models are a good indicator of the nature of packet content. We can leverage this technique and use frequency distributions as alerts, either with the corresponding normalized frequency counts or with an approximation of this information.

Frequency Distribution. A packet payload can be represented by its byte frequency distribution, making it nearly impossible to reconstruct the actual payload except in degenerate cases—the byte distribution contains byte values but no sequential information. Given two packets with their respective distributions, we can apply standard distance metrics to determine similarity; Manhattan distance is efficient ($O(n)$ in length of the alert) yet produces a good approximation of the actual distance. Frequency-based alerts are comparatively sized compared to packets; a floating-precision frequency distribution takes 1KB of space.

Z-String. A more compact frequency representation based upon the packet payload's byte distribution is what we term a "Z-String", short for "Zipf String" [47]. As its name implies, when a byte frequency distribution is rank-ordered, it usually produces a Zipf-like distribution (exponentially decreasing frequency values). We rank order the distribution of a suspicious packet from most frequent to least and drop the frequency counts, resulting in a Z-String. A Z-String relies on the relative notion of frequency just by the ordering of the individual byte values, and since it is a string, we can apply the raw matching techniques described above to the Z-Strings themselves. Z-Strings are also often smaller than full packets (e.g., 8-bit byte-based packets would be referenced by a 256-byte Z-String), and as such the string comparison times are generally shorter than on the raw packets themselves. However, Z-Strings still have an $O(n \lg n)$ creation overhead in the size of the alphabet. (See section 4.3 for an example generated Z-String.)

3.2.3 Binary-modeled n-gram alert correlation

While frequency-modeled 1-gram alerts offer a measure of privacy, 1-gram modeling cannot represent a *sequence* of characters. For worms and other malicious binary payloads, we may want to capture such sequences, as they may serve as invariants across multiple suspect payloads that can be correlated. As discussed in [46], binary-based modeling produces surprisingly good results and leads to two different possible alert types.

N-gram signature. We can generate a list of n-grams that are found to be suspicious from an originating packet. Such a "signature" is position-independent while capturing specific malicious byte sequences. Given two n-gram signatures, we can simply compute the intersection of the two and threshold the cardinality of the intersected set to determine a similarity score. Such an intersection is linear time in the length of the signatures by using fast set-based data structures; depending on the n-gram size and packet content, this can vary significantly; while most packets are regular and have few n-grams, encrypted traffic, with a very flat byte distribution, can have as many n-grams as the size of the packet itself. In either case, an n-gram signature is a degenerate form of a raw packet; when distributing large n-grams, this is clearly not privacy-preserving, as even a 5-gram can contain a password. In these cases, we need a transformation on the n-gram itself.

Bloom filter n-gram signature. Instead of publishing an n-gram signature, we can instead insert the n-grams into a Bloom filter and publish it. Since Bloom filters support both insert and verify, set intersections can be done between a (local) "raw" n-gram signature and a published BF n-gram signature; such an approach can identify the same n-grams as the previous technique, but without yielding *other*, potentially sensitive n-grams. This approach is also linear time but leverages the space efficiency afforded by Bloom filters. Additionally, space and computation overhead can be reduced by publishing *multiple* alerts via a single Bloom filter; instead of a single collection of n-grams, we treat the Bloom filter as a *bag* of suspicious n-grams. This enables a multiplicative reduction in the

amount of data transmitted and work needed to compute intersections.

Incidentally, correlating *two* BF n-gram signatures can be done via bitwise AND, but the actual n-gram content cannot then be retrieved—the only extractable feature is an approximate score, which can give a general threat metric, but which loses some of the advantage of payload analysis. A more general approach, model comparison, is potential future work (see section 6).

4. RESULTS

4.1 Similarity Score

As discussed in section 3.1, we compute a set of *similarity scores* for every correlation technique, $0 \leq score \leq 1$, with a higher score implying a more similar pair of alerts.

Raw packets and Z-Strings. For both of these alert types, our basket of string comparisons can be used. For SE, the score is binary: 0 or 1, where 1 means equality. For LCS and LCSeq, we use the percentage of the common LCS or LCSeq length out of the total length of candidate strings: $score = 2 * C/(L_1 + L_2)$, where C is the length of LCS/LCSeq and L_i is the length of string i. For ED, larger values imply dissimilarity; we normalize it as $score = 1 - D/(L_1 + L_2)$, where D is the computed edit distance and L_i the same as LCS/LCSeq.

Frequency distributions. As mentioned before, frequency distributions are compared using Manhattan distance: $M = \sum_{i=1}^{n} |x_i - y_i|$, $score = M/2$.

Raw and BF n-grams. Since we no longer have full packet content, we instead compute the percentage of common n-grams: $score = 2 * N_c/(N_1 + N_2)$, where N_c is the number of common n-grams and N_i the number of suspicious n-grams in alert i. If a Bloom filter is used, a count may be kept with it *or* approximated by N_b/N_h, i.e., the number of bits set divided by the number of hash functions used.

4.1.1 Testing with real traffic

To compare the approaches, we randomly sampled packets from three sources: 100 packets each from *www* and *www1* (two heavily-trafficked Columbia CS webservers), and 56 malicious packets from a sample of attacks (CodeRed, CodeRed II, WebDAV, Mirela, a phpBB forum attack, and an IIS buffer overflow (MS03-022) exploit). These packets were paired off in three sets: 100^2 "good-vs-good" pairs of *www* and *www1* traffic, $56*55/2$ "bad-vs-bad" pairs formed in the cross-product of the malicious packet dataset, and $100 * 56$ "good-vs-bad" pairs of *www1* and malicious packets. Similarity scores were generated for all of the resulting pairs with all techniques, except SE, which is too brittle to produce meaningful comparisons, and the n-gram analyses, which cannot be compared over an entire packet.

Figure 1 visualizes a small random subset (80 pairs) of the scores generated from the "good-vs-good" source. As figure 1 shows, the performance plots of the methods appear similar, although their centers and scale values differ as the scores are not normalized between the correlation methods. On raw payloads, LCSeq and ED bear very similar results, while comparisons on Z-Strings yield "flatter" results, as less information is compared.

As a more complete experiment, normalized scores were generated and compared for all of the pairs formed amongst the three datasets.

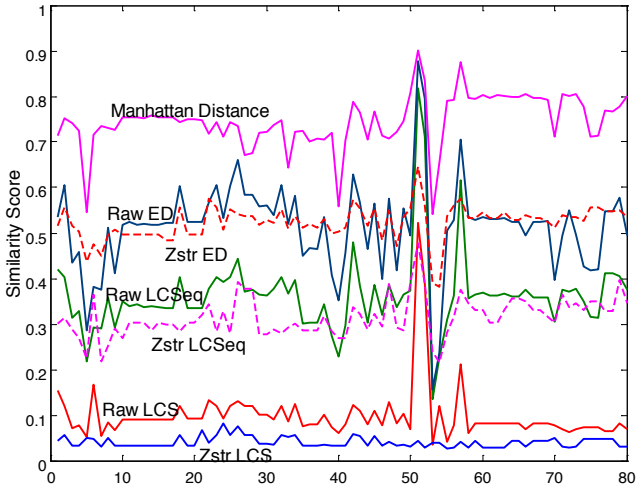

Figure 1: Similarity score comparison of 80 random pairs of "good-vs-good" alerts.

To normalize the scores for a comparison, we first compute *similarity score vectors* V_A, V_B for the same data over two techniques A and B. The center of the two vectors are then aligned by shifting the median of V_A to match V_B. Finally, V_A's range is scaled proportionally so that its min and max values match V_B's. This normalization allows us to compute the Manhattan Distance of the two vectors, $distance = \sum_{i=1}^{n} |V_{A_i} - V_{B_i}|$; smaller values imply greater similarity between the two methods. Note that these scores are relative and dependent on the data used; the normalized results are only useful for comparing against a baseline, not as a source of absolute values or across datasets. These pairs were tested with each technique, and the resulting scores were normalized against and compared to the LCSeq score over raw packets. Table 1 shows the computed results.

Type	Raw-LCS	Raw-ED	MD	ZStr-LCS	ZStr-LCSeq	ZStr-ED
G-G	.0948	.0336	.0669	.2079	.0794	.0667
B-B	.0508	.0441	.0653	.0399	.0263	.0669
G-B	.0251	.0241	.0110	.0310	.0191	.0233

Table 1: Manhattan distance from Raw-LCSeq; lower is better.

Averaged over the three scores, Raw-ED is, unsurprisingly, closest to Raw-LCSeq. When privacy-preserving methods are considered, Manhattan distance performs the best overall, and particularly well for good-vs-bad comparison. All of the privacy-preserving methods are close when correlating pairs with attack traffic; we conjecture that significantly different byte distributions enable effective comparison even when some information is lost via privacy-preservation.

4.2 Cross-Domain Alert Correlation

Next, we compare the techniques by examining their actual performance in identifying true alerts from false positives. Ideally, all false alerts are eliminated by a small similarity score (i.e. the site that produced the alert was the only site that saw this suspicious packet) while true alerts are identified with high similarity scores (i.e. the attack has been launched against more than one site).

In this experiment, we first randomly mix the aforementioned collection of attacks into two hours' traffic from *www* and *www1*, respectively. Multiple instances of attacks—4 for CodeRed and 3 for CodeRed II—are present to simulate a real-world worm attack.

The attacks are also fragmented differently, as CodeRed does in the wild; for instance, CodeRed may fragment into a sequence of $(1448, 1448, 1143)$ length packets, $(4, 375, 1460, 1460, 740)$ length packets, etc. Multiple instances also enable testing correlation *between* different attack types (e.g., CodeRed vs. CodeRed II).

Next, the two mixed traffic sets are each run through PAYL and Anagram with previously-built models and with the alerting threshold lowered so that 100% of the attacks are detected, but with higher (and comparable) false positive rates. The resulting alert sets are correlated against each other using each of the techniques; the results are summarized in figure 2. For each method, the stacked bar represents correlation results for *false positives*. The shaded portion of the bar represents the 99.9% percentile similarity score range, while the white represents the worst-case (highest) score; in other words, while the worst-case FP score can be high, the vast majority of false positives score relatively low.

The asterisk-marked ("*") lines represent the range of similarity scores when instances of the same worm are correlated, and the open circle-marked ("o") lines represent scores across CodeRed and CodeRed II—a very simple measure of polymorphism. The other worms, which were inserted without fragmentation, all scored at or near 1, and so are not shown.[2]

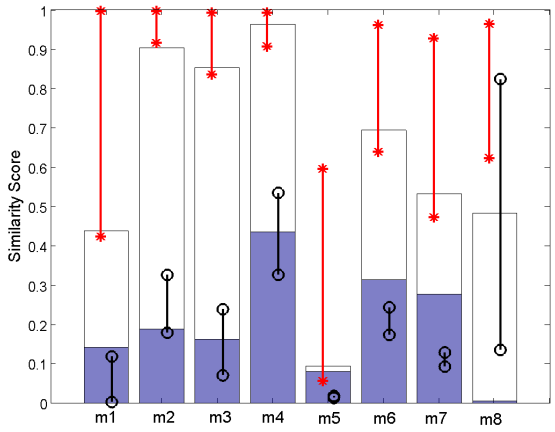

Figure 2: Methods comparison. The correlation methods are, from 1 to 8, Raw-LCS; Raw-LCSeq; Raw-ED; Frequency-MD; Zstr-LCS; Zstr-LCSeq; Zstr-ED; N-grams with $n = 5$.

We can draw several conclusions. First, correlation of identical (non-polymorphic) attacks works perfectly and accurately for all techniques. Most of the techniques can also correlate multiple instances of fragmented attacks; of the privacy-preserving techniques, MD, LCSeq and ED on Z-Strings, and n-gram analysis[3] all perform well. (As intuition may suggest, ZStr-LCS is not particularly effective.) Polymorphic worm detection is far harder—even in the case of CR vs. CRII, only Raw-LCSeq and n-grams achieve promising results. N-gram analysis, in particular, stands out; it produces accurate results and is particularly effective at eliminating false positives, and the use of BFs enables privacy-preservation.

4.3 Signature Generation

Correlating alerts across sites also enables the possibility of automatic signature generation and deployment, once true alerts are identified. (We can also potentially use the scores computed during similarity comparison as a "confidence" measure in mitigation strategies to determine whether to deploy a signature.)

Raw packet-based signatures. Given the ability to share raw alerts, we can exchange the LCS or LCSeq of highly similar packets. This has been the subject of much recent work (section 5), is not privacy-preserving, and we do not discuss it further here.

Byte frequency/Z-Strings. Given the first packet of a CodeRed II attack in figure 3 and its byte distribution displayed in figure 4, we can generate a Z-String by ordering the distribution by most frequent to least and dropping frequency information. Figure 5 shows the first 20 bytes of the generated Z-String for the distribution in figure 4, with nonprintable characters shown by their ASCII values. Both frequency distributions and Z-Strings can be used as signatures.

```
GET./default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090
%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%
u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u
00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u0
```

Figure 3: Raw packet of CRII; only the first 301 bytes are shown for brevity.

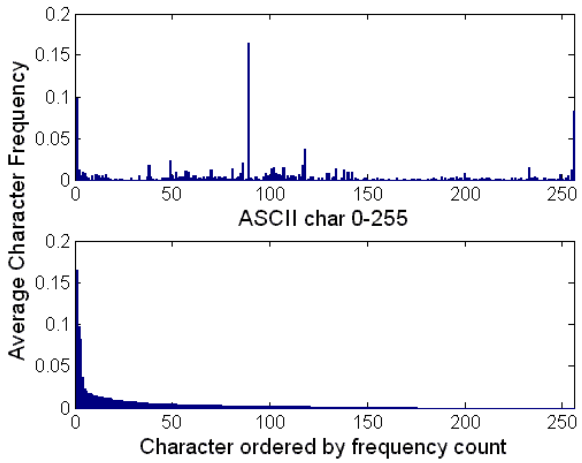

Figure 4: Frequency distribution for the CRII packet.

```
88  0  255  117  48  85  116  37  232  100
100  106  69  133  137  80  254  1  56  51
```

Figure 5: First 20 bytes of the Z-String computed from the CRII packet.

N-Grams. N-grams are an intriguing approach to signature generation; n-grams are position-independent, making them robust to reordering and fragmentation. Additionally, if position information is kept, such a collection *can* be transformed into a flat signature if desired. Figure 6 shows the results when a collection of 5-grams based on the CodeRed II example packet are "flattened". Non-printable characters are represented by "."; "*" represents a wildcard for signature matching. Compared to the original, figure 6 successfully captures the malicious encoding and deemphasizes the padding "noise". Results with different n-gram sizes and another CRII packet are presented in an appendix.

[2]We could have artificially fragmented these worms to simulate the CodeRed experiment, but we expect similar results.

[3]We do not distinguish between published raw n-grams and published BF-based n-grams here, as they produce virtually identical results.

```
* /def*ult.ida?XXXX*XXXX%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%u
cbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8
b00%u531b%u53ff%u0078%u0000%u00=a HT*: 3379
```

Figure 6: Generated 5-gram signature from the CRII packet; only the first 172 bytes are shown for brevity.

4.4 Measuring Privacy Gain

As discussed earlier, we can use a probabilistic model as a first-order approximation to measure *the relative privacy* of each privacy-preserving technique.

Frequency-based approaches. Recovery of the original text from its privacy-preserving encoding can be modeled as follows: given a frequency distribution $f = \{(b_0, \pi_0), (b_1, \pi_1), ..., (b_{n-1}, \pi_{n-1})\}$, where n is the size of the alphabet, b_i is the byte value with probability π_i, and a target length l, we construct the content of a packet $p = \{b_0 b_0 ... b_0 b_1 b_1 ... b_1 \cdots b_{n-1} b_{n-1} ... b_{n-1}\}$, with $\pi_0 l$ copies of b_0, $\pi_1 l$ copies of b_1, and so on.

We can now characterize the *recovery likelihood* $R = 1/(l!/((\pi 0l)!(\pi 1l)!...(\pi n - 1l)!))$, where the denominator is simply the count of all permutations of p. This is an effective estimate of the privacy of the frequency distribution, as it represents the likelihood an attacker will be able to correctly guess the true content of the original packets. This number is, additionally, vanishingly small. For the frequency distribution of the CRII packet shown in figure 4, $R \approx 1/2^{8208}$, well beyond the scope of feasibility, despite the packet's regularity thanks to the large padded section.

The value of R for a Z-String is orders-of-magnitude smaller; not only do permutations of a packet p_i have to be computed, there are many such packets; since no frequency information is stored, one must guess the frequencies for each of the bytes b_i in the packet. In short, effectively guessing the correct base packet p_i *and* its correct permutation is intractable.

N-grams. We can consider the privacy of both a raw collection of n-grams and a corresponding Bloom filter encoding. The raw collection is *not* very privacy-preserving; not only can a byte sequence contain valuable information (e.g., an entire password), significant n-gram collections enable reassembly of much of the original packet, even without position information. Given a 5-gram $\{b_1 b_2 b_3 b_4 b_5\}$ from packet p, one can search for the 5-gram $\{b_2 b_3 b_4 b_5 b_x\}$ in the collection, where b_x is any byte; if found, one can reasonably assume the presence of the 6-gram $\{b_1 ... b_5 b_x\}$ in packet p; since 256^4, the number of "common" 4-grams contained in both two 5-grams, is much greater than a packet's size, it is highly unlikely that the common 4-gram happens to be a coincidence. Combined with the fact that high-scoring alerts can contain nearly as many n-grams as the original packet's size, this is impractical from a privacy perspective.

Instead, as previously proposed, we insert the collection of n-grams into a Bloom filter before publishing it. The size of the Bloom filter need not be much more than the number of n-grams; we can pick a size, say 2^{12} bits, which is more than twice the size of any individual packet and not prone to significant false positives. (This Bloom filter still takes substantially less memory than the n-gram collection itself.) Given such an encoding, the only practical way of recovering the data is to brute-force verify every possible n-gram against the Bloom filter. For example, if we know that only 5-grams are contained in the Bloom filter, there are 256^5 possible n-grams. Not only is testing all such n-grams computationally infeasible, a brute-force attempt is likely to generate many, many false positives, since there are $256^5/2^{12}$ possible n-grams for each set bit in the Bloom filter (assuming one hash function[4]); the recovery likelihood $R = (2^{12}/256^5)^m$, where m is the number of n-grams recovered, is again vanishingly small. This number grows even smaller if multiple n-gram sizes are embedded in the same Bloom filter.

Interestingly, despite the number of possible n-grams for each bit of the Bloom filter, correlating such filters is *not* prone to significant misclassification. We can characterize the "unlucky coincidence" rate $= (\frac{256^5/2^{12}}{256^5})^m$, that is, the likelihood that we happen to incorrectly verify m possible n-grams, each represented by a particular bit b_i, out of all possible n-grams. This simplifies to $(\frac{1}{2^{12}})^m$, which rapidly grows smaller with increasing m. In our experiments, we found that a similarity score threshold of 0.1 produced good results; combined with the fact that the average number of n-grams in a false positive alert is approximately 55, the probability of miscorrelating a Bloom filter alert due to *5* unlucky coincidences is $(\frac{1}{2^{12}})^5$—not a major concern. In short, testing multiple n-grams eliminates coincidences very rapidly. Sizing the Bloom filter appropriately to avoid saturation is a far more important issue.

Given the effectiveness of n-gram analysis, combined with its strong privacy guarantees and compact size, we believe there is great promise for this form of payload-based correlation.

5. RELATED WORK

We discuss selected related work from a number of different network security and intrusion-detection areas, and encourage readers to see the related work sections of [47, 45, 46] for a full discussion on network anomaly detection, frequency and n-gram analysis.

Distributed intrusion detection. Distributed intrusion detection has been researched for over 10 years; most research, e.g., [37, 34, 25] has focused on distribution *within* an enclave, although recent work [2, 19, 50] has looked at Internet-scale correlation and detection. These approaches primarily focus on packet header information and none of them are privacy-preserving; some use end-to-end encryption, but this does not alleviate the notion of sensitive data exchange.

DShield [41] is the most active volunteer-based DIDS project on the Internet that we are aware of, focusing on "top 10"-style reports and blacklists. *DOMINO* [50] organizes a decentralized, heterogeneous collection of NIDS sensors; the paper measured, using DShield alert logs, the notion of information gain—and concluded that 40-60 sites enables building summaries and/or blacklists with high degrees of confidence. An interesting application of this approach would be to measure confidence when payloads are involved.

Signature generation and exchange. Another approach is to exchange "known bad" or *exploit-specific* signatures. Classic payload-based work in this field includes Earlybird [36], Honeycomb [23], and Autograph [20]. These approaches generally implement string-style payload comparison algorithms, including LCS, LCSeq, and Rabin fingerprints, and can be considered alongside the baseline techniques discussed here. Polygraph [33] explicitly addresses the notion of polymorphic worms using LCSeq-like techniques. FLIPS [31] pairs PAYL with an Instruction Set Randomization infrastructure for zero-day worm signature generation. PADS [40], or "Position-Aware Distribution Signatures", seek to blend frequency distributions and packet signature positioning.

[4]Additional hash functions do not affect our analysis.

More recently, work has focused on building *semantic-aware* or *vulnerability*-based signatures to handle multiple (or polymorphic) attacks for the same exploit. Kruegel et. al. [24] use structural analysis of binary code and generate control-flow graphs to catch worm mutations. Shield [43] provides vulnerability-specific but exploit-generic filters based on predefined protocol-based policies. Vigilante [8] introduces the notion of vulnerability-specific *self-certifying alerts* that focus on filtering undesirable execution control, code execution, or function arguments, and can be exchanged via P2P systems. VSEF [32] builds *execution-based filters* that filter out vulnerable processor instruction-based traces. COVERS [26] analyzes attack-triggered memory errors on a host and develops structural memory signatures. Nemean [51] uses session-layer and application-protocol semantics to reduce false positives. Some of these signatures and filter descriptions may be exchangeable using our techniques.

Privacy-preserving collaboration. There have been recent efforts to focus on the privacy of alerts to enable correlation. Lincoln et al. [27] suggest hash-based sanitization of several header fields, enabling equality matching (e.g., identifying the same source IP) while removing other features, including payloads; instead, our techniques keep (and analyze) these payloads. Kissner [21] describes the notion of privacy-preserving set operations using cryptographic techniques; this achieves stronger privacy guarantees than hashing approaches described by Lincoln and those used in section 3.2.3, but it is restricted to set union, intersection, element reduction (set count difference), which could still potentially be used with n-gram analysis. *Privacy-Preserving Friends Troubleshooting Network* [18, 17] extends earlier work on PeerPressure [44]—a collaborative model for software configuration diagnosis—with a privacy-preserving architecture utilizing a "friend"-based neighbor approach to collaboration, including the use of secure multiparty computation to vote on configuration outliers and homomorphic encryption to protect privacy. Xu [48] introduces the notion of "concept hierarchies" to abstract low-level concepts, along with the use of entropy, to balance the sanitization and information gain of alerts; a similar use of entropy may also be applicable here.

Privacy-preserving databases and data mining. There is a tremendous volume of work on various aspects of data mining and databases, e.g. [1, 28]; these primarily assume secure querying, perturbation, and aggregate computation of values amongst one or two databases, and does not generally scale to the collaboration described here. Additionally, most of the research in this field is more concerned with offline analysis.

Bloom filters. Research has been done using Bloom filters as a means of secure indexing and data exchange [4, 3, 16, 14]; in most of these cases, the model focuses on two-party interaction and precisely-defined entities. [12, 6] use Bloom filters for hardware-based packet inspection and classification.

Secure multiparty computation (SMC) [49] is a theoretically attractive way to accomplish privacy-preservation; certain forms of correlation, such as intersection, can be fashioned as such a computation problem. Du et al. [13] discuss a model to transform standard computation problems to secure multiparty computations, and review the possibility of sharing intrusion detection information; however, algorithmic cost remains a concern [27], especially with large alert streams.

6. FUTURE WORK

There are significant opportunities for future research, including:

Wide-scale deployment. These techniques are specifically designed to be deployed at many enclaves to increase the correlation power and confidence provided by sensors at different sites with different content flows. We have begun an early integration of this work with our *Worminator* distributed collaborative intrusion detection platform [39, 29] and anticipate reporting on the results in the near future.

Polymorphic worm detection and mimicry attack. As suggested by section 4, n-gram analysis has the potential of detecting polymorphic worms. While the problem becomes significantly harder as polymorphic worm engines launch *mimicry attacks* [42] to mask themselves, such attacks are generally site-specific (see [22, 11] for examples of such approaches); correlating across sites *may* have the potential to better detect such attacks.

Information-theoretic analysis. A more general approach to comparing correlation techniques is to measure the comparative *information gain* each correlation technique provides. However, this requires an accurate characterization of the distribution of packet content, which is both protocol and site content flow-specific. Additionally, information gain does not directly translate to correlation ability: as our results show, it is possible to correlate alerts reasonably well with significantly less information. We are currently working on acquiring data from multiple sites so we can build such a characterization, and seek collaborators to work with us on this problem using our respective content flows.

Model correlation. Given different site anomaly models, there is the possibility of privacy-preserving model comparison. For example, given two Bloom filters that represent anomaly models for Anagram, we can do a bitwise AND of the two Bloom filters to estimate the number of common "good" n-grams, or a bitwise OR of the two Bloom filters to aggregate and update the respective models. Further discussion of this concept is beyond the scope of this paper; see [15, 9] for an application of this approach to enhance access control.

7. CONCLUSION

We have presented a view of cross-site and cross-domain collaborative security by way of sharing content-based alerts among sites. It is to everyone's benefit to share important information without violating policies that inhibit the disclosure of information. In particular, content-based alerts generated by locally-trained payload anomaly detectors reveals an opportunity to detect the early onset of zero-day worm or targeted attacks. We present a comparative evaluation of alternative correlation strategies and accuracy measures using test data sets with known worm exploits; the methods include a proposed estimate of the "privacy gain" each method affords. This is important in approaching the problem analytically in order to help break down barriers to collaboration. We find that cross-site and cross-domain privacy-preserving "suspect payload" alert sharing is feasible and useful as revealed in the analysis of Bloom filter-exchanged alerts encoding suspect anomalous n-grams.

The techniques hold promise for other purposes as well. For example, sites may exchange their respective anomaly detection models to measure their respective "content flow diversity", enabling estimation of the relative value of different anomaly alerts generated by different sites. More similar sites may have a higher chance of detecting common exploits. Finally, privacy-preserving content alerts may also be useful for other problems, such as collaborative spam filtering, suspicious content detection for botnet command-and-control data streams, etc.

8. ACKNOWLEDGMENTS

We would like to thank Gabriela Cretu, Wei-Jen Li, Vanessa Frias-Martinez, Michael Locasto, Angelos Stavrou, and Angelos Keromytis for their feedback and collaboration on our design. We would also like to thank Panagiotis Manolios and Peter Dillinger for their suggestions in Bloom filter design. This work was partially sponsored under a grant with the Army Research Office, No. DA W911NF-04-1-0442.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In *ACM SIGMOD*, 2000.

[2] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis, and D. Li. A Cooperative Immunization System for an Untrusting Internet. In *IEEE International Conference on Networks*, 2003.

[3] M. Bawa, R. J. Bayardo Jr., and R. Agrawal. Privacy-Preserving Indexing of Documents on the Network. In *VLDB*, 2003.

[4] S. M. Bellovin and W. R. Cheswick. Privacy-Enhanced Searches Using Encrypted Bloom Filters, 2004.

[5] B. H. Bloom. Space/time trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] F. Chang, W.-c. Feng, and K. Li. Approximate Caches for Packet Classification. In *IEEE INFOCOM*, 2004.

[7] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting and Disrupting Botnets. In *USENIX SRUTI Workshop*, Cambridge, MA, 2005.

[8] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham. Vigilante: End-to-End Containment of Internet Worms. In *ACM SOSP*, 2005.

[9] G. Cretu, J. J. Parekh, K. Wang, and S. J. Stolfo. Intrusion and Anomaly Detection Model Exchange for Mobile Ad-Hoc Networks. In *IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, 2006.

[10] D. Dagon, C. Zou, and W. Lee. Modeling Botnet Propagation Using Time Zones. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, 2006.

[11] T. Detristan, T. Ulenspiegel, Y. Malcom, and M. von Underduk. Polymorphic Shellcode Engine Using Spectrum Analysis, 2003.

[12] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Deep Packet Inspection using Parallel Bloom Filters. In *IEEE Symposium on High Performance Interconnects (HOTI)*, 2003.

[13] W. Du and M. Atallah. Secure Multi-Party Computation Problems and Their Applications: A Review and Open Problems. In *New Security Paradigms Workshop*, 2001.

[14] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *ACM SIGCOMM*, 1998.

[15] V. Frias-Martinez and S. J. Stolfo. BARTER: Profile Model Exchange for Behavior-based Access Control. Technical report, Columbia University, 2006. Submitted to conference.

[16] E.-J. Goh. Secure Indexes. Technical report, Stanford University, 2004.

[17] Q. Huang, D. Jao, and H. J. Wang. Applications of Secure Electronic Voting to Automated Privacy-Preserving Troubleshooting. In *ACM Conference on Computer and Communications Security*, Alexandria, VA, 2005.

[18] Q. Huang, H. J. Wang, and N. Borisov. Privacy-Preserving Friends Troubleshooting Network. In *NDSS*, San Diego, CA, 2005.

[19] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *WETICE*, 2003.

[20] H.-A. Kim and B. Karp. Autograph: Toward Automated, Distributed Worm Signature Detection. In *USENIX Security Symposium*, San Diego, CA, 2004.

[21] L. Kissner and D. Song. Privacy-Preserving Set Operations. In *CRYPTO*, 2005.

[22] O. Kolesnikov, D. Dagon, and W. Lee. Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic, 2006.

[23] C. Kreibich and J. Crowcroft. Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. In *ACM Workshop on Hot Topics in Networks*, Boston, MA, 2003.

[24] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Polymorphic Worm Detection Using Structural Information of Executables. In *Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, 2005.

[25] C. Kruegel, T. Toth, and C. Kerer. Decentralized Event Correlation for Intrusion Detection. In *International Conference on Information Security and Cryptology*, 2002.

[26] Z. Liang and R. Sekar. Fast and Automated Generation of Attack Signatures: A Basis for Building Self-Protecing Servers. In *ACM Conference on Computer and Communications Security*, Alexandria, VA, 2005.

[27] P. Lincoln, P. Porras, and V. Shmatikov. Privacy-Preserving Sharing and Correlation of Security Alerts. In *USENIX Security*, 2004.

[28] Y. Lindell and B. Pinkas. Privacy Preserving Data Mining. *Cryptology*, 15(3):177–206, 2002.

[29] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo. Towards Collaborative Security and P2P Intrusion Detection. In *IEEE Information Assurance Workshop*, West Point, NY, 2005.

[30] M. E. Locasto, S. Sidiroglou, and A. D. Keromytis. Software Self-Healing Using Collaborative Application Communities. In *Internet Society (ISOC) Symposium on Network and Distributed Systems Security*, pages 95–106, San Diego, CA, 2006.

[31] M. E. Locasto, K. Wang, A. D. Keromytis, and S. J. Stolfo. FLIPS: Hybrid Adaptive Intrusion Prevention. In *Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, 2005.

[32] J. Newsome, D. Brumley, and D. Song. Vulnerability-Specific Execution Filtering for Exploit Prevention on Commodity Software. In *Network and Distributed Security Symposium (NDSS)*, San Diego, CA, 2006.

[33] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically Generating Signatures for Polymorphic Worms. In *IEEE Security and Privacy*, Oakland, CA, 2005.

[34] P. Porras and P. G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *National Information Systems Security Conference*, 1997.

[35] H. Project and R. Alliance. Know your Enemy: Tracking Botnets, 3/13/05 2005. `http://www.honeynet.org/papers/bots/`.

[36] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated Worm Fingerprinting. In *6th Symposium on Operating Systems Design and Implementation (OSDI '04)*, San Francisco, CA, 2004.

[37] S. Staniford-Chen, S. Cheung, R. Crawford, and M. Dilger. GrIDS - A Graph Based Intrusion Detection System for Large Networks. In *National Information Computer Security Conference*, Baltimore, MD, 1996.

[38] S. Staniford-Chen, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. In *USENIX Security*, 2002.

[39] S. J. Stolfo. Worm and Attack Early Warning: Piercing Stealthy Reconnaissance. *IEEE Security and Privacy*, 2004.

[40] Y. Tang and S. Chen. Defending Against Internet Worms: A Signature-Based Approach. In *IEEE Infocom*, Miami, FL, 2005.

[41] J. Ullrich. DShield home page, 2005. `http://www.dshield.org`.

[42] D. Wagner and P. Soto. Mimicry Attacks on Host-Based Intrusion Detection Systems. In *ACM CCS*, 2002.

[43] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits. In *ACM SIGCOMM*, 2004.

[44] H. J. Wang, J. C. Platt, Y. Chen, R. Zhang, and Y.-M. Wang. Automatic Misconfiguration Troubleshooting with PeerPressure. In *OSDI*, San Francisco, 2004.

[45] K. Wang, G. Cretu, and S. J. Stolfo. Anomalous Payload-based Worm Detection and Signature Generation. In *Symposium on Recent Advances in Intrusion Detection*, Seattle, WA, 2005.

[46] K. Wang, J. J. Parekh, and S. J. Stolfo. Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. Technical Report CUCS-020-06, Columbia University, 2006. Submitted to conference.

[47] K. Wang and S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Symposium on Recent Advances in Intrusion Detection*, Sophia Antipolis, France, 2004.

[48] D. Xu and P. Ning. Privacy-Preserving Alert Correlation: A Concept Hierarchy Based Approach. In *21st Annual Computer Security Applications Conference*, Tucson, AZ, 2005.

[49] A. C. Yao. Protocols for Secure Computations. In *IEEE Symposium on Foundations of Computer Science*, 1982.

[50] V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *NDSS*, 2004.

[51] V. Yegneswaran, J. T. Giffin, P. Barford, and S. Jha. An Architecture for Generating Semantics-Aware Signatures. In *USENIX Security Symposium*, 2005.

APPENDIX

We show further results of signature generation here; for brevity, we only show the beginning part of the packet or generated signature. Please see section 4 for the full discussion.

First, in figure 7, we present the generated signature using 7-grams for the first packet of CodeRed II as shown in figure 3.

```
%*ET /default.ida?XXXXXX*XXXXXX%u9090%u6858
%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%
u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u
0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HT
TP*type:
```

Figure 7: Flattened 7-gram signature for CRII's first packet in figure 3.

To give a better example, we can look at the signatures generated for another packet P' of CodeRed II, which explicitly contains malicious system calls. We first present part of the signature generated by using LCS, which can also represent partial original raw packet, and then the generated n-grams signatures.

```
|d0|$@|0 ff|5|d0|$@|0|h|d0| @|0|j|1|j|0|U|f
f|5|d8|$@|0 e8 19 0 0 0 c3 ff|%‘0@|0 ff|%d0
@|0 ff|%h0@|0 ff|%p0@|0 ff|%t0@|0 ff|%x0@|0
 ff|%|0@|fc fc fc fc fc fc fc fc fc fc fc f
c fc fc fc fc fc fc fc 0 0 0 0 0 0 0 0 0 0
0 0 0|\EXPLORER.EXE|0 0 0|SOFTWARE\Microsof
t\Windows NT\CurrentVersion\Winlogon|0 0 0|
SFCDisable|0 0 9d ff ff ff|SYSTEM\CurrentCo
ntrolSet\Services\W3SVC\Parameters\Virtual
Roots|0 0 0 0|/Scripts|0 0 0 0|/MSADC|0 0|/
C|0 0|/D|0 0|c:\,,217|0 0 0 0|d:\,,217|fc f
c fc fc fc fc fc fc fc fc fc fc fc fc fc
 fc fc fc fc fc fc fc fc
```

Figure 8: Part of the LCS result for the CRII packet P'.

```
|%d0@|0 ff|%h0@|0 ff|%p0@|0 ff|%t0@|0 ff|%x
0@|0 ff|%|0@|fc fc fc fc fc fc fc fc fc fc
fc fc fc fc fc fc fc fc fc 0 0 0 0 0 0 0 0
0 0 0 0 0|\EXPLORER.EXE|0 0 0|SOFTWARE\Micr
*soft\Wind*s NT\CurrentVersion\Winlogon|0 0
 0|SFCDisable|0 0 9d ff ff ff|SYSTEM\Curren
tC*trolSet\Services\W3SVC\Para*ters\Virtual
 Roots|0 0 0 0|/Scripts|0 0 0 0|/MSADC|0 0|
/C|0 0|/D|0 0|c:\,,217|0 0 0 0|d:\,,217|fc
fc fc fc fc fc fc fc fc fc
```

Figure 9: Part of the flattened 5-gram signature for the CRII packet P'.

```
%d0@|0 ff|%h0@|0 ff|%p0@|0 ff|%t0@|0 ff|%x0
@|0 ff|%|0@|fc fc fc fc fc fc fc fc fc fc f
c fc fc fc fc fc fc fc fc 0 0 0 0 0 0 0 0 0
 0 0 0 0|\EXPLORER.EXE|0 0 0|SOFTWARE\Micro
soft\Windows NT\CurrentVersion\Winlogon|0 0
 0|SFCDisable|0 0 9d ff ff ff|SYSTEM\Curren
tControlSet\Services\W3SVC\Parameters\Virtu
al Roots|0 0 0 0|/Scripts|0 0 0 0|/MSADC|0
0|/C|0 0|/D|0 0|c:\,,217|0 0 0 0|d:\,,217|f
c fc fc fc fc fc fc fc fc fc f
```

Figure 10: Part of the flattened 7-gram signature for the CRII packet P'.