KEEPING BETA TESTS CONFIDENTIAL

A comprehensive guide to prerelease customer validation information security.

Centercode

Before We Begin

Please read the following before continuing.

About Private Betas

This guide is designed to specifically address private (closed) beta tests, also commonly referred to as **customer validation (CV), customer acceptance tests (CAT), user acceptance tests (UAT), prerelease, or field trials**. In this stage, your test is likely comprised of a small pool of customers or prospects who are asked to provide feedback through private channels on your upcoming product. With this in mind, this guide is not intended to address public beta tests, where confidentiality is not a concern.

For the purposes of this guide, we'll be referring to private customer tests as **beta**.

About Centercode

We've been in the business of managing private beta tests since 2001. Within that time, we've successfully executed hundreds of beta tests covering a vast array of products and industries.

In addition, we offer the most advanced beta test management platform available globally, which has been used to run thousands of private tests and includes a wide variety of features specifically designed to further secure your test results.

With all of our experience running beta tests, we've maintained a stellar track record for ensuring confidentiality, resulting in the proven best practices detailed in this guide.

Intended Audience

This guide is written for companies of all types who are concerned about the unofficial disclosure (i.e. leak) of information regarding their product. In particular, Product, Program, Project, Quality, and Beta Managers will find this information helpful in managing their prerelease products.

While much of this document specifically addresses this issue in the context of customer testing, the majority of the concepts and best practices included are equally relevant to the prevention of all prerelease information leaks.

If you've already experienced a leak (or are dealing with one right now), this guide will provide useful information for finding the individual responsible for the leak, mitigating the exposure and impact, and getting back on track to a positive release.

Introduction

Preparing the launch of a new product requires months — and sometimes years — of tremendous team effort from individuals representing nearly every facet of your company.

More often than not, this is done in complete secrecy and with a focus on making a big splash when the product launches, hopefully concealing the early details from both your customers and competitors. Because of this, many companies dread the notion of their first customer-facing test.

Delivering your unreleased product into the hands of beta testers can be a scary prospect. The thought of sending your unfinished product to complete strangers is tough enough, but considering the number of blogs, forums, tweets and news sites out there, it may feel like the risk of exposure is too great. In reality, a well-executed beta test is rarely the source of prerelease information leaks.

Throughout this guide we'll be detailing everything you need to know about keeping your beta tests confidential via the following topics:

- The Types of Prerelease Leaks
- Why Companies Fear Prelaunch Leaks
- Why Beta Testers Are Unlikely to Share Your Secrets
- Where Leaks Really Come From
- Why Leaks Do Sometimes Occur in Beta
- How to Increase Beta Confidentiality
- Additional Steps for Further Reducing Risk
- Monitoring For Leaks
- What to Do if There is a Leak
- Why Beta Leaks Really Don't Matter Too Much
- How Centercode Can Help



The Types of Prerelease Leaks

There are two distinct types of leaks that companies try to avoid during beta testing, both of which we'll cover in depth throughout this guide.

Product Leak

In the case of desktop and mobile software products, most companies fear their product being pirated in beta form prior to release. Like all software sharing, the intention of these leaks is generally split between casual (sharing with friends/family) and true piracy (purposely releasing it publicly to a broader community of "pirates"). Thankfully, true piracy is generally rare and easily mitigated.

Unauthorized redistribution of physical prerelease products is also a concern for many companies. In this extremely rare scenario, a beta tester might try to sell their beta unit on sites like eBay or Craigslist.

Information Leak

The more common leaks, which affect every product type, are information leaks. These exist in the form of written text, photos, or video, which are posted on sites like YouTube, Flickr, blogs, and discussion forums. Often these leaks provide early details and/or opinions about the product, much of which is not based on factual evidence, or is not an accurate representation of the final product.



Why Companies Fear Prelaunch Leaks

There are a number of common and valid reasons why companies fear prerelease leaks. Note that these issues are not specific to beta testing, but are concerns relevant to any unauthorized distribution of unreleased product or related information. In this guide we will discuss best practices for greatly diminishing the risk of these issues, as well as useful response techniques which you may utilize in the event that a leak does happen.

Direct Loss of Sales	If the users already have the product, then there isn't any of incentive for them to purchase it.
Delayed Sales	If your customers become aware that a new release is imminent, they may hold off purchasing the current release, thus stalling your revenue.
Negative Perception	Beta versions are generally not representative of the final release product, and will likely provide a negative experience to prospective customers — some of which may not even realize the product is in an unfinished beta form. This can result in a tarnished reputation for your company or product and a further reduction in sales.
Misinformation	Products change tremendously prior to release. Features in an early beta may be removed, in which case it's better for the broader audience to not know they ever existed, since they don't represent the final product that they're purchasing. Testers and other early users may also not have their facts straight, resulting in the inaccurate details being shared.
Message Control	Leaks are most commonly feared by marketing and PR. These individuals work hard to craft a highly-refined message and customer experience, and really don't want someone else announcing their product for them.
Competitive Advantage	Competitors gaining early intelligence gives them an unfair advantage in product development and perception — both of which can be detrimental to your own product.
Security Exploitation	The last reason, which applies to some products, involves technical security. Given that beta releases inherently have both known and unknown bugs, these issues can lead to holes which can be exploited in negative ways.

Why Testers Are Unlikely to Share Your Secrets

Beta testers can often be the first suspects if there is a leak leading up to a product launch. What many people forget, however, is that there are a number of good reasons why beta testers are unlikely to share your secrets. The following are a few of the more notable rationales:

Exclusivity	Beta testers thrive on the exclusivity of being members of a small group of individuals who have been handpicked to participate in your project. They love to be the first to know about your product. Sharing this information negates this exclusivity.
Fear	If you have an NDA (Non-Disclosure Agreement) in place — which you should — your testers do not want to face your legal wrath in the event that a leak comes from their project. Beta participants are generally just enthusiastic customers who have signed up for something exciting, exclusive, and secret. Why would they risk a lawsuit and the potential cost of engaging an attorney for a fleeting moment in the sun?
Respect / Self-Interest	These reasons are relative to one of the core attractions of beta testing opportunities in the first place. People enjoy beta testing because they're passionate about specific products and the companies that make them. These products solve their problems, whether it's business hardware that increases their profits, productivity software that organizes their lives, or games that entertain them. Given this fact, beta testers want to see you succeed. They not only want your product, but they want it to be better, and most of all, they want a personal stake in that success. If a leak will potentially impact the product in a negative way, it's contrary to the reason they volunteered to test. Your beta testers are investing their time and effort in your success. Leaking it will not help them achieve that goal.
Rewards	Your beta testers want to be rewarded for being a part of the project. As long as you're clear about the consequences of a leak, they will make every effort to keep your product a secret. They want to earn the incentive at the end of a project, whether that incentive is the product itself or something else entirely. More importantly, they often want the opportunity to maintain their exclusive relationship with you for future beta projects. Leaking information would absolutely prohibit any opportunities for participation in future tests.

Where Leaks Really Come From

If your project is ready for beta testing, there are a whole host of other places that are much more likely to divulge your secret than a beta tester. At this stage in the product's development, more and more people are touching the product with each passing day. As this happens, the risk of a leak increases exponentially.

If your product has any sort of ecosystem (i.e. plug-ins, API, accessories), then it's possible that your technology partners may share details either deliberately or by accident. In today's market, it's highly uncommon that any product is developed entirely in a silo, so in our experience these types of third party developers and partners are the single most common source of prerelease leaks.

If you're developing hardware, your component vendors might disclose details as you prep your channel for release. Similarly, compliance organizations like the Federal Communications Commission (FCC) produce public records which are constantly watched by interested parties, which could lead to your product being revealed.

Your sales people might begin talking about the product to seed their funnel, or entry-level employees from larger groups such as QA and support may be a little too excited to finally get their hands on an important product. As a result, they could share information that they deem unlikely to cause harm. While rare, we've also seen disgruntled employees serve as a source of untimely leaks.

Even the largest and most secretive companies (i.e. Sony, Apple, etc.) have trouble keeping these avenues secure throughout their development process.

Software developers also have a lot to consider. If you develop for web or mobile, remember that the Internet is a very public place. People are constantly searching for new data, and if you make a popular product, people are carefully scrutinizing everything you do. Even changes on your website can provide hints to users about upcoming products.

The reality is simple: every day it gets harder and harder to keep your secrets a secret. There are so many channels for people to communicate; you have much larger issues than chatty beta testers to worry about — most of which have much greater motivations to share information. When you properly educate test participants on the need for secrecy, they are often the last group you need to worry about in regard to confidential information getting out.

Why Leaks Do Sometimes Occur in Beta

There is always some risk involved when distributing a product to strangers. While we stand by the aforementioned statements, there are other situations you need to consider.

In reality, beta test leaks are very infrequent and can be mitigated if you take the proper steps up front to thoroughly educate your users. In any beta test, the best offense is a strong defense. Take early measures to make certain every tester knows what's at stake and what the consequences of any leak are.

Unclear Requirements	The single most common reason for leaks in beta tests is a simple lack of understanding of the confidentiality requirements of those beta tests. It can be taken for granted that beta testers are intended to represent your average customer, who may or may not include the most technically-savvy or legally-astute individuals. Quite often, companies fail to communicate confidentiality requirements clearly or (in many cases) at all to beta participants. Put simply, it's much more likely that testers will share information if they don't know they're not supposed to.
Excitement	Sometimes the excitement of being on a project gets the better of testers. They might share that they're on a beta for you but never reveal any details. For many testers this is simply enough to stroke their ego, and generally causes little to no harm (harm not being their intention). It's highly unlikely that your product isn't at least announced by the beta stage, and generally the release window is either announced (often with a firm date) or somewhat obvious to the market.
Early Evangelists	Another situation you might encounter is the overzealous evangelist. After weeks of using your product rigorously, some testers become passionate about its benefits. From this experience, they want to spread the word about why everyone needs to own your product. This passion is great, but again, it does violate the terms of the agreement.
Bad Seeds	The rarest of all leaks are the ones that don't necessarily intend harm, but still intended to share information, generally for personal or professional gain. This includes competitors (i.e. industrial espionage), and those looking to increase their status in an online community.

How to Increase Beta Confidentiality

There are a number of simple best practices that can be used to significantly increase the security of any beta test. We'd recommend taking these actions on nearly any project where secrecy is a concern, as they generally require minimal additional effort and often provide supplemental benefits to your beta test, such as increased engagement.

1. Announce Your Test Quietly

You should limit the announcement of your beta test recruiting to the smallest audience possible while still ensuring you still have an adequate number of testers for your project (generally 5 to 10x the total number of testers you desire). This greatly decreases the odds that any competitors who are paying close attention will slip into your beta team.

2. Preface with Confidentiality Obligations

The initial announcement of the beta opportunity is also your first opportunity to communicate the importance of confidentiality. Before testers even apply, they should have a clear understanding of these requirements. This is also the first place to make it clear that competitors and members of the press are not (currently) welcome within your beta test.

3. Recruit Testers Selectively

It's important to be selective about who is participating in the test. All candidates should be taken through an application process that surveys them for key information. While most of this information is intended to profile them for the target market of the product, the same information will help you understand and research the users themselves. All results should be considered under heavy scrutiny, ensuring selection focused on those who demonstrate no negative signs.

4. Limit Your Tester Team Size

It's simple math — the more testers you have the greater chance that someone will leak information, unintentionally or otherwise. Given that industry average beta participation rates are in the range of 20-30%, most companies are forced to over-recruit 3 to 5x as many testers as they need to achieve their testing and feedback goals. Reducing these larger test teams can greatly reduce the risk of leaks.

INCREASE YOUR PARTICIPATION

Our Managed Beta Tests generally achieve greater than 90% participation rates. This alone can cut your chances of beta leaks by 60% or more. We've also published a free whitepaper ("Reaching 90% Beta Participation") which details our entire process.

5. Always Provide an NDA

First, always have a Non-Disclosure Agreement (NDA) in place. The NDA is your first and best tool for ensuring secrecy. It spells out to the tester that you take this very seriously and that there will be consequences should they decide to share any details. Everyone fears legal issues. The concept of a large, well-funded company setting its legal sights on them will induce substantial dread in nearly any person. It's important (and fair) to use this anxiety to your advantage.

FREE NDA KIT

We offer a free Beta Test Agreement Kit which includes a detailed NDA constructed specifically for beta testing, in addition to a plain-English overview which follows these best practices. You're free to customize our NDAs to fit your own needs.

6. Explain your NDA in Plain English

It's important to clearly communicate the specific details of your NDA to every tester. How many of us have clicked past a legal notice, license agreement, or some other important form? Don't let the tester get off that easy. Remind them about what they are signing and always go the extra step to spell out the terms in plain English. It's helpful to cite specific text from the agreement and then remind them of the consequences of ignoring the agreement.

7. Maintain the Secrecy Message

It's also important to consistently evoke the concept of secrecy throughout all ongoing communication with testers. Subtle details such as having "Confidential" noted on beta documents or sending periodic reminders via email about confidentiality help people understand the importance of security. Keep secrecy at the forefront of all your communication and testers will take it seriously.

AVOID AGGRESSIVE MESSAGING

While it's important to express the importance of confidentiality, you should avoid being overly aggressive. If beta testers feel that they're under constant watch or threat, they're less likely to enjoy the experience and/or they will tune out completely. It's important that they understand why confidentiality is so important, but they shouldn't feel under threat just because they're beta testers.

8. Time-bomb Beta Software

While unlikely, threats of piracy should always be considered in every beta test. In an effort to address piracy risks, all beta software should include an expiration at which point it will no longer function. While this won't deter the most technically-astute pirate organizations, it will limit the far more likely risk of exposure from casual piracy.

NEVER HINDER THE BETA RELEASE

While all beta software should be time-bombed, it's also important not to hinder the software in such a way that it's not representative of a real user experience, which would invalidate your beta results. In other words, we never recommend releasing a limited version of your software unless you're specifically focusing on the limited functionality. Beta products that are as close to the final product as possible produce the best feedback.

9. Clearly Identify Beta Status

It's important to clearly mark your product with signs that identify it as an unreleased beta product. This applies to both software and hardware. Given the general understanding that beta means "not finished," this further reduces the risk of negatively impacting the impression of your product or brand in the unlikely event that piracy or unauthorized distribution occurs.

For software, make sure to clarify that the software is a beta release (big red message on loading screen, title bar, etc.). In the event of a true product leak (i.e. piracy), it'll be clear to users that the product they're using is incomplete (perhaps with a time-bomb further ensuring this understanding) and not representative of a purchased product.

For physical products we'd recommend going beyond writing "beta test device" on the product. If possible, include serial numbers and other details so that users realize the hardware can be traced back to them.

BUT WHAT ABOUT PHOTOSHOP?

Of course we all know that serial numbers can be Photoshopped out or simply photographed from an angle that doesn't include them — but that's not really the point. The real goal is to further drill in the message that this should not be shown publicly, and that the unit is specific to you.

10. Contain Their Excitement

We live in a world of constant information sharing. Immensely popular social media sites such as Facebook and Twitter encourage users to share everything they experience on a moment-by-moment basis. Unfortunately, this is counterintuitive to the confidential ambitions of your beta test, where testers are expected to keep some of their most interesting news and experiences completely private.

Thankfully, this is easily combated by providing a secure outlet for these users, most commonly in the form of an online discussion forum. By providing a community forum for your testers, you're not only allowing them to openly share their experiences (both positive and frustrating), but you may also leverage this information to directly improve the product. In this case, not only are you increasing privacy, but you're also increasing engagement by giving testers a reason to come back every day, while also ensuring easy access to all relevant information about your product.

TESTING YOUR FORUMS

If you're planning to use a forum system that's already available to you, it's important that you confirm that your discussions aren't publicly available. Note that just because you're logged in when you view them, this doesn't necessarily mean users who aren't logged in won't be able to. Even if you're not technologically savvy, it's easy to test. Navigate to a post in your forum that's representative of something you wouldn't want shared with the world. Copy the complete URL from the top bar in your browser. Now open a new private window (in Internet Explorer it's called InPrivate Browsing, in Chrome it's called Incognito, and in Firefox it's called Private Browsing). Paste the copied URL into the bar and press enter. You should be asked to log in. If you can see the post before logging in, it's not private. Not only can it be shared, but it's equally likely that search engines such as Google will index your posts and return them in search results. Note that true security goes much further than this, but this will at least determine the public availability of your forums.



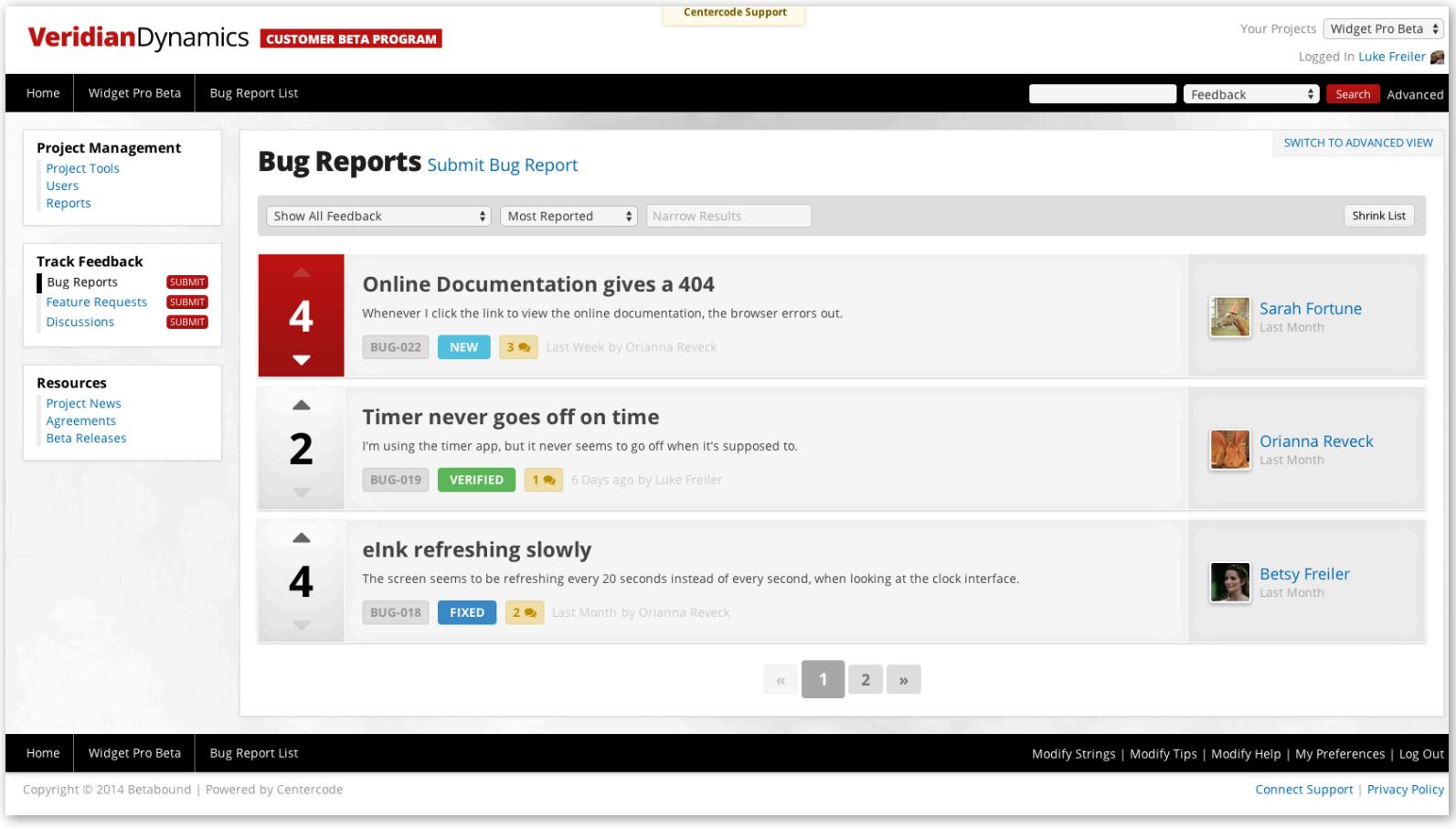
11. Provide Simple Feedback Channels

Beyond an online discussion forum, it's crucial to provide your beta testers with a user-friendly tool that is password protected to share feedback and communicate their ideas, issues, and opinions. Emails can be forwarded, public forums can be linked, and chats can be shared. Use a tool that ensures the security and privacy of your data. It will provide users a place to freely express their ideas and share their excitement, while making it far less likely for them to want to share it other places.

Your efforts to protect your product send a clear message about how serious you treat secrecy. People are used to warning signs in everyday life. Giving beta testers a clear message about what you expect and the consequences of not meeting those expectations helps keep your beta test private.

CENTERCODE

Our beta test management platform allows you to offer your testers secure channels for whatever type of feedback you need (bugs, feature requests, forums, surveys, etc.). Read more at www.centercode.com/software.



Additional Steps for Further Reducing Risk

If your product warrants an extreme concern for security during beta, there are extra measures which can be taken at the cost of additional time investment. While we don't consider these necessary steps for every product, they are activities that we often engage in to provide an extra layer of security.

1. Interview Testers / Validate Phone Numbers

By calling each tester individually, you're accomplishing two important things. First, you're creating an opportunity to specifically address the importance of confidentiality in a very personal and meaningful voice. A real person on the phone has a much greater chance of successful communication than any email.

Second, you're verifying that candidates are using valid phone numbers, which provides a more concrete line between you and them. This indirectly tells them that you would be able to easily find and engage with them in the event that legal recourse was necessary. While it's highly unlikely that this would ever actually happen, the anxiety it introduces will further enhance your security.

TEXT MESSAGE VERIFICATION

You can also meet the second of these two goals by implementing a text message verification component into your beta tester onboarding process. You've probably experienced a system like this in your own personal online banking. In the past we've used Twilio (www.twilio.com) to accomplish this.

2. Use Proven Testers

In the most extreme circumstances, you can use testers who've already proven themselves as capable of keeping beta information to themselves. If you have a beta program in place, you can easily identify successful participants from prior tests. We maintain detailed records on all of our testers in our beta tester community, Betabound, which helps us locate proven testers based on past responsiveness and performance.

DON'T LOSE SIGHT OF YOUR TARGET MARKET

While proven testers can increase your security, it's still essential that they accurately represent the target market of your product. For this reason (and many others), friends, family, and employees are a bad choice for testers, despite their ability to keep information a secret. In this case, you're simply solving one problem by introducing another.

3. Implement Traceable Technology

For software, there are technologies that exist which allow you to transparently encode individual downloads with tokens that uniquely associate a specific file download with the user. In the event of a product leak, it's possible to link that file to the user, thus allowing you to either enforce the NDA or simply blacklist that user from future testing.

Implementing traceable technology doesn't necessarily need to match the message you present to your users. In other words, the mere idea that downloads may be tracked will be enough to discourage most users from casually sharing your product.

4. Include Distinctive Marks on Physical Units

We've worked with hardware customers to include special physical markings on each beta unit. To the naked eye, these marks appear as nothing more than pre-production defects or unintentional scuffs, but are enough to visually identify these units in the event of a leak.

We wouldn't recommend this as a necessary step for most betas, but this strategy may reduce stress of the most fearful of product teams, and in the rare event of a leak, this will allow you to identify the individual so that you can enforce an NDA and/or ensure they're not included in future projects.

5. Get Proactive About the Press

In scenarios where it's very likely that specific websites (blogs, etc.) are chomping at the bit to gain (and expose) early insight into your release, it may make sense to actually invite these individuals to your beta test. In this scenario, you'd still have an NDA, but also an embargo that details when they can publicly talk about your product.

This formal relationship can provide mutually beneficial advantages. It provides them with the early look (and potentially exclusive story) that they want, but it also ensures a reasonable level of control about when they are able to speak about your product. This could turn a fearsome unknown factor into something which is greatly advantageous at release time.

SEGMENTING THE PRESS

If you are planning to include the press in your beta test, you'll want to do so in a different capacity than other testers. If possible (any beta test management system will include this functionality), you'll want to segment them out of your typical tester pool to further reduce the risk of a negative experience (later releases, limited information). Similarly, you should expect and accept limited participation. Keep in mind that their goal is to learn about the product, not test it or expose its bugs.

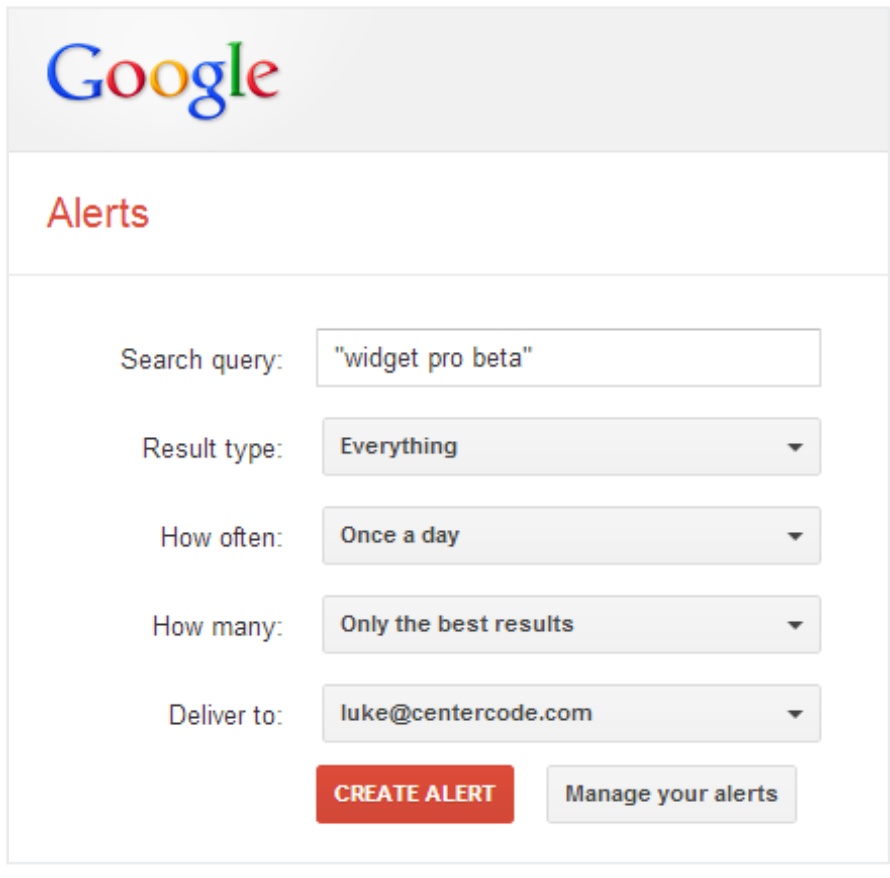
Monitoring For Leaks

If information security is a major concern (and if you're reading this, it probably is), then there ways to keep tabs on the broader Internet environment so that you're the first to know if something does leak, allowing you to react rapidly.

The simplest way to monitor Internet leaks is by combining a set of free and useful tools offered by Google. These tools are Google Alerts (www.google.com/alerts) and Google Reader (www.google.com/reader). Note that while this will not cover 100% of the Internet, it will cover the vast majority of it.

Google Alerts

Google Alerts is a tool which allows you to nominate search phrases that you'd like to be notified of whenever they enter the Google Search Index (e.g. become searchable by Google). Given that Google is designed to monitor the Internet in almost real time, this means that mentions of these terms will raise a flag very quickly, generally within hours.



Alerts can be emailed to you or delivered to a feed. In this case, a feed is a custom RSS (Really Simple Syndication) compilation, which is effectively a custom news stream which you can access via the next tool.

Steps to Setup Google Alerts (www.google.com/alerts):

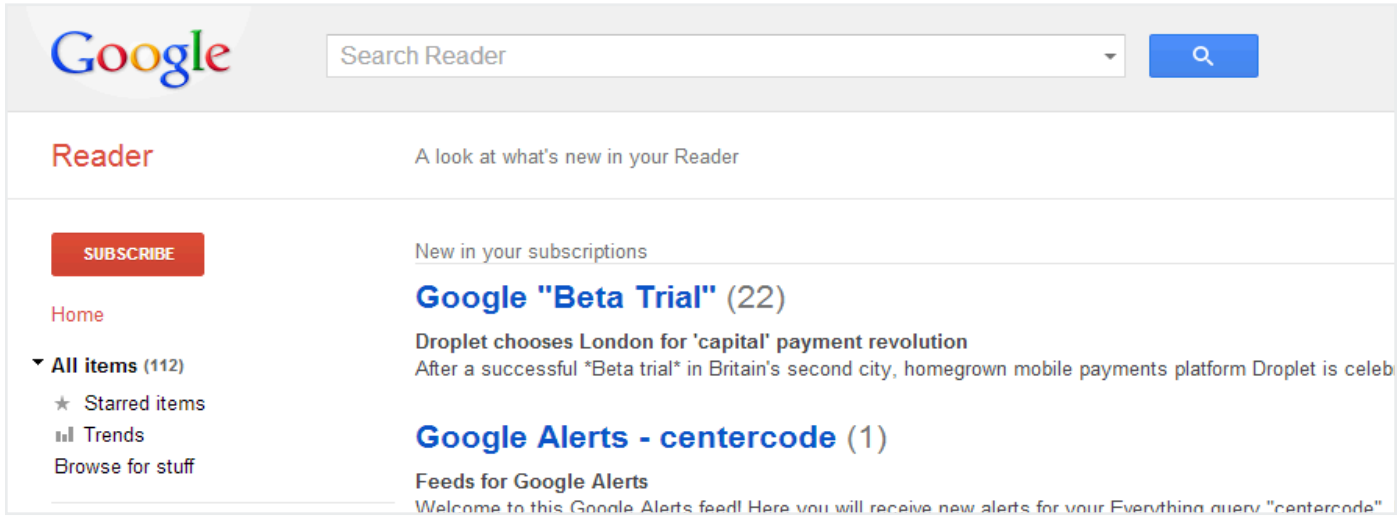
1. Provide a search query to watch for (e.g. “Acme Widget Beta”)
2. For the next options, select “include everything,” “as it happens,” and “all results”
3. For “deliver to,” select “feed” or provide an email address
4. Repeat these steps for additional search terms. We recommend using a wide variety of terms that will identify your product. Surround your search term in double quotes (“this”) to match exact strings, which can be useful in narrowing results.

FEED OR EMAIL?
Email may be a better option if your product is currently in a highly confidential state, thus there will be very minimal traffic. If you’ve announced a private beta in any form, the email option will likely generate a good deal of noise which would be easier to read in a feed format. The main advantage of an email is that alerts can be sent to you in real time, allowing you to react instantly, while the feed must be monitored manually.

Google Reader

Once you’ve created a full set of alerts, you’ll want to use Google Reader to aggregate them into a running list which functions similarly to a custom blog. To do this, you’ll simply click “Manage Alerts” (from within Google Alerts), and then click the Google Reader link under the Deliver to column for each item in the list.

If you now navigate to Google Reader (www.google.com/reader), you’ll be able to see a list of links and summaries of any mention of your beta product across the Internet. Over time this will grow, identifying any buzz surrounding your beta.



What to Do if There is a Leak

There are few experiences in business more disheartening than opening up your web browser or email and discovering that your highly-sensitive and protected information has been leaked.

Regardless of the source (which at this point may or may not be your beta testers), if you've taken all the right steps and people still go out and share your confidential information, then you can still take measures to stop the leak in its tracks.

The process of handling a leak breaks down into five basic steps:

1. Survey the scope of the leak
2. Plan your strategic response
3. Deal with your testers
4. Remove the leaked information
5. Find the rat!
6. Decide what to do with the rat

Each of these will vary based on the severity and impact of the leak.

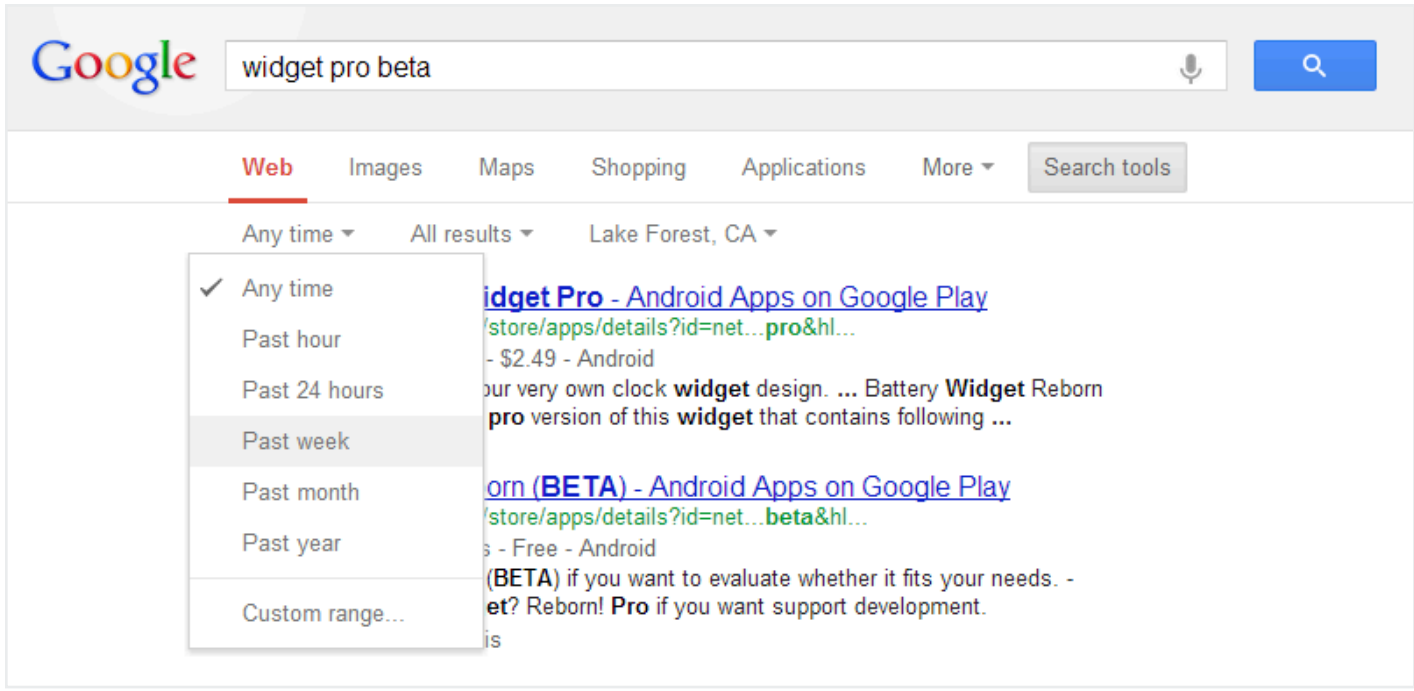
1. Survey the Scope of the Leak

Before you respond to a leak, it's important to survey the situation. Not all leaks are equally impactful, so it's important to build a clear picture prior to taking action.

If the leak happened recently (which is likely the case), you can use Google's near-real-time search capabilities to identify the scope of the availability of the leaked information.

Start by searching for key terms that would accurately identify your project (e.g. "Acme Widget Beta"). It's likely that you'll need to try a number of queries and be specific with your searches in order to avoid false matches. To help with this, you should add quotes around your search strings to require the result to match an exact phrase.

On the search results page, click "search tools," and then "any time." Begin by searching the past hour, then fan out to the day, and finally, the week. While this isn't 100% accurate, it does cover most substantial sites in a near-real-time capacity.



If the leak appears on more than a single site then you'll likely want to start cataloging coverage, including notes about the sentiment of each post and the apparent audience size and relevance of each site's reach.

2. Plan Your Response

Next, you'll want to communicate the results of your search to the group of stakeholders impacted by the leak. This will likely include marketing, public relations, product management, and potentially others including support and sales.

We recommend holding a meeting to discuss the details and impact of the leak. Most importantly, you'll want to determine if there is in fact a negative impact, and if so, how to respond strategically as a company.

If you're late enough in your beta test that the leaked information will cause little impact (which is very likely), ignoring it may be a better use of your time and energy.

3. Deal With Your Testers

Regardless of the severity of the leak, you'll want to address the issue with your beta testers. It's important that they know that you are aware of the leak.

Next, you'll want to want to suspend the beta test immediately for at least a short time. Lock all of your testers out and communicate the lockout via email (and possibly, if serious enough, by telephone). If the product is hardware and the impact is severe, you should also demand that all beta units be returned immediately.

More than likely, your testers will initially be outraged. Explain to them that, given the risk, you're forced to ban every tester from the project and any future testing until you can find the source of the leak. You have no other choice in the matter.

Balance this severity by communicating a willingness to forgive the perpetrator, as it's likely that they didn't understand the gravity of the leak. Emphasize the point that, should the person who committed the violation be willing to help remove the leak, you'll reopen the test and continue the project.

While this doesn't correct the situation, if it was truly an accident or misunderstanding (which is probable), then this provides a simpler path to putting this event in the past.

The more important result is that all of the other testers will do whatever they can to solve the mystery. It's important not to underestimate the ingenuity of a group of motivated testers, as there's a high probability that they'll identify the culprit.

4. Remove the Leaked Information

If the information hasn't gone viral (which is the case 9 times out of 10), and you've decided that it's detrimental enough to warrant efforts to have it removed, then your next step is to contact the leak location and ask for the data to be removed.

BUT WHAT IF IT'S GONE VIRAL?

If the information has spread between more than a handful of sites, chances are you'll have little success in removing it. Your best strategy would be to focus your strategy on how to benefit from the buzz.

Industry Sites and Blogs (Personal Reach)

Using the list of sites covering your leak, you'll want to address each one individually and keep notes on your progress.

Begin with a polite email or private message asking for their help in stopping the leak from spreading. It's very important that you communicate this with extreme care. Emails to site hosts should be respectful, clear, and concise. It's okay to explain why the leak is potentially harmful, so long as you feel that the person will understand and hopefully empathize.

Your email should clearly be written by an individual rather than an organization (i.e. no "legalese"). Chances are good that they're not looking to cause you pain (which is why it's important that your message is framed in a personal email), but your email should be written under the context that if they were to post it publicly, it would not magnify an already negative experience.

If they're unwilling to help, but the impact is great, you'll likely need to escalate the matter. Even small companies have C-level executives with impressive titles. Their help can sometimes help influence and move the process of removal.

KEEP IT PRIVATE

Never engage sites or individuals about beta leaks in a public way (Twitter, forums, etc.). Regardless of your intentions or right, the Internet is full of people who look at companies (even those that make the products they love most) as evil entities. The last thing you want to do is start a flame war, which would further intensify the spotlight on the issue.

Another option is to leverage the beta test itself as a possible relationship builder. You may want to offer them an opportunity to join the project or provide an exclusive preview when the product is complete if they will join your side. A little good will goes a long way when dealing with these kinds of issues, and this may have additional PR benefits as well.

If you've exhausted all other avenues, the absolute last resort is to threaten legal action (takedown notice, breach of contract, etc.). The reality is that it's extremely rare for a beta leak situation to escalate this far.

Large Sites and Search Engines (Takedown Requests)

Depending on the specific violation, there are a number of angles you can use to have beta information removed online, generally falling under the attributes of the DMCA (Digital Millennium Copyright Act). Given that the NDA is an agreement between you and your beta testers (e.g. the tester violated their contract, not the site), this message will likely have greater impact on most services.

Most larger sites and services deal with these types of issues (piracy, copyright infringement) regularly, and therefore have streamlined forms and processes for removing content in place, which you can use. The following are some of the most popular ones:

Google	http://support.google.com/bin/static.py?hl=en&ts=1114905&page=ts.cs
Bing	http://www.microsoft.com/info/FormForSearch.aspx
YouTube	http://www.youtube.com/yt/copyright/copyright-complaint.html
Twitter	https://support.twitter.com/forms/dmca
Facebook	https://www.facebook.com/help/contact/208282075858952

5. Find the Rat!

Discovering the source of your leak should always be your top priority. The sooner you identify the person, the sooner you can move past it. In addition, this will plug the hole, ensuring that this person is blacklisted from all future projects.

While a bit of detective work might sound complicated, it's usually pretty easy to nail down the culprit with limited information. Assuming it's not one of the larger consumer portals (Twitter, Facebook, etc.); Begin by approaching the site operators where the data was leaked. If it's a forum or blog, speak to the moderator or editor and explain your situation. Explain that any detail, no matter how small, is useful. While they may not have a lot of details, you'd be surprised how little information you actually need.

The first big clue in finding the source of the leak is location. If your test includes people from around the world, the pool of suspects narrows quickly if you can identify the geographic area of the user. If you're testing in a specific location, header data and IPs can still help you narrow the possible offending candidate.

Crosscheck the location against any telling data in the leak. Most people who do these kinds of things are not clever and/or dedicated enough to cover their trail. Look for distinctive hints in their text. The language of the post may contain clues to their identity. Very often, the offender will provide subtle bits of data that you can compare against data you gathered for the project.

REMEMBER, IT'S STILL UNLIKELY IT'S A BETA TESTER!

This is a good point to remind you that there is a higher-than-likely chance that the leak did not come from your beta testers. As covered earlier in this document, it's more likely that it came from an employee or partner. With this in mind, you actually have two goals. (1) Clear your beta testers (allowing you to ease the relationship and continue testing), and (2) find the real culprit (clearing your beta itself of blame!).

Once you believe you've identified the suspect, assess whatever profile data you have available for things that match the post. While it's unlikely that they used their real name (which you should have access to for all of your testers), it's extremely likely that some sort of handle or username is associated with the leaked information. Assuming that your beta system offers usernames, this will often quickly point to an answer.

If available, search for that username (with and without suspected real names) online. You'll find that most people use the same username for at least a couple of services, and often many. It's very likely that your offender has posted something somewhere before that provides telling clues about whether or not they leaked the information about your beta. Their Facebook account, Tumblr page, or Twitter feed might hold signs or even outright admissions of guilt. More often than not, these services will lead you to their real name.

ASK YOUR BETA TESTERS FOR HELP!

As mentioned previously, your beta testers are a crowd of highly-motivated people who can help identify the culprit. If you're light on resources and/or think it would help speed up the process, simply ask your beta tester team for help. Remember that the vast majority have done nothing wrong and are both invested in your project and want to clear their name. Why not use 50-100 helping hands in a time of need?

6. Decide What to Do With the Rat

Once you're confident that you've accurately identified the culprit you have a couple of tactical options. First, you need to decide if you're going to approach them at all.

If you've reached the end of your beta test and the leak had negligible impact, it may make sense to avoid a confrontation entirely. While this may seem counterproductive, the fact that you know, but they don't, actually provides you with a small strategic advantage. With that information you can silently blacklist them from future projects, without them knowing. If they actually were intending to leak (i.e. it wasn't an accident), they may try to sneak in again. If they don't know they've been caught then they'll likely apply for the next test, but simply not be accepted.

If you've not reached the end of the beta, or for whatever other reason you do feel it necessary to confront the leaker, you'll most likely initially do so via email, followed by phone calls if necessary. You'll want to send a stern message which provides little detail, but ensures they understand that you know. For the purposes of this email it is best to remain relatively vague, and leave as much to their imagination as possible. Let them sweat over the possibilities (e.g. legal action, black suits in windowless vans, etc.).

They need to understand that they've explicitly broken both their contract (NDA) and the rules of the project, and have been removed from the test, and all future testing opportunities. If the product is hardware, they need to return the hardware immediately.

They'll likely be both surprised (not knowing how you found them) and afraid. If the leak was an accident, and possibly even if it wasn't, there's a good chance they'll apologize profusely, and offer to help remove the leaked information as penance. While you should take them up on this offer (if it will help), they should still be blacklisted from all future opportunities.

IS LEGAL ACTION WORTH THE TROUBLE?

The short answer is "probably not." Taking legal action against an individual serves to accomplish little beyond sending an extremely expensive message. That said, this is not a decision you should make without the input of executive representation and or your legal counsel. The severity of the leak (e.g. Was it a competitor?) may have a greater impact that will need to be considered.



Why Beta Leaks Really Don't Matter Too Much

As we've already covered, there are a number of good reasons why maintaining secrecy in development is important. That said, in a traditional private beta test, the project is conducted only weeks before release on a product that is nearly complete. The following items match up to our initial list of concerns.

Direct Loss of Sales	If this threat is substantial, then the awareness is also increased (e.g. free marketing).
Delayed Sales	Most industries already offer free upgrades for recent buyers. Publicizing this will offset this risk.
Negative Perception	Beta should generally take place late enough in the cycle to produce a near-release-quality product, and under most circumstances, a limited audience will gain access to the beta.
Misinformation	Generally, beta happens late enough that major changes to the product are rare. In the event that outdated or inaccurate information is shared, this needs to be factored into in your response plan.
Message Control	While controlling the message until release is optimal, social buzz surrounding a beta can still be harvested for benefit.
Competitive Advantage	Your competitors may get a look at your product a few weeks earlier than planned, but it won't be enough time to pivot in a meaningful way. At best they'll be able to adjust their messaging, but given the timing, they would have likely done this anyway.
Security Exploitation	This specifically applies to few products, in which case beta security (and the guidelines in this document) should be simply enforced to a higher standard.

Lastly, your leaked beta sends a message. It tells the world that you care about quality. Beta testing demonstrates that your company implements customer input into your product designs. Marketing can use this fact to hint that the product is close to release, but that your company wants to ensure it is ready before it goes out into the world.

Leaks are definitely a concern, and all good beta programs should take steps to ensure that they don't happen. Using secrecy to help conduct a good beta test is a best practice. However, there is no reason for you not to turn that lemon into lemonade by leveraging the leak to help generate more interest in your product.

centercode

Delta Testing Solutions

Leading product ops teams use Centercode to identify real-world issues to fix, areas to improve, and what their users love in order to build amazing products.



Centercode's delta testing platform is built on decades of experience user testing the world's most successful products. Our platform equips you with the automation, best practices, and processes you need to get actionable user insights for any use case.

PRODUCT LAUNCH

Engage your **Target Market** to successfully bring new products and services to market

USE CASES:

- Beta Tests
- User Acceptance Tests
- Field Tests

PRODUCT GROWTH

Recruit **Existing Customers** to continuously mature live products and services

USE CASES:

- Progressive Delivery
- Early Access
- Customer Advisory Boards

CUSTOMER ZERO

Gather **Employee** feedback to enhance your products while fostering a product culture

USE CASES:

- Alpha Tests
- Dogfooding
- Employee Tests

COMPETITIVE EDGE

Enlist **Competitors' Customers** to rapidly gain actionable intel about their products

USE CASES:

- Competitive Intelligence
- Market Testing
- Comparative Research

DELTA TESTING PLATFORM

Get a live demo of Centercode to experience the easy way to run your user tests, including:

- Auto-generated test designs
- Ted, your tester engagement bot
- Smart feedback prioritization
- Instant KPIs and dashboards

SCHEDULE A DEMO

WHY CENTERCODE?

Increase Revenue
½ ☆ INCREASE
in product ratings

Release Sooner
50% FASTER
test cycles

Gain Participation
75% INCREASE
in tester engagement

Save Time
40% + DECREASE
in time managing tests